

**TITLE OF INVENTION: [A METHOD AND SYSTEM FOR MANAGING A GROUP-
SHARED PRIVATE KEY FOR MULTI-PARTICIPANT]**

REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the priority to Singapore patent application No. 10202301827Y, filed on 26 June 2023, the contents of which are incorporated herein by reference.

TECHNICAL FIELD

[0002] The present disclosure relates generally to a method and system for managing encryption keys in the fields such as cyber security, fintech and blockchain technology.

BACKGROUND

[0003] The following discussion of the background to the invention is intended to facilitate an understanding of the present invention only. It should be appreciated that the discussion is not an acknowledgement or admission that any of the material referred to was published, known or part of the common general knowledge of the person skilled in the art in any jurisdiction as at the priority date of the invention.

[0004] Blockchain technology has significantly impacted the way digital assets are managed and traded, offering new possibilities and efficiencies. Nonetheless, a critical issue in digital asset management is preventing the centralization of control over these assets by a single entity. To address this, there is a growing need for effective and secure methodologies that support joint control over digital assets, ensuring a decentralized and balanced approach.

[0005] The inclusion of multiple stakeholders in managing the same digital assets, however, introduces complexities such as increased risk, diminished transparency, security concerns, elevated costs, and limited flexibility. For instance, it becomes important for technical solutions to accommodate the participation of various entities, including those whose trustworthiness may be questionable or subject to changes over time, in using a shared control key. As a result, the system must be intricately designed to allow effective and collective management of this key, which is vital for safeguarding digital assets.

[0006] Additionally, maintaining fairness and impartiality among the involved parties is required at a technical level. This requires the integration of controls and mechanisms within the system's architecture, aimed at ensuring equal participation and preventing any party from acquiring disproportionate influence or control.

[0007] Another challenge in multi-participant key management is the secure and reliable recovery of keys if any participant loses their portion. Moreover, it is beneficial to control the costs associated with managing digital assets by multiple participants, such as costs related to implementation, operation and maintenance, so that the multi-participant digital asset management method can be economically viable.

[0008] Current approaches to digital asset management, such as multiple digital signature methods, blockchain contract-based multi-participant digital signature methods, and multi-participant private computation-based digital signature methods, exhibit several limitations, ranging from lack of adaptability and reliance on specific signature algorithms to security vulnerabilities, high costs, deployment complexities, and limited versatility. For instance, the Chinese patent application CN115001669A, titled "METHOD AND SYSTEM FOR PROCESSING PRIVATE KEY SHARDS," details a method and system for managing private key shards. This invention conducts the entire key and shard generation process within a Trusted Execution Environment (TEE), ensuring key confidentiality and accessibility exclusively through a hardware encryption chip. However, this confinement to the TEE may reduce the system's flexibility in dealing with complex digital asset transactions involving multiple participants with varied profiles and requirements. Similarly, the Chinese patent application CN113922957A, titled "VIRTUAL CLOUD WALLET BASED ON PRIVACY PROTECTION CALCULATION," introduces a virtual cloud wallet that operates key services within a TEE as well. This includes not only key generation but also managing the associations (gid, uid, password, key) of group identity, user identity, and password fragment of the private key. While this arrangement ensures security, having numerous key service functions within the TEE could potentially hinder the key management system's adaptability and versatility, particularly in scenarios involving groups with intricate participant relationships in digital asset management.

[0009] Therefore, there exists a demand for an enhanced method of multi-participant digital asset management that effectively caters to a diverse range of participants, each with unique profiles and needs. This method should aim to address and mitigate several of the technical challenges prevalent in existing solutions.

SUMMARY

[0010] Specifically, to address the above-mentioned technical problems, the present invention specifically uses the following technical solutions:

[0011] In accordance with one aspect of the present invention, there is provided a method for managing a Group-shared Private Key (GPK), comprising:

[0012] S101 transmitting, by an initiating Key Shard Holder (KSH) from a plurality of KSHs, a GPK generation request to a Communication Coordination Application (CCA); wherein each of the plurality of KSHs is in possession of a unique asymmetric key pair comprising a public key and an individual private key (IPK); wherein the GPK generation request is transmitted to the CCA together with the public key from the initiating KSH.

[0013] S102 creating, by the CCA upon receiving the GPK generation request, a key creation session to invite the plurality of KSHs to participate.

[0014] S103 accepting, by the CCA, participating KSHs from the plurality of KSHs to join the key creation session; and further receiving, by the CCA, the public keys from the participating KSHs.

[0015] S104 forwarding, by the CCA, the GPK generation request to a Trusted Execution Environment (TEE) together with all of the public keys received by the CCA, wherein the TEE comprises a TEE program.

[0016] S105 generating, by the TEE program, a GPK and multiple GPK shards based on the GPK.

[0017] S106 encrypting, by the TEE program, the multiple GPK shards with the public keys to obtain multiple encrypted GPK shards, wherein each of the encrypted GPK shards corresponds to a corresponding public key of a corresponding KSH.

[0018] S107 transmitting, from the TEE, the encrypted GPK shards to the CCA.

[0019] S108 creating, by the CCA upon receiving the encrypted GPK shards, a key distribution session to distribute the each of the encrypted GPK shards to its corresponding KSH.

[0020] In some embodiments, the CCA is operable to forward the GPK generation request to the TEE if number of the participating KSHs is above a first threshold count.

[0021] In some embodiments, the multiple GPK shards are generated from the GPK based on a Shamir's Secret Sharing algorithm.

[0022] In some embodiments, a first number of the multiple GPK shards is required to regenerate the GPK; wherein the first number is: (i) larger than or equal to two; and (ii) smaller than or equal to total number of the multiple GPK shards.

[0023] In some embodiments, the TEE is secured with hardware-based memory encryption.

[0024] In some embodiments, the each of the encrypted GPK shards is operable to be decrypted by the IPK of the corresponding KSH.

[0025] In some embodiments, the GPK shard is operable to be further sliced into a plurality of Secondary GPK Shards (SGS).

[0026] In some embodiments, a second number of SGS is required to regenerate the GPK shard; wherein the second number is: (i) larger than or equal to two; and (ii) smaller than or equal to total number of the multiple SGS generated.

[0027] In accordance with one aspect of the present invention, there is provided a system for managing a Group-shared Private Key (GPK), comprising: a plurality of Key Shard Holders (KSHs), wherein each of the plurality of KSHs is in possession of an unique asymmetric key pair, wherein the unique asymmetric key pair comprises a public key and an individual private key (IPK); a Communication Coordination Application (CCA); a Trusted Execution Environment (TEE); and a GPK generation application operating in the TEE.

[0028] Wherein, an initiating KSH from the plurality of KSHs is operable to transmit a GPK generation request to the CCA together with the public key from the initiating KSH; the CCA, upon receiving the GPK generation request, is operable to create a key creation session to invite and accept participating KSHs from the plurality of KSHs to join the key creation session, and to receive the public keys from the participating KSHs; the CCA is operable to forward the GPK generation request to the TEE together with all of the public keys received.

[0029] In the TEE, the GPK generation application, upon receiving the GPK generation request and the public keys, is operable to (i) generate a GPK and multiple GPK shards based on the GPK; (ii) encrypt the multiple GPK shards with the public keys to obtain multiple encrypted GPK shards, wherein each of the encrypted GPK shards corresponds to a

corresponding public key of a corresponding KSH; and (iii) transmit the encrypted GPK shards to the CCA.

[0030] The CCA, upon receiving the encrypted GPK shards, is operable to create a key distribution session to distribute the each of the encrypted GPK shards to its corresponding KSH.

[0031] In some embodiments, the CCA is operable to forward the GPK generation request to the TEE if number of the participating KSHs is above a first threshold count.

[0032] In some embodiments, the multiple GPK shards are generated from the GPK based on a Shamir's Secret Sharing algorithm.

[0033] In some embodiments, a first number of the multiple GPK shards is required to regenerate the GPK; wherein the first number is: (i) larger than or equal to two; and (ii) smaller than or equal to total number of the multiple GPK shards.

[0034] In some embodiments, the TEE is secured with hardware-based memory encryption.

[0035] In some embodiments, the each of the encrypted GPK shards is operable to be decrypted by the IPK of the corresponding KSH.

[0036] In some embodiments, the GPK shard is operable to be further sliced into a plurality of Secondary GPK Shards (SGS).

[0037] In some embodiments, a second number of SGS is required to regenerate the GPK shard; wherein the second number is: (i) larger than or equal to two; and (ii) smaller than or equal to total number of the plurality of SGS generated.

[0038] In accordance with one aspect of the present invention, there is provided a method for signing a request data using a Group-shared Private Key (GPK), comprising: S201 transmitting, by an initiating Key Shard Holder (KSH), a signing request and a request data, to a Communication Coordination Application (CCA); wherein the signing request comprises a first hash of the request data and a GPK shard of the initiating KSH. The request data includes detailed information regarding the actions planned by the initiating KSH, such as (i) initiating a financial transfer from a corporate account to a recipient's account, or (ii) issuing a sensitive operational command to a hardware device, which could involve, for example, activating a monitoring sensor or instructing a manufacturing device to commence operations upon receiving the specified operation command signed by the GPK.

[0039] S202 creating, by the CCA, a signing session to invite the plurality of KSHs to join the signing session as participating KSHs; wherein the CCA is operable to transmit the request data to the plurality of KSHs for the plurality of KSHs to decide whether to become the participating KSHs.

[0040] S203 transmitting, by each of the participating KSHs, an approval message to the CCA, wherein the approval message comprises a second hash of the request data and a GPK shard of the each of the participating KSHs.

[0041] S204 forwarding, by the CCA, the signing request and a plurality of the approval messages to a Trusted Execution Environment (TEE), wherein the TEE comprises a TEE program.

[0042] S205 authenticating, by the TEE program, the request data based on the first hash of the request data and a plurality of the second hashes of the request data. In some embodiments, the TEE program is operable to hash the request data to generate a third hash, and the authentication of the request data compares the first hash, the plurality of the second hashes and the third hash.

[0043] S206 regenerating, by the TEE program, the GPK based on the GPK shards extracted from the signing request and the plurality of the approval messages.

[0044] S207 signing, by the TEE program, the request data with the GPK to obtain a signed request data.

[0045] S208 transmitting, from the TEE, the signed request data to the CCA.

[0046] In some embodiments, the TEE is arranged to be data communication with a Key Management Service (KMS).

[0047] In some embodiments, the signing request is operable (i) to be encrypted by a KMS public key in S201, and (ii) to be decrypted by a KMS private key in S205.

[0048] In some embodiments, the approval message is operable (i) to be encrypted by a Key Management Service (KMS) public key in S203, and (ii) to be decrypted by a KMS private key in S205.

[0049] In some embodiments, the request data comprises digital asset transaction data or an operation data.

[0050] In some embodiments, the TEE is secured with hardware-based memory encryption.

[0051] In some embodiments, the CCA is operable to initiate the step of S204 if total number of the GPK shards received by the CCA is sufficient for regenerating the GPK.

[0052] In some embodiments, the KSH is operable to transmit a rejection message to the CCA if the KSH decides not to approval the request data.

[0053] In some embodiments, the GPK shard is assembled from a second number of Secondary GPK Shards (SGS); wherein SGS is obtained from further slicing of the GPK shard, and the second number is: (i) larger than or equal to two; and (ii) smaller than or equal to total number of SGS generated.

[0054] To those skilled in the art, it is clear that the present invention offers an efficient solution for encryption key management. This application is not solely confined to digital asset management but extends to other cyber security applications that involve the management of devices or systems requiring multi-participant control. An example of this would be an apparatus such as a monitoring sensor or an operational system like a surveillance network, which is activated only upon approval from a specified number of participants.

[0055] Other aspects and features of the present invention will become apparent to those of ordinary skill in the art upon review of the following description of specific embodiments of the invention in conjunction with the accompanying figures.

BRIEF DESCRIPTION OF THE DRAWINGS

[0056] In the figures, which illustrate, by way of non-limiting examples only, embodiments of the present invention,

[0057] **[Fig. 1a]:** a block diagram illustrating a system for managing a GPK, including generating the GPK, slicing the GPK and distributing the GPK shards.

[0058] **[Fig. 1b]:** a block diagram illustrating a system for managing a GPK shard, including slicing the GPK shard further into multiple SGS, and distributing the multiple SGS to their respective KSHs.

[0059] **[Fig. 2a]:** a flowchart illustrating operations for managing a GPK, including generating the GPK, slicing the GPK and distributing the GPK shards.

[0060] **[Fig. 2b]:** a flowchart illustrating operations for managing a GPK shard, including slicing the GPK shard further into multiple SGS, and distributing the multiple SGS to their respective KSHs.

[0061] **[Fig. 3a]:** a block diagram illustrating a system for using a GPK to sign a request data by multi-participants.

[0062] **[Fig. 3b]:** a block diagram illustrating a system for reconstructing a GPK shard using a plurality of SGS.

[0063] **[Fig. 4a]:** a flowchart illustrating operations for using a GPK to sign a request data by multi-participants.

[0064] **[Fig. 4b]:** a flowchart illustrating operations for using a plurality of SGS to reconstruct a GPK shard.

[0065] **[Fig. 5]:** a block diagram illustrating the slicing of a GPK into multiple GPK shards, and the further slicing of a GPK shard into a plurality of SGS.

DETAILED DESCRIPTION

[0066] Definitions

[0067] Throughout this document, unless otherwise indicated to the contrary, the terms “comprising”, “consisting of”, “having” and the like, are to be construed as non-exhaustive, or in other words, as meaning “including, but not limited to”.

[0068] Furthermore, throughout the document, unless the context requires otherwise, the word “include” or variations such as “includes” or “including” will be understood to imply the inclusion of a stated integer or group of integers but not the exclusion of any other integer or group of integers.

[0069] Throughout the description, it is to be appreciated that the term “asymmetric encryption”, also known as public-key cryptography, refers to the use of a key pair, comprising a public key and a private key. Its characteristics include the ability for data encrypted with the public key to only be decrypted by its corresponding private key. The public key can be shared openly, allowing anyone to encrypt data, but decryption remains exclusive to the private key holder.

[0070] Throughout the description, it is to be appreciated that the term "Key Management Service" (KMS) refers to a system designed for the management and protection of cryptographic keys used in data encryption processes. A KMS provides a platform for generating, storing, distributing, and retiring cryptographic keys. Its core objective is to enhance data security by maintaining the integrity and confidentiality of encryption keys, thereby ensuring that encrypted data remains protected against unauthorized access.

[0071] Throughout the description, it is to be appreciated that the term "Trusted Execution Environment" (TEE) refers to a secure, isolated, and independent computing area within a device. This environment operates separately from the device's main, and potentially untrusted, operating system. Its primary function is to provide a safe and confidential space for the processing and storage of private data and sensitive computations, especially in environments where security cannot be fully guaranteed by the operating system alone. The integrity and security of the TEE are typically reinforced through hardware-based mechanisms, ensuring robust protection against external threats and unauthorized access. In the context of the present invention, TEE can also refer to an independent, standalone device functioning collaboratively with a computing system.

[0072] Throughout the description, it is to be appreciated that the term "Shamir Key Slicing Algorithm" refers to a cryptographic method that employs Lagrangian polynomial interpolation for key slicing. This algorithm is designed to divide a cryptographic key into N distinct slices (also known as "shards"), and any T shards (where T is less than or equal to N) are sufficient to regenerate the complete key. This means that the loss or compromise of up to $N-T$ slices does not jeopardize the ability to recover the entire key. In other words, no single slice alone can compromise the key.

[0073] Throughout the description, it is to be appreciated that the term "Group-shared Private Key" (GPK) refers to a unique kind of private key that is collectively held by a group of users. This key is distributed among the users in such a way that each member possesses only a fragment of the entire GPK, often referred to as a key slice or key shard. In the context of the present invention, the GPK is guarded against public disclosure, and no individual user is permitted to possess the entirety of the GPK. Also, in the context of the present invention, the complete GPK can only be reconstructed when a required number of the GPK shards are brought together, thereby preventing any single user from unilaterally accessing or utilizing the full capabilities of the GPK.

[0074] Throughout the description, it is to be appreciated that the term "Individual Private Key" (IPK) refers to private key exclusively possessed by a single user. In alignment with the

principles of asymmetric encryption, each IPK is associated with a corresponding public key, data encrypted with the public key can only be decrypted using the corresponding IPK. In the context of the present invention, in contrast to the GPK, which is collectively held and accessed by multiple users, the IPK is entirely under the control of an individual user. This grants the user autonomous authority to utilize his or her IPK for various intended purposes independently, without the need for collaboration or consent from other users. For example, in the present invention, as the GPK shard is encrypted using the corresponding public key, a KSH is able to decrypt the encrypted GPK shard using its corresponding IPK.

[0075] Unless defined otherwise, all other technical and scientific terms used herein have the same meaning as is commonly understood by a skilled person to which the subject matter herein belongs.

[0076] Embodiment.1

[0077] In accordance with various embodiments of the present invention, a method and system are provided for managing a Group-shared Private Key (GPK). An initiating Key Shard Holder (KSH) from a plurality of KSHs, each possessing a unique asymmetric key pair, transmits a GPK generation request to a Communication Coordination Application (CCA). The CCA creates a key creation session, inviting the KSHs to participate and receive their public keys. The CCA forwards the GPK generation request and the public keys to a Trusted Execution Environment (TEE) which generates a GPK and multiple GPK shards based on the GPK. The TEE encrypts the GPK shards with the public keys to obtain encrypted GPK shards, each corresponding to a KSH's public key. The encrypted GPK shards are transmitted to the CCA, which creates a key distribution session to distribute each encrypted GPK shard to its corresponding KSH.

[0078] Specifically, referring to Fig. 1a and Fig. 2a, there is a plurality of KSHs (e.g., A, B, C in Fig. 1a), each holding a unique asymmetric key pair comprising a public key and an IPK. In the first step (S101), one KSH (i.e., "initiating KSH") initiates the GPK generation process by sending a request to generate a GPK to the CCA. The CCA is a software application that manages the communication between different entities, modules and parties involved in the process. The GPK generation request sent by the initiating KSH includes the public key of the initiating KSH, which is a part of the unique asymmetric key pair that the initiating KSH possesses. The other part of this pair is the IPK, which is kept secret by the initiating KSH.

[0079] In the subsequent step (S102), upon receiving the GPK generation request from

step (101), the CCA establishes a key creation session for other KSHs to join in creating the GPK. The CCA oversees this session, ensuring the involvement of all essential entities (e.g., for example, the number of the KSHs participating in the key creation session must exceed the predetermined threshold). In some embodiments, this key creation session is a designated timeframe during which the CCA actively invites the remaining Key Shard Holders (KSHs) to join in the GPK generation process. In some embodiments, wherein there are multiple subgroups on the KSHs, the CCA is operable to invite only a specific subgroup of the KSHs for the generation of the GPK. Additionally, in some embodiments, the CCA may impose a time limit for KSH participation (e.g., one hour), with those failing to respond within the timeframe missing the opportunity to be part of the GPK generation.

[0080] In some embodiments, the CCA is equipped to verify the GPK generation request prior to establishing a key creation session. This includes assessing the profile of the initiating KSH to determine their eligibility for initiating such a request. The profiles of KSHs might be pre-stored inside the CCA, or submitted together with the GPK generation request. For example, the CCA evaluates whether the initiating KSH is part of a division within a corporation that holds the requisite authority for requesting GPK generation. If the initiating KSH lacks the necessary authority, the CCA is programmed to reject the GPK generation request immediately. This verification step is integral to ensure that only authorized KSHs can commence the GPK generation process.

[0081] In some embodiments, the CCA is operable to evaluate the underlying rationale for GPK generation. This assessment involves verifying if the reason (submitted together with the GPK generation request) aligns with specific predetermined criteria, such as the use of the GPK for highly confidential digital transactions involving potentially untrustworthy parties or controlling the operation of a highly sensitive apparatus or system. Upon confirming that the criteria are met, the CCA proceeds to process the GPK generation request by initiating the key creation session. This step ensures that the GPK is generated for legitimate and critical purposes, aligning with the security protocols and objectives of the system.

[0082] In some embodiments, the KSHs are organized into multiple subgroups, with each subgroup potentially requiring its own GPK tailored to its members. In these scenarios, the CCA's decision to process a GPK generation request may hinge on the profiling of the specific subgroup involved. For instance, if an initiating KSH seeks a GPK for use within their subgroup, and this subgroup meets certain predetermined criteria (such as being responsible for managing high-risk digital transactions above a specified financial threshold), then the

CCA will give priority to this request. Additionally, understanding that the GPK is intended for a particular subgroup enables the CCA to direct invitations exclusively to KSHs within that subgroup, rather than to those outside it. This targeted approach prevents the dissemination of invitations to KSHs who are either irrelevant to the specific GPK or should ideally remain unaware of its existence, thereby further maintaining the confidentiality and relevance of the GPK generation process.

[0083] In step S103, the CCA is operable to facilitate the participation of KSHs in the key creation session. During this session, an activity of the CCA is to collect the public keys from the participating KSHs. These public keys are integral to the later stages of the process, especially for encrypting the GPK shards. Detailed in steps S106 – S108, these encrypted shards are then sent back to the respective KSHs. The encryption of each shard uses the specific public key of a KSH, ensuring that it can only be decrypted by that KSH's corresponding IPK. For example, the GPK shard (A) encrypted by the public key (A) of KSH (A) can only be decrypted the corresponding IPK (A), which is being safe-guarded by KSH (A). Other KSHs, such as KSH (B) and KSH (C), even if receiving the encrypted GPK shard (A), won't be able to access the GPK shard (A). This process effectively maintains the integrity and confidentiality of the generated GPK by ensuring that each KSH can only decrypt the GPK shard that is specifically intended for them.

[0084] In some embodiments, the CCA's responsibilities are not limited to merely collecting public keys from KSHs, but also include verifying the validity of these public keys received. Should any irregularities, flaws, or potential security issues be identified in a public key, the CCA is equipped to request a resubmission from the concerned KSH. In some embodiments, to support this function, the CCA is configured to incorporate a specific program tasked with monitoring and evaluating the details and status of the public keys. One advantage of this approach is the early identification and correction of problematic public keys during the key creation sessions. This preemptive action ensures that public keys, which could potentially compromise the GPK shard encryption, do not advance into the GPK generation process, thereby safeguarding the integrity and security of the overall GPK creation process.

[0085] In some embodiments, the CCA is operable to forward the GPK generation request to the TEE if number of the participating KSHs is above a first threshold count. In some embodiments, the first threshold is suggested by the initiating KSH, who knows the parties that they want to group-share the GPK. In such a case, the initiating KSH transmits the minimal number of the participating KSHs required to the CCA, which will then establish

such a key creation session to invite the relevant parties to join. In these embodiments, the duration of the key creation session can be adjusted (e.g., extended beyond the normal timeframe) to ensure all relevant parties have sufficient time to participate in creating the GPK.

[0086] In step S104, the CCA forwards the GPK generation request along with all the received public keys to the TEE, which is a secure environment where sensitive operations like key generation and encryption are performed. The TEE includes a specific program (TEE program) that carries out these operations. In some embodiments, the TEE is secured with hardware-based memory encryption.

[0087] In step S105, in the TEE, the TEE program generates the GPK and multiple GPK shards based on the GPK. The GPK shards are smaller parts of the GPK that are created using a secret sharing algorithm. Each shard is a piece of the GPK that can be combined with other shards to recreate the GPK.

[0088] In some embodiments, the multiple GPK shards are generated from the GPK based on a Shamir's Secret Sharing algorithm, and a first number (T) of the multiple GPK shards is required to regenerate the GPK; wherein the first number (T) is:

- (i) larger than or equal to two; and
- (ii) smaller than or equal to total number of the multiple GPK shards created.

[0089] In these embodiments, the Shamir's Secret Sharing algorithm is utilized to divide the complete GPK into N GPK shards (or GPK slices), using the Lagrange polynomial interpolation method. In this context, N corresponds to the number of public keys sent to the TEE, as well as the total count of KSHs involved in the GPK creation, including both initiating and participating KSHs. In these embodiments, any T of the N GPK shards are sufficient to reconstruct the complete GPK, where $2 \leq T \leq N$. This condition prevents any single KSH from using the GPK shard they possess to reconstruct the entire GPK, thereby enhancing security and ensuring fairness. Moreover, this arrangement allows for the resilience of the system against the loss or damage of a certain number (i.e., $N - T$) of key shards without compromising the recovery of the complete GPK. Essentially, each participant, whether initiating or participating, holds only a GPK shard, and accordingly no single participant has access to the full GPK. Furthermore, the utilization of the GPK for signing or encrypting data necessitates the amalgamation of at least T key shards from T different KSHs to reconstruct the GPK.

[0090] In some embodiments, the CCA is responsible for setting the specific values of N and T prior to forwarding the GPK generation request to the TEE, with these values being tailored according to the intended application of the GPK or a specific suggestion made by the initiating KSH (e.g., the initiating KSH specifically transmits an instruction data, indicating the N and T of the GPK to be created). For example, for high-risk or highly confidential transactions, the CCA typically involves a larger group of qualified KSHs in the GPK generation, leading to more GPK shards being created. Concurrently, the CCA sets a higher T value, indicative of the transaction's criticality; the more sensitive or risky the transaction or operation, the larger the number of shards required for GPK reconstruction. This approach may entail setting T to a figure close to, or matching, N in highly secure scenarios. On the other hand, for transactions with lower risk or value, N and T can be adjusted downwards, simplifying the process while ensuring adequate security. In such situations, a minimal count of GPK shards, say two or three, might suffice for reconstructing the GPK. However, it is evident to those skilled in the art that the calibration of these security measures, such as determining the specific values of N and T, can be managed within the TEE through its programming as well. However, there are scenarios where predefining N and T in the CCA can streamline the process. This approach reduces the programming tasks within the TEE, allowing it to focus primarily on more sensitive operations like generating the GPK and GPK shards, and encrypting data using the GPK. Thus, such an arrangement is able to further optimize the workflow, balancing security concerns with operational efficiency.

[0091] In step S106, the TEE program encrypts the GPK shards using the N public keys received from the CCA. Each GPK shard is encrypted with the corresponding public key of a KSH. This ensures that only the KSH with the corresponding IPK can decrypt the GPK shard later, after step S108. For example, KSH (A) (e.g., the Chief Financial Officer) holds an asymmetric key pair (A), comprising a public key (A) and a IPK (A). In step S106, the GPK shard (A) encrypted by the IPK(A) in the TEE is intended to be transmitted to KSH (A) eventually, who is able to decrypt the encrypted GPK shard (A) using the public key (A).

[0092] In step S107, the TEE transmits the encrypted GPK shards back to the CCA. The CCA receives these encrypted shards and prepares to distribute them to the corresponding KSHs.

[0093] Finally, in step S108, the CCA creates a key distribution session upon receiving the encrypted GPK shards. During this session, the CCA distributes each of the encrypted GPK shards to its corresponding KSH. The KSHs can then decrypt the GPK shards using their IPKs (i.e., each encrypted GPK shard is operable to be decrypted by the IPK of the

corresponding KSH), completing the process of generating the GPK and distributing the GPK shards. Specifically, a GPK shard (A) encrypted by the public key (A) from the initiating KSH (A) is distributed back to the initiating KSH (A), who holds the IPK (A) operable to decrypt the encrypted GPK shard (A).

[0094] In some embodiments, the CCA's role includes a task of verifying the integrity of GPK shards received. This verification involves ensuring that, for example, the number of GPK shards equals the expected count of N and verifying that each shard is properly encrypted. If no discrepancies are found, the CCA initiates the key distribution session. In cases where irregularities are detected, the CCA pauses the distribution process to investigate the causes. If the issue is identified as originating outside the TEE, the CCA either resolves it internally (such as addressing complications in the key creation session) or communicates the issue to the respective KSHs (for instance, errors in submitted public keys) to restart the GPK generation and slicing process. Alternatively, if the problem appears to stem from within the TEE, indicating a significant compromise in the system's security, the CCA then activates an alert to signal the need for immediate system inspection and maintenance. The CCA's function above helps avert the distribution of potentially compromised GPK shards to KSHs. By performing these initial checks, the CCA serves as a barrier, ensuring that all GPK shards distributed have been validated for security.

[0095] In certain embodiments, the CCA is operable to oversee KSH activities during the GPK creation phase, with a focus on detecting any unusual patterns of participation. For instance, the CCA is operable to track and identify whether certain registered participants (KSHs), recognized for their roles in managing digital assets or operation, have not been active in key creation sessions for a prolonged duration. In such cases, the CCA can either flag this inactivity for the system manager's review or, as an alternative approach in some embodiments, directly contact these inactive participants with a reminder. This reminder seeks to gather their input or address any uncertainties, for instance, verifying whether they have received prior invitations to key creation sessions. These measures allow the CCA to effectively monitor engagement levels and uncover any disparities between expected and actual participation in managing digital assets.

[0096] However, it is important to note that some degree of inactivity might be anticipated in certain roles. For example, a KSH like a company chairman might not be involved in regular transactional or operational management, thus remaining dormant most of the time is expected. Similarly, a KSH serving as a backup server, holding a GPK shard for use only in emergencies, would typically exhibit infrequent activity. Therefore, the CCA is designed to

evaluate a KSH's performance in light of their anticipated level of involvement, ensuring a nuanced and context-sensitive assessment of participation.

[0097] As shown in Fig. 1b, Fig. 2b and Fig. 5, in some embodiments, a GPK shard can be further sliced into several Secondary GPK Shards (SGS), utilizing a slicing method similar to that employed for the GPK itself. In this scenario, a KSH who holds a GPK shard is linked to a group of secondary KSHs. Take, for instance, secondary KSHs who are part of a specific business division. The division head, acting as a primary KSH, is the sole holder of the complete GPK shard. However, their availability to approve transactions or operations by submitting this GPK shard for GPK reconstruction may not be constant. To address this, the division's GPK shard can be further sliced into multiple SGS, which are then distributed among the secondary KSHs in the division (e.g., division members working under the division head). Consequently, when the division head is unavailable, a specific number of division members, each possessing an SGS, can collectively reconstruct the GPK shard for its submission and subsequent GPK reconstruction.

[0098] In the process of obtaining an SGS, a division member (KSH B1) may, in some embodiments, submit a Further Slicing Request (FSR) to the CCA (S301). The CCA, upon receiving the FSR, commences a Further Slicing Session, inviting other members of the division (KSHs B2, B3, B4 ... B7) to join (S302). Both the initiating secondary KSH (KSH B1) and the participating KSHs are required to submit their public keys to the CCA (S303). These public keys, along with the GPK shard, are then forwarded to the TEE (S304). The TEE program is configured to slice the GPK shard into a plurality of SGS (S305) and encrypt them using the submitted public keys (S306). For example, the SGS (B1) encrypted with public key (B1) is sent to the CCA and further distributed by the CCA to secondary KSH (B1) (S307 – S308), who has the corresponding IPK (B1) necessary to decrypt this encrypted SGS (B1). This setup ensures a secure and distributed mechanism of control within a division, especially in the absence of the primary KSH.

[0099] In some embodiments, acceptance of a FSR by the CCA are contingent upon authorization from the primary KSH, such as a division head or an individual with higher authority within the system. This authorization is necessary for the further slicing of the GPK shard into smaller segments (i.e., SGS). In other embodiments, where no specific hierarchy exists within a division, no single member is permitted to possess the complete GPK shard. Consequently, the management of the GPK shard mirrors that of the GPK itself, with no individual entity being capable of reconstructing the entire GPK shard independently. A specified minimum number of SGS is required to reconstruct the GPK shard, ensuring

distributed control and security. Additionally, this system is designed to withstand the loss of some SGS without compromising the functionality or security of the GPK shard, thus further maintaining the robustness and resilience of key management process.

[00100] In some embodiments, a SGS can also be sliced into even smaller segments. This multi-level slicing approach creates a hierarchical structure within the GPK fragments (such as GPK shards, SGS, and their further fragments), where certain GPK fragments are designated as having "higher priority" over those at lower levels. In this structure, fewer high-priority GPK fragments (e.g., T GPK shards) might suffice to reconstruct the GPK. Conversely, a greater number of GPK fragments (e.g., $T1 > T$) might be necessary to reconstruct the GPK, especially when SGS are used to first assemble a GPK shard.

[00101] This layered system enhances GPK management's versatility and control, enabling more detailed access and operation. For instance, in situations requiring quick GPK reconstruction, like signing an urgent transaction or activating an urgent hardware operation, primary KSHs can use their GPK shards to expedite the process (e.g., two GPK shards from high-ranking officers like the chairman and CEO are enough to reconstruct the GPK). For less critical tasks where senior management's involvement is not essential, division members can use their SGS to form a GPK shard, possibly combined with another GPK shard assembled by a different division's SGS. These GPK shards can then be collectively used for authorizing actions. Therefore, high-priority GPK fragments, typically the primary GPK shards, are generally allocated to key KSHs holding significant authority or having critical roles in managing digital transactions or operations with the GPK. This allocation method ensures that crucial decision-makers or essential personnel can rapidly compile the GPK when needed, boosting the system's efficiency and responsiveness in urgent or high-importance situations.

[00102] Embodiment.2

[00103] In accordance with various embodiments of the present invention, a method and system are provided for signing request data using a GPK, which can be generated using the methods as described in Embodiment.1. Briefly, an initiating KSH transmits a signing request, together with the request data, to the CCA, wherein the signing request comprises a first hash of the request data, and a GPK shard. The CCA creates a signing session, inviting KSHs to participate. Each participating KSH transmits an approval message, including a second hash of the request data and a GPK shard of the each of the participating KSHs, to the CCA. The CCA further forwards the signing request and approval messages to the TEE. The TEE program authenticates the request data based on the hashes received,

regenerates the GPK from the GPK shards received, and signs the request data with the regenerated GPK to obtain signed request data. The signed request data is then transmitted to the CCA for further process (e.g., transaction of digital assets, or execution of an operation command on a specific hardware device).

[00104] Specifically, referring to Fig. 3a and Fig. 4a, in the first step (S201), an initiating KSH initiates a signing request. This signing request is associated with a request data for signing (e.g., digital asset transaction data or device operation command). The signing request comprises a first hash of the request data, and a GPK shard that the initiating KSH holds. The initiating KSH uses an encryption management service to generate the first hash of the request data. The initiating KSH then sends this signing request to a CCA, together with the request data itself.

[00105] In the second step (202), the CCA, a software pivotal in managing communications among KSHs, initiates a signing session. This signing session is designed to facilitate collective decision-making regarding the request data (e.g., whether to approve a financial transaction or activate an operation on a hardware device). In some embodiments, the session takes the form of a virtual meeting, where KSHs, excluding the initiating KSH, are invited to review the request data. In some embodiments, the CCA actively dispatches invitations along with the request data to the KSHs. This approach ensures that each KSH has an opportunity to review and decide whether to endorse the request data. Those KSHs decide to approve the request data will then become the participating KSHs to the signing session by transmitting their approval messages to the CCA. Those KSHs that decide to reject the request data will transmit a rejection message to the CCA or take no action.

[00106] In some embodiments, the CCA is equipped to assess the profiles of these initiating KSHs. If these initiating KSHs are identified as holding higher authority or are prioritized in managing digital financial transactions with the GPK (e.g., Chief Technology Officer in charge of operating a hardware device), the CCA is programmed to expedite the establishment of a signing session, and/or to proactively urge the other KSHs to review and approve the request data by proactively disseminating the invitation messages to these KSHs, and repeatedly in some embodiments. Furthermore, in some embodiments, the CCA evaluates the stated reasons behind the signing request or the details of the request data, wherein the stated reasons are indicated in the request data. This evaluation informs the CCA's decision on whether to proceed with establishing the signing session, based on the nature and urgency of the stated reasons. This functionality of the CCA ensures a responsive and dynamic approach to handling signing requests, aligning its priorities with the importance

and urgency indicated by the initiating KSHs and the context of the request data.

[00107] In certain embodiments, the CCA is programmed for efficient facilitation of the request data signing process. This involves closely tracking the count of both approval and rejection messages received from KSHs. If the number of rejection messages surpasses a predefined threshold – indicating that even with full approval from the remaining KSHs, there would still be insufficient GPK shards to reconstruct the GPK – the CCA is then set to suspend the signing session and send a notification of the failed signing attempt to the initiating KSHs. On the other hand, if a KSH with high authority in GPK-based digital asset transaction signing is involved, the CCA might opt to extend the invitation to a wider group of KSHs, beyond those initially contacted, instead of halting the session. This is particularly considered if not all potential KSHs have yet been notified in the first round. Furthermore, in some embodiments, upon receiving a rejection message, the CCA can prompt the rejecting KSH to reassess their decision to participate, especially if the request originates from an initiating KSH who holds a higher authority or priority level compared to the KSHs who initially rejected the request. For instance, consider a scenario where the initiating KSH holds a high-ranking position, such as the chairman of the board or a senior management member, and the KSH who rejects the request data is a mid-level manager overseeing a specific business division. This distinction in hierarchy helps prevent unintended rejections of request data, where a lower-ranked KSH might not fully grasp the significance of the request or could inadvertently submit a rejection message due to error or misunderstanding. This adaptive approach by the CCA helps facilitate participation for achieving a successful signing session, accounting for the dynamics of authority and priority among the KSHs.

[00108] In the third step (203), the participating KSHs, who have decided to join the encryption session, send an approval message to the CCA. In some embodiments, this approval message includes a second hash of the request data and a GPK shard that each participating KSH retains. The second hash of the request data is generated by the participating KSH using a hashing method similar to the first hash generated by the initiating KSH.

[00109] In step 204, the CCA is operable to forward the signing request and the approval messages to the TEE, where highly sensitive encryption occurs. In some embodiments, the CCA's role includes assessing the number of received GPK shards to confirm that they are sufficient for reconstructing the complete GPK, pending successful authentication of the request data in the TEE. Additionally, the CCA checks for any irregularities in the approval messages. This includes verifying if each approval message is correctly encrypted (e.g., with

the correct public key from the correct KSH). If an irregularity is detected in an approval message, the CCA is operable to request the concerned KSH to review and possibly resubmit their approval. This step by the CCA helps ensure that the signing process proceeds smoothly, with all necessary validations and authentications in place before the encryption stage in the TEE.

[00110] In the fifth step (205), a TEE program operating in the TEE verifies the request data pending to be signed. This is done by comparing the first hash of the request data (from the initiating KSH) and the second hashes of the request data (from the participating KSHs). If these hashes match, it means that the request data, received and reviewed by the initiating KSH and the participating KSHs, is the same and has not been tampered with during the transmission.

[00111] In some embodiments, the TEE program also hashes the request data to generate a third hash, and the verification of the request data involves checking whether all hashes, including the first hash, the second hashes and the third hash generated by the TEE program, match. If all hashes correctly match, the TEE program then proceeds to the next step – reconstruction of the GPK.

[00112] In some embodiments, the present invention is arranged to be associated with a Key Management System (KMS), which includes an asymmetric key pair of KMS public key and KMS private key. In some embodiments, the signing request is operable (i) to be encrypted by a KMS public key in S201, and (ii) to be decrypted by a KMS private key in S205. In some embodiments, the approval message is operable (i) to be encrypted by a KMS public key in S203, and (ii) to be decrypted by a KMS private key in S205. In some embodiments, the KMS is arranged to be in data communication with the TEE, and also secured with hardware-based memory encryption. In some embodiments, the request data remains unencrypted to enable various participants in the signing process to review and examine it. In these specific cases, where the first hash, second hash, and GPK shards are all encrypted using the KMS private key, it ensures that no participant outside the TEE has access to or the ability to alter the created hashes and GPK shards necessary for reconstructing the GPK.

[00113] As both the signing request and the approval message are encrypted by the KMS public key, the TEE is able to decrypt the signing request and the approval message using the KMS private key, obtaining the hashes of the request data, as well as the GPK shards. In this step, the TEE compare the hashes, including the first hash, the second hash and the third hash. If all these hashes correctly match, the authenticity of the request data, approved

by multiple KSHs, can be acknowledged.

[00114] In the sixth step (206), the TEE program reconstructs the GPK in the TEE. This is done by combining the GPK shards extracted from the signing request and the approval messages.

[00115] In the seventh step (207), the reconstructed GPK is then used to sign the request data, by the TEE program, to obtain the signed request data. In some embodiments, the TEE program converts the request data into an instruction command executable by entities downstream (e.g., instructing a transaction to be conducted over the blockchain or by a digital bank; or instructing a specific hardware device to conduct a certain operation), and creates a signature data based on the request data. The signature data is associated with the instruction command. In other words, the signed request data comprises both the instruction command and the signature data. In the absence of the signature data, the instruction command is inexecutable at the downstream, e.g., transaction of digital assets or apparatus activation will be rejected.

[00116] In the final step (208), the TEE program returns the signed request data to the CCA, which then forwards the signed request data for the intended purposes, e.g., initiating a digital financial transaction or apparatus activation as specified by the request data. In some embodiments, the signed request data will be checked by the CCA, and if any irregularities are detected (e.g., missing signature data), the CCA will stop the transmission of the signed request data, and deny any execution initiated by the signed request data.

[00117] In some embodiments, referring to Fig. 3b and Fig. 4b, a secondary KSH receiving an invitation from the CCA for joining a signing session and approving the request data. However, lacking a complete GPK shard, this secondary KSH is unable to independently participate in the signing session. To join, the secondary KSH must collaborate with other secondary KSHs in their group (for example, those in the same business division) to collectively reassemble the GPK shard assigned to their group. Upon invitation from the CCA, the secondary KSH submits a shard assembly request (S401), which includes their encrypted Secondary GPK Shards (SGS) and public key, using a KMS public key.

[00118] In response to this request, the CCA initiates a shard assembly session (S402), inviting other secondary KSHs within the same division to join. The secondary KSHs, willing to contribute their SGS towards reconstructing the GPK shard, then participate in the signing session by sending their approval messages to the CCA. These approval messages,

comprising their SGS and public keys and encrypted with a KMS public key, are required for the shard assembly process (S403).

[00119] Once the CCA receives a sufficient number of approval messages, indicating enough SGS to reconstruct the GPK shard, it forwards the shard assembly request and all received approval messages to the TEE (S404). Inside the TEE, the program authenticates the SGS by decrypting the shard assembly request and approval messages with the KMS private key (S405). Successfully authenticated, the TEE program then extracts and combines the SGS to recreate the GPK shard (S406). This regenerated GPK shard is sent back to the CCA (S407), which then uses it, along with other GPK shards, to regenerate the complete GPK (S408). This process ensures secure collaboration among secondary KSHs for effective participation in the signing session.

[00120] Embodiment.3

[00121] In accordance with one aspect of the present invention, there is provided a generic multi-participant participatory digital signature method, characterized by: Generate the complete key in the trusted execution environment, including: (a) generating complete keys only in the trusted execution environment; (b) use Shamir key slicing technique for private key slicing; and (c) asymmetric encryption of the key slice information.

[00122] In some embodiments, step (a) generates the complete key only in the trusted environment, while the trusted execution environment uses hardware- related mechanisms that can guarantee the security of the generated key.

[00123] In some embodiments, the Shamir key slicing technique in step (b) processes the private key slicing, which further comprises: (i) Slice using the Lagrange polynomial interpolation algorithm to divide the complete private key into N key slices, where any T key slices can be used to recover the complete private key, $T \leq N$, to ensure the security and fairness of the key; (ii) Allow the specified number of key slices to be lost without affecting the complete private key recovery; (iii) Each of the multiple participants holds the key slice information, and no party can hold the complete private key; and (iv) the completion of the signature of the data by multiple participants requires the completion of at least T key slice holders to be specified.

[00124] In some embodiments, the key slice information in step (c) is encrypted in an asymmetric manner to ensure the security of the transmitted critical data.

[00125] In some embodiments, the slice generated in step (a) based on Shamir key slicing technique does not contain any information about the key and the actual key information is not transmitted on the network, so the key is not compromised.

[00126] In some embodiments, the specified number of key slices lost in step b can be adjusted according to the specific application requirements to meet the requirements of disaster tolerance and fault tolerance.

[00127] In some embodiments, if any party in step (c) is unable to hold the complete private key and recover the complete key, the control of the digital asset will not be controlled by a single subject.

[00128] In some embodiments, at least T key slice holders need to be specified in step d, indicating that the specified number of key slice holders are allowed to jointly participate in completing the operation of using the key. to meet the requirement of multi-participant collaborative management.

[00129] In some embodiments, the method further comprises: (i) the key slicing client creates its own asymmetric encryption public and private keys, and the trusted execution environment program uses its public key to encrypt key data to the key slicing client; and (ii) KMS provides the asymmetric encryption public key, and the key slicing client uses its public key to encrypt key data to the trusted execution environment program.

[00130] In some embodiments, after the trusted execution environment program in step a encrypts the key data using the public key provided by the key slicing client, only the corresponding private key provider can decrypt the data, and others cannot decrypt the key data even if they get the data, so the key data will not be leaked and tampered with.

[00131] In some embodiments, the key slicing client in step (b) does hash calculation on the provided application data and then merges it with the private key slice. After the data is encrypted with the asymmetric encryption public key provided by KMS, only the trusted execution environment uses the decryption service provided by KMS to decrypt the data, and others cannot decrypt the key data even if they get the data, so the key data will not be leaked and tampered with. At the same time, the trusted execution environment program can ensure that the application data provided by the key slicing client will not be tampered with by performing hash matching.

[00132] In some embodiments, the KMS must be subject to strict authorization control and used only to designate a trusted execution environment to decrypt the data.

[00133] In some embodiments, since only the trusted execution environment can be decrypted, the recovery of the complete key is performed only in the trusted execution environment, and therefore the key is not compromised.

[00134] In some embodiments, since only the trusted execution environment can be decrypted, the use of the key takes place only in the trusted execution environment, and therefore the key is not compromised.

[00135] In some embodiments, Trusted Execution Environment (TEE) perform the following activities: Generate key; Generating key slices using Shamir key slicing technology; Use key slice client public key encryption for critical data; Communication with the communication coordinator; Communicate with KMS to decrypt the data provided by the key slicing client; Extracting the key slice to calculate the full key; Extract data for processing (e.g. signature and other private key related operations).

[00136] In some embodiments, Key Management Service (KMS) performs the following activities: Providing asymmetric decryption with strict authorization; Decrypt data encrypted with the KMS public key from the key splitting client; Secure critical data provided by key slicing clients to trusted environments.

[00137] In some embodiments, Key Shard Holder (KSH) performs the following activities: Accepts the key slice distributed by the trusted execution environment and stores it encrypted locally; The key slicing client, as a participant, needs to verify the data and calculate the hash of the data; Combine hash and key slice and encrypt with KMS public key; Communication with the communication coordinator; and Key slicing client generates its own asymmetric encryption public-private key.

[00138] In some embodiments, Communication Coordination Application (CCA) performs the following activities: Communication with trusted execution environment programs; Communicating with the key slicing client; Forwarding data from the above two parties to assist in specific business functions.

[00139] In some embodiments of the present invention, there is provided The Private key creation and distribution process for managing digital assets: a key slicing client initiates a request to the communication orchestrator containing a private key slicing rule, requesting the creation of a private key for managing digital assets. The private key slicing rule is: the number of private key slices is N , at least T ($T \leq N$) slices are allowed to be combined into a complete private key, and the number of private key slices allowed to be lost is $N-T$. The

communication coordinator caches the request and data and creates a session that accepts other key slicing clients to join. N key slicing clients, join the session, confirm the private key slicing rules, and transmit their respective public keys to the communication coordinator. The communication orchestrator generates a private key creation request using the slicing rules and the public keys of these N key slicing clients, which is forwarded to the trusted execution environment program to request the creation of a private key for managing digital assets. After the trusted execution environment program receives the request for private key creation, it creates a private key for managing digital assets and slices the private key using Shamir private key slicing technology, and then encrypts the generated private key slices with the public keys of these N key slicing clients respectively, and returns the result to the communication coordination program. N key slice clients receive the private key slice encrypted data returned from the trusted environment from the communication coordination program, and then decrypt the private key slice assigned to them with their respective private keys, completing the private key creation and distribution process for managing digital assets.

[00140] In some embodiments of the present invention, the private key is used to manage digital assets signs the transaction data process: A key slicing client initiates a digital asset transaction request, hashes the requested transaction data, then merges this hash with its own saved private key slice, encrypts the data with the KMS public key used by the trusted execution environment and transmits it together with the transaction data to the communication coordinator, which caches the request and data of the transaction and notifies other key slicing clients. After receiving the transaction request, the other key slice clients check the digital asset transaction data separately, and if they agree to execute it, they will hash the transaction data, then merge this hash with the private key slice they keep (the private key slice used for managing digital assets), encrypt this data with the KMS public key used by the trusted execution environment and transmit it to the communication coordination program together with the transaction data. The communication coordinator waits for the number of agreed requests to reach T and forwards them to the trusted execution environment program for execution. Conversely, if the number of rejected slice holders is greater than $N - T$, the transaction request is cancelled. The Trusted Execution Environment program uses KMS to decrypt the data submitted by the key slicing client. The trusted execution environment program verifies whether the hash submitted by all key slicing clients and the transaction data hash are consistent, and if they are consistent, it uses the private key slicing to calculate the complete private key and sign the transaction data, otherwise it refuses to execute the signature. Trusted execution environment program

returns the result to the communication coordination program, and completes the process of managing the private key of digital assets to sign the transaction data.

[00141] It should be further appreciated by the person skilled in the art that variations and combinations of features described above, not being alternatives or substitutes, may be combined to form yet further embodiments falling within the intended scope of the invention.

[00142] As would be understood by a person skilled in the art, each embodiment, may be used in combination with other embodiment or several embodiments.

Claims

Claim 1. A method for managing a Group-shared Private Key (GPK), comprising:

S101 transmitting, by an initiating Key Shard Holder (KSH) from a plurality of KSHs, a GPK generation request to a Communication Coordination Application (CCA);

wherein each of the plurality of KSHs is in possession of an unique asymmetric key pair comprising a public key and an individual private key (IPK);

wherein the GPK generation request is transmitted to the CCA together with the public key from the initiating KSH;

S102 creating, by the CCA upon receiving the GPK generation request, a key creation session to invite the plurality of KSHs to participate;

S103 accepting, by the CCA, participating KSHs from the plurality of KSHs to join the key creation session; and further receiving, by the CCA, the public keys from the participating KSHs;

S104 forwarding, by the CCA, the GPK generation request to a Trusted Execution Environment (TEE) together with all of the public keys received by the CCA, wherein the TEE comprises a TEE program;

S105 generating, by the TEE program, a GPK and multiple GPK shards based on the GPK;

S106 encrypting, by the TEE program, the multiple GPK shards with the public keys to obtain multiple encrypted GPK shards, wherein each of the encrypted GPK shards corresponds to a corresponding public key of a corresponding KSH;

S107 transmitting, from the TEE, the encrypted GPK shards to the CCA; and

S108 creating, by the CCA upon receiving the encrypted GPK shards, a key distribution session to distribute the each of the encrypted GPK shards to its corresponding KSH.

Claim 2. The method according to Claim 1, wherein the CCA is operable to forward the GPK generation request to the TEE if number of the participating KSHs is above a first threshold count.

Claim 3. The method according to Claim 1, wherein the multiple GPK shards are generated from the GPK based on a Shamir's Secret Sharing algorithm.

Claim 4. The method according to Claim 3, wherein a first number of the multiple GPK shards is required to regenerate the GPK; wherein the first number is:

- (i) larger than or equal to two; and
- (ii) smaller than or equal to total number of the multiple GPK shards.

Claim 5. The method according to Claim 1, wherein the TEE is secured with hardware-based memory encryption.

Claim 6. The method according to Claim 1, wherein the each of the encrypted GPK shards is operable to be decrypted by the IPK of the corresponding KSH.

Claim 7. The method according to Claim 1, wherein the GPK shard is operable to be further sliced into a plurality of Secondary GPK Shards (SGS).

Claim 8. The method according to Claim 7, wherein a second number of SGS is required to regenerate the GPK shard; wherein the second number is:

- (i) larger than or equal to two; and
- (ii) smaller than or equal to total number of the plurality of SGS generated.

Claim 9. A system for managing a Group-shared Private Key (GPK), comprising:

a plurality of Key Shard Holders (KSHs), wherein each of the plurality of KSHs is in possession of a unique asymmetric key pair, wherein the unique asymmetric key pair comprises a public key and an individual private key (IPK);

a Communication Coordination Application (CCA);

a Trusted Execution Environment (TEE); and a GPK generation application operating in the TEE, wherein:

an initiating KSH from the plurality of KSHs is operable to transmit a GPK generation request to the CCA together with the public key from the initiating KSH,

the CCA, upon receiving the GPK generation request, is operable to create a key creation session to invite and accept participating KSHs from the plurality of KSHs to join the key creation session, and to receive the public keys from the participating KSHs;

the CCA is operable to forward the GPK generation request to the TEE together with all of the public keys received;

in the TEE, the GPK generation application, upon receiving the GPK generation request and the public keys, is operable to

- (i) generate a GPK and multiple GPK shards based on the GPK;
- (ii) encrypt the multiple GPK shards with the public keys to obtain multiple encrypted GPK shards, wherein each of the encrypted GPK shards corresponds to a corresponding public key of a corresponding KSH; and,
- (iii) transmit the encrypted GPK shards to the CCA;

the CCA, upon receiving the encrypted GPK shards, is operable to create a key distribution session to distribute the each of the encrypted GPK shards to its corresponding KSH.

Claim 10. The system according to Claim 9, wherein the CCA is operable to forward the GPK generation request to the TEE if number of the participating KSHs is above a first threshold count.

Claim 11. The system according to Claim 9, wherein the multiple GPK shards are generated from the GPK based on a Shamir's Secret Sharing algorithm.

Claim 12. The system according to Claim 9, wherein a first number of the multiple GPK shards is required to regenerate the GPK; wherein the first number is:

- (i) larger than or equal to two; and

- (ii) smaller than or equal to total number of the multiple GPK shards.

Claim 13. The system according to Claim 9, wherein the TEE is secured with hardware-based memory encryption.

Claim 14. The system according to Claim 9, wherein the each of the encrypted GPK shards is operable to be decrypted by the IPK of the corresponding KSH.

Claim 15. The method according to Claim 9, wherein the GPK shard is operable to be further sliced into a plurality of Secondary GPK Shards (SGS).

Claim 16. The method according to Claim 15, wherein a second number of SGS is required to regenerate the GPK shard; wherein the second number is:

- (i) larger than or equal to two; and
- (ii) smaller than or equal to total number of the plurality of SGS generated.

Claim 17. A method for signing a request data using a Group-shared Private Key (GPK), comprising:

S201 transmitting, by an initiating Key Shard Holder (KSH), a signing request and a request data, to a Communication Coordination Application (CCA); wherein the signing request comprises a first hash of the request data and a GPK shard of the initiating KSH;

S202 creating, by the CCA, a signing session to invite the plurality of KSHs to join the signing session as participating KSHs; wherein the CCA is operable to transmit the request data to the plurality of KSHs for the plurality of KSHs to decide whether to become the participating KSHs;

S203 transmitting, by each of the participating KSHs, an approval message to the CCA, wherein the approval message comprises a second hash of the request data and a GPK shard of the each of the participating KSHs;

S204 forwarding, by the CCA, the signing request and a plurality of the approval messages to a Trusted Execution Environment (TEE), wherein the TEE comprises a TEE program;

S205 authenticating, by the TEE program, the request data based on the first hash of the request data and a plurality of the second hashes of the request data;

S206 regenerating, by the TEE program, the GPK based on the GPK shards from the signing request and the plurality of the approval messages;

S207 signing, by the TEE program, the request data with the GPK to obtain a signed request data; and

S208 transmitting, from the TEE, the signed request data to the CCA.

Claim 18. The method according to Claim 17, wherein the TEE is arranged to be data communication with a Key Management Service (KMS).

Claim 19. The method according to Claim 18, wherein, the signing request is operable

- (i) to be encrypted by a KMS public key in S201, and
- (ii) to be decrypted by a KMS private key in S205.

Claim 20. The method according to Claim 18, wherein, the approval message is operable

- (i) to be encrypted by a Key Management Service (KMS) public key in S203, and
- (ii) to be decrypted by a KMS private key in S205.

Claim 21. The method according to Claim 17, wherein the request data comprises digital asset transaction data or an operation command.

Claim 22. The method according to Claim 17, wherein the TEE is secured with hardware-based memory encryption.

Claim 23. The method according to Claim 17, wherein, the CCA is operable to initiate the step of S204 if total number of the GPK shards received by the CCA is sufficient for regenerating the GPK.

Claim 24. The method according to Claim 17, wherein, the KSH is operable to transmit a rejection message to the CCA if the KSH decides not to approval the request data.

Claim 25. The method according to Claim 17, wherein the GPK shard is assembled from a second number of Secondary GPK Shards (SGS); wherein SGS is obtained from further slicing of the GPK shard, and the second number is:

- (i) larger than or equal to two; and
- (ii) smaller than or equal to total number of SGS generated.