

SYSTEMS AND METHODS FOR TRUSTLESS DISTRIBUTED SYMMETRIC ENCRYPTION

Technical Field

[0001] The embodiments disclosed herein relate to cryptography and encryption protocols, and more particularly to systems and methods for cryptography symmetric encryption without a trusted third party.

Introduction

[0002] The basis for secret-sharing schemes were described in 1979 by Adi Shamir's article *How to Share a Secret* (Shamir, A., Communications of the ACM, 1979, vol. 22, is. 11, pp. 612-613). The article introduces the notion of a secret embedded in a polynomial of degree $k-1$, where k is the number of participants required to recover a secret. In the described scheme, the degree $k-1$ polynomial is evaluated at n points, and each evaluation is given to a different participant, a subset of whom can then jointly recover the secret with Lagrange interpolation.

[0003] This article opened the door for many new areas of research. Today, the sum of that research is jointly known as work on *threshold cryptography*, or the study of cryptographic systems where a group of n participants are involved, from which a subset k participants are required to perform operations (typically encryption and decryption).

[0004] Of particular interest is the research on Distributed Pseudo-Random Functions (DPRF for short), where a group of participants collaborate to calculate the seemingly random output of a function. DPRFs on their own refer to schemes where all participants are involved, and any subgroup of participants is unable to obtain the right output.

[0005] Naor, Pinkas and Reingold's 1999 article *Distributed Pseudo-Random Functions and KDCs* (Naor, M., et al., Eurocrypt'99 International Conference: Lecture Notes in Computer Science, 1999, vol. 1592, pp. 327-346) introduced the concept of *threshold DPRF*, where the output of the DPRF may be correctly evaluated by a subgroup of k participants among a total of n participants, whereas a subgroup of up to $k-1$ participants are unable to do so. In this scheme, only a subgroup of fixed size of

participants is required to properly evaluate the function. This work is a component of the invention described here.

[0006] While research on threshold encryption is a fairly old concept, dating as far back as 1994 (De Santis, A., et al, "How to share a function securely," STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of Computing, May 1994, pp. 522-533), it is only in the 2010s that it gained momentum in the cryptographic community. The National Institute of Standards and Technology (NIST) conducted a workshop on threshold cryptography in 2019, and published a "Roadmap Toward Criteria for Threshold Schemes for Cryptographic Primitives" in 2020 (Brandao, L., et al., NISTIR 8214A, 2000).

[0007] Until 2018, researchers mostly focused on public-key threshold cryptography. When it comes to symmetric threshold cryptography, core definitions and concepts were introduced in 2018 (Agarwal, S., et al., "DiSE: Distributed Symmetric-key Encryption," CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 1993-2010). The article also details a generic construction of threshold authenticated encryption based on any DPRF. The researchers described the scheme as an alternative to the costly usage of hardware security modules, or naïve usage of single-point-of-failure solutions. They present participants as a group of servers, presumably owned by the same corporate entity.

[0008] In existing symmetric threshold cryptographic schemes, a fundamental limitation and security risk is the requirement for a trusted third party to facilitate or mediate interactions between participants. If the trusted third party is compromised, hacked, or otherwise unavailable, the entire cryptographic process, including the secret, may be compromised.

[0009] Accordingly, there is a need for improved cryptographic systems and methods for threshold symmetric encryption in the absence of a trusted third party while preserving all security guarantees that overcomes at least some of the limitations of existing systems and methods.

Summary

[0010] Systems and methods for trustless distributed symmetric encryption are described. According to an embodiment, there is a system for symmetric encryption. The system includes a plurality of participant devices that operate without a trusted third party to eliminate potential security risks associated with the third party and allows for the holder of a piece of information to encrypt it with the help of a subset of k among n participants in such way that the information can only be recovered with the help of another subset of k among those same n participants, which may be the same or differ from the subset of participants involved in encryption.

[0011] According to another embodiment, there is a method for symmetric encryption without a trusted third party. The method includes a setup phase, involving all participants, and further includes encryption and decryption phases involving k among n participants.

[0012] The setup phase comprises: initiating a setup request through direct communication between participants or facilitated by a communication-facilitating subsystem such as a web portal or a dedicated server; creating and sending setup information to a plurality of participant devices directly or through the communication-facilitating subsystem; verifying, by each participant device, the setup information and creating whitelists of the identifiers of encryption requesters and the identifiers of decryption requesters; creating partial secrets by each participant device; communicating partial secret shares amongst the participant devices; adding the partial secret shares to obtain a secret share for each participant device, storing the secret share in a cryptographic storage device; calculating and publishing commitments to a public storage, by each participant device; gathering the commitments for validating future encryption and decryption operations; and validating the data generated by a test encryption using the identifiers in the whitelists and commitments stored in the public storage.

[0013] The encryption phase comprises: sending, by an information holder device, an encryption request including its own identifier and a commitment to participant devices, either directly or through a communication-facilitating subsystem; communicating, directly or through the communication-facilitating subsystem, the identifier and the commitment,

to the participant devices; verifying, by each participant device, that the identifier is whitelisted; calculating, by each participant device, an evaluation and a proof to send to the information holder device; verifying, by the information holder device, the evaluation and proof received from each participant device using the commitments stored in the public storage; combining the received evaluations, by the information handler device, to form a mask which will be used to conceal the information, or in the case of a large volume of information, to conceal a symmetric key that can be used to recover the information; and publishing the masked data, the identifier and the commitment to the public storage.

[0014] The decryption phase comprises: retrieving, by an information requesting device, an identifier, commitments to participants' secret shares and the commitment to the original message, either through direct communication or with the help of a communication-facilitating subsystem such as a web portal or a dedicated server; sending the identifiers and the commitment to the message to each participant device, either directly or with the help of the aforementioned communication-facilitating subsystem; verifying, by each participant device, that the information requesting device is whitelisted for decryption requests and that an information holding device is whitelisted for encryption requests; calculating, by each participant device, an evaluation that is a function of the received commitment and their individual secret share, along with a proof to send to the information requester device; sending, by each participant device, the aforementioned evaluation and proof; validating, by the information requester device, the received proofs; combining, by the information requester device, the evaluations to re-form a mask; decrypting, using the aforementioned mask, the ciphertext retrieved from the public storage; and verifying, using the message commitment, the integrity of the retrieved message.

[0015] Other aspects and features will become apparent, to those ordinarily skilled in the art, upon review of the following description of some exemplary embodiments.

Brief Description of the Drawings

[0016] The drawings included herewith are for illustrating various examples of articles, methods, and apparatuses of the present specification. In the drawings:

[0017] FIG. 1 is a diagram of a system for threshold symmetric encryption shown during a setup phase, according to an embodiment.

[0018] FIG. 2 is a diagram of the system for threshold symmetric encryption shown during an encryption phase, according to an embodiment.

[0019] FIG. 3 is a diagram of the system for threshold symmetric encryption shown during a decryption phase, according to an embodiment;

[0020] FIG. 4A is a flow diagram of a setup phase of a threshold symmetric encryption method, according to an embodiment;

[0021] FIG. 4B is flow diagram of an encryption phase of a threshold symmetric encryption method, according to an embodiment;

[0022] Fig. 4C is a flow diagram of a decryption phase of a threshold symmetric encryption method, according to an embodiment;

[0023] FIG. 5 is a diagram of an exemplary implementation of the setup phase shown in FIGS. 1 and 4A;

[0024] FIG. 6 is a diagram of an exemplary implementation of the encryption phase shown in FIGS. 2 and 4B; and

[0025] FIG. 7 is a diagram of an exemplary implementation of the decryption phase shown in FIGS. 3 and 4C.

Detailed Description

[0026] Various apparatuses or processes will be described below to provide an example of each claimed embodiment. No embodiment described below limits any claimed embodiment and any claimed embodiment may cover processes or apparatuses that differ from those described below. The claimed embodiments are not limited to apparatuses or processes having all of the features of any one apparatus or process described below or to features common to multiple or all of the apparatuses described below.

[0027] One or more systems described herein may be implemented in computer programs executing on programmable computers, each comprising at least one

processor, a data storage system (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. For example, and without limitation, the programmable computer may be a programmable logic unit, a mainframe computer, server, and personal computer, cloud-based program or system, laptop, personal data assistance, cellular telephone, smartphone, or tablet device.

[0028] Each program is preferably implemented in a high-level procedural or object-oriented programming and/or scripting language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language. Each such computer program is preferably stored on a storage media or a device readable by a general or special purpose programmable computer for configuring and operating the computer when the storage media or device is read by the computer to perform the procedures described herein.

[0029] A description of an embodiment with several components in communication with each other does not imply that all such components are required. On the contrary, a variety of optional components are described to illustrate the wide variety of possible embodiments of the present invention.

[0030] Further, although process steps, method steps, algorithms or the like may be described (in the disclosure and / or in the claims) in a sequential order, such processes, methods and algorithms may be configured to work in alternate orders. In other words, any sequence or order of steps that may be described does not necessarily indicate a requirement that the steps be performed in that order. The steps of processes described herein may be performed in any order that is practical. Further, some steps may be performed simultaneously.

[0031] When a single device or article is described herein, it will be readily apparent that more than one device / article (whether or not they cooperate) may be used in place of a single device / article. Similarly, where more than one device or article is described herein (whether or not they cooperate), it will be readily apparent that a single device / article may be used in place of the more than one device or article.

[0032] Reference herein to “participant(s)” means an individual/device/system that holds cryptographic information and may mathematically contribute to an encryption or decryption operation. An “entity” may request an operation of encryption or decryption. Participants may or may not be entities. It should be noted that certain values herein include the prefix “0x” indicating hexadecimal notation.

[0033] The cryptographic system to be described is a system that operates without a trusted third party to eliminate potential security risks associated with the third party and allows for the holder of a piece of information to encrypt it with the help of a subset of k among n participants in such way that the information can only be recovered with the help of another subset k among those same n participants. In particular, compared to traditional encryption protocols, the present system is set up without a trusted third party while preserving all security guarantees. Furthermore, a different setting, moving away from a group of servers and towards a collaboration between corporate entities and individual clients is described.

[0034] Herein, *security* is defined as the presence of three qualities: strong-correctness, message privacy and strong-authenticity, as defined by Agarwal, S., et al. The invention comprises an algorithm subdivided into three operational phases, referred to as the *setup phase*, *encryption phase* and *decryption phase*. Herein, the following terminology is used: *k-of-n* threshold scheme (with $0 < k \leq n$) to communicate the fact that k among n participants will be involved in the encryption and decryption phases. All participants are involved in the setup phase.

[0035] SETUP PHASE

[0036] Referring to FIG. 1, shown therein is a diagram of a system for threshold symmetric encryption 100 during a setup phase 100, according to an embodiment. The system 100 provides a way to jointly encrypt arbitrary information or documents in such way as to ensure its security despite a subset of participants acting maliciously. Generally, there may be any number, n , participants (i.e., $p_1, p_2, , p_4 \dots p_n$).

[0037] In the setup phase, an initiator 10 will identify the involved participants 12a, 12b, 12c, 12d and provide them with setup information 11 to contact each other. The setup information 11 will also identify the location of a shared public storage repository

13 to be used by all participants 12a, 12b, 12c, 12d and will provide the participants with a list of authorized entities allowed to initiate the encryption and decryption phases, called a whitelist, as well as public information required to start the cryptographic process. It should be noted that a secure channel is not required to communicate the setup information 11 to the participants 12a, 12b, 12c, 12d.

[0038] The setup information 11 includes: a total number of participants, n ; a required number of participants involved in an operation of encryption or decryption, k ; the public storage 13; a list of entities allowed to make encryption requests and a list of entities allowed to make decryption requests. The list of entities allowed to make a decryption request may differ from the list of entities allowed to make encryption requests.

[0039] The setup information 11 further includes: a cyclic group, G , of order q , where the Discrete Logarithm Problem (DLP) is hard; two generators chosen independently from one another, which shall be provided in conjunction with proof that the initiator 10 is not aware of any mathematical relationship between those two generators; an ordering for the participants; and a unique identifier for this group of participants.

[0040] After receiving the setup information 11, each participant 12a, 12b, 12c, 12d will individually do the following: (1) create two whitelists, matching the provided lists of entities allowed to make encryption and decryption requests, respectively; (2) create a secret value between 0 and $q-1$ (inclusive), the order of the group, G , referred to as a partial secret; (3) split the partial secret into n partial secret shares, k of which are required to reform the partial secret; and (4) distribute those n partial secret shares 22 to all n participants over an end-to-end secure channel.

[0041] Upon receiving all n partial secret shares 22, each participant 12a, 12b, 12c, 12d will combine those partial secret shares 22 to form a secret share. The secret share is represented as a cartesian coordinate (x, y) , where an x -coordinate may be public and corresponds to their ordering in group, G , and a y -coordinate is a participant's partial secret, s .

[0042] The participants 12a, 12b, 12c, 12d then publish a Pedersen commitment to their individual secret share 33a, 33b, 33c, 33d in the shared public storage 13 by

calculating the following: for $gen1$, $gen2$, the two generators provided by the initiator 10, s the y-coordinate of the *secret share*, and r a value chosen at random between 1 and $q-1$ (inclusive), calculate $gen1^s * gen2^r = \gamma$. Each participant 12a, 12b, 12c, 12d privately hold s and r as a private key.

[0043] Once all commitments 33a, 33b, 33c, 33d have been published, each participant 12a, 12b, 12c, 12d will verify that no malicious behavior has taken place during the setup phase (not shown). Each participant 12a, 12b, 12c, 12d will ensure the correctness of the generated shares 22 by contacting all other participants and requesting their help to encrypt a random value. For the purposes of this verification only, the whitelist is set to be the list of participants. A participant 12a, 12b, 12c, 12d will verify the consistency of all shares 22 by verifying that the encryption of the random value with the help of k participant is the same as the encryption of the random value with the help of n participants, but differs from the encryption of the random value with the help of $k-1$ participants. Details of the encryption process are described below.

[0044] ENCRYPTION PHASE

[0045] Referring to FIG. 2, shown therein is a diagram of an encryption phase 110, according to an embodiment. The encryption phase 110 can be initiated by any entity allowed to do so, as per the whitelist identified in the setup phase. The encryption phase allows one such entity to encrypt any piece of information with the help of k among n participants 12a, 12b, 12c, 12d in such way that at least k among n participants is required to decrypt the information. Note that the participant 12d represents a subset of participants that are not involved in the encryption, and indeed do not have to be online at all, since only k among n participants are required. Note that the participants 12a, 12b, 12c contributing to the encryption are not necessarily the same as the participants contributing to the decryption (FIG. 3).

[0046] The encryption phase 100 is initiated by an information holder (info holder) 14, which may or may not be a participant as identified in the setup phase. The info holder 14 will contact k participants 12a, 12b, 12c (or $k-1$, if it is itself a participant) and will provide them with a packet 44 including a commitment, α , which is a function of the information/message, m , they want to encrypt, and their own identifier.

[0047] The info holder 14 first calculates the commitment, α , of m under freshly generated randomness, ρ , as follows: $\alpha = \text{XOF}(m \parallel \rho)$, where XOF is an eXtensible Output Function, such as SHAKE256, and $\parallel$ is the concatenation symbol. The info holder 14 sends its identity id and α in the packet 44 to k participants 12a, 12b, 12c.

[0048] Each contacted participant 12a, 12b, 12c will first confirm that the info holder 14 is part of the provided whitelist and is allowed to request encryption before providing a response 55 to the info holder 14. The response 55 consists of two elements: (A) a value that will be a function of the data contained in the packet 44 and that participant's secret share, and (B) a proof that will be used to establish security.

[0049] Each contacted participant 12a, 12b, 12c, calculates the value (A) as:

- $\omega = H(id \parallel \alpha)$, where $H()$ is a cryptographically secure hash function;
- $h = \omega^s$ (using s as defined in the setup phase);
-

[0050] Each contacted participant 12a, 12b, 12c sets the proof (B) as:

- randomly sample $mask1$, $mask2$ in the range $[1, q-1]$;
- set $t = \omega^{mask1}$, $t' = gen1^{mask1} * gen2^{mask2}$;
- set $challenge = H'(h, \omega, \gamma, gen1, gen2, t, t')$, for $H'()$ a cryptographically secure hash function that differs from $H()$;
- set $sig1 = (mask1 - challenge * s)$, $sig2 = (mask2 - challenge * r)$ (using r as defined in the setup phase); and
- call $(challenge, sig1, sig2)$, the proof, as it proves that the participant used its secret share in its calculations.

[0051] Each participant 12a, 12b, 12c then returns the evaluation h , the proof $(challenge, sig1, sig2)$, and optionally ω , to the info holder 14 as the response 55 over an end-to-end secure channel.

[0052] After receiving the response 55 from all contacted participants, the info holder 14 will then validate the received proof, and will combine the received values to form a mask:

- calculate $\omega = H(id \parallel \alpha)$;
- for each response received:
 - compute $t = \omega^{sig1} * h^{challenge}$;
 - compute $t' = gen1^{sig1} * gen2^{sig2} * \gamma^{challenge}$; and
 - verify that $challenge = H'(h, \omega, \gamma, gen1, gen2, t, t')$.

[0053] If all verifications are not successful, the encryption phase is terminated. If all verifications are successful, the info holder 14 calculates the Lagrange interpolation, *comb*, of all received responses 55, using the participant's identifier as listed in the ordered list of participants and the value *h* provided in the participant's response 55. Next, the info holder 14 calculates $e = PRG(comb) \text{ xor } (m \parallel \rho)$, where PRG is a pseudo-random number generator seeded by *comb*.

[0054] Preferably, though not mandatory, the info holder 14 will encrypt its information using any symmetrical cipher that matches its target security level, and will hide the symmetrical cipher key with the mask. According to other embodiments, the information is encrypted using the mask (without symmetrical cipher), although this typically requires more storage space. The info holder 14 will then publish 66 the commitment α , the identity of the info holder 14, *id*, and a value, *e*, (that masks both the encrypted information/message, *m*, and the randomness used to calculate α), in the designated public storage 13.

[0055] DECRYPTION PHASE

[0056] Referring to FIG. 3, shown therein is a diagram of a decryption phase 120, according to an embodiment. The decryption phase 120 can be initiated by any entity allowed to do so, as per the list identified in the setup phase. In the decryption phase 120, an information requester (info requester) 15, which may or may not be a participant as identified in the setup phase, will contact *k* participants 12b, 12c, 12d (or *k-1*, if it is itself

a participant) and will provide them with a packet 77, which can be found in the designated public storage 13, matching the information they wish to retrieve, along with their identifier, id' . Note that the participant 12a represents a subset of participants that are not involved in the decryption, and indeed do not have to be online at all, since only k among n participants are required. Note that the participants 12b, 12c, 12d contributing to the decryption are not necessarily the same as the participants contributing to the encryption (see FIG. 2).

[0057] The info requester 15 will first retrieve the encrypted information 67, namely id , α and e , from the public storage 13. The info requester 15 will then send its own packet 77 including its own identifier id' and $id \parallel \alpha$ to k participants 12a, 12b, 12c. Note the distinction between id , the identifier of the entity that requested the encryption, and id' , the identifier of the entity requesting the decryption.

[0058] The contacted participants 12b, 12c, 12d will first confirm that the info requester 15 is part of the provided whitelist and allowed to request decryption, and will then provide a response 88 to the info requester 15 over an end-to-end secure channel. The response 88 consists of two elements: (A) an evaluation that will be a function of the commitment, α , to the information/message, m , and that participant's secret share, and (B) a *proof* that will be used to establish security. Each contacted participant 12b, 12c, 12d confirms both the id and id' are an allowed information requester and an allowed information holder, respectively, then calculates (A) and (B) in the same manner described above in the encryption phase.

[0059] The info requester 15 will then validate the received proof, and will combine the received values to form a mask:

- calculate $\omega = H(id \parallel \alpha)$;
- for each response 88 received:
 - compute $t = \omega^{sig1} * h^{challenge}$;
 - compute $t' = gen1^{sig1} * gen2^{sig2} * \gamma^{challenge}$; and
 - verify that $challenge = H'(h, \omega, \gamma, gen1, gen2, t, t')$

[0060] If all verifications are successful, calculate the Lagrange interpolation of all received responses 88, *comb*, using the participant's identifier as listed in the ordered list of participants and the value *h* provided in the participant's answer.

[0061] Next the info requester 15 calculate $m \parallel \rho = \text{PRG}(\text{comb}) \text{ xor } e$ and verifies that $\alpha = \text{XOF}(m \parallel \rho)$.

[0062] If this last check succeeds, the info requester 15 has successfully recovered the original information/message, *m*. As noted above, *m*, may or may not be a symmetrical cipher key itself, since use of a symmetrical cipher in the encryption phase is optional. In embodiments where a symmetrical cipher is used, the info requester 15 will use the mask to reveal the symmetrical cipher key used in the encryption phase, and will use that symmetrical cipher key to decrypt the desired information, thus completing the decryption phase. In embodiments where a symmetrical cipher is not used, *m* is, by itself, the desired information.

[0063] FURTHER CONSIDERATIONS

[0064] It is often inconvenient to manipulate large amounts of data directly. To accelerate calculations and limit the amount of required storage space, it is recommended to first encrypt a large information/message, *m*, using a cryptographically secure symmetric encryption scheme such as AES, then use the present scheme to encrypt only the symmetric encryption key itself rather than the entirety of the data (including the information/message and/or the encryption key).

[0065] Referring to FIGS. 4A-4C, shown therein are flow diagrams of a threshold symmetric encryption method including a setup phase 300 (FIG. 4A), an encryption phase 330 (FIG. 4B) and a decryption phase 350 (FIG. 4C). The phases 300, 330, 350 may be performed in sequence as part of an encryption method. Note that the setup phase 300 (FIG. 4A) must be completed only once, and the encryption phase 330 (FIG. 4B) and decryption phase 350 (FIG. 4C) can be completed multiple times, based on the same setup phase 300.

[0066] Referring to FIG. 4A, shown therein is a flow diagram of a setup phase 300 for a threshold symmetric encryption method. The setup phase 300 may be performed by the system 100 shown in FIG. 1.

[0067] At 302, a setup request is initiated from an end user device. The setup request may be sent directly to each participant device or be sent via a communication-facilitating subsystem. Preferably, the communication-facilitating subsystem is a web portal, but it may be a different communication-facilitating subsystem such as a dedicated server.

[0068] At 304, the setup information for each participant device as specified in the setup request is sent to each participant device. Generally, the setup information specifies details of encryption requesters, decryption requesters and the number and identity of participant devices.

[0069] At 306, each participant device verifies the setup information and generates its own whitelists for encryption and decryption. According to some embodiments, each participant device verifies the setup information locally. According to other embodiments, the setup information may be verified in a distributed fashion across more than one participant device.

[0070] At 308, each participant device creates a partial secret and partial secret shares according to Shamir's secret sharing scheme.

[0071] At 310, each participant device shares its partial secret shares with each other participant device over an end-to-end secure channel.

[0072] At 312, each participant device adds the partial secret shares it receives to obtain its secret share. The secret share is stored in a cryptographic storage device.

[0073] At 314, each participant device calculates and publishes secret share commitments to a public storage. Generally, the commitments will be used later for encryption and/or decryption.

[0074] At 316, the participants devices gather the published commitments from the public storage for validating future encryption and decryption operations.

[0075] At 318, it is preferred that the data generated during the setup phase is verified by a single test encryption (i.e., a test run of the data generated including the participant device identity, whitelists, secret shares, commitments, etc.) to ensure that no malicious behavior has taken place during the setup phase. According to some embodiments, this step 318 may be omitted.

[0076] Referring to FIG. 4B, shown therein is a flow diagram of an encryption phase 330 for a threshold symmetric encryption method. The encryption phase 330 may be performed by the system 110 shown in FIG. 2.

[0077] At 332, an information holder device initiates an encryption request including the identifier and the commitment of the information holder device. The encryption request may be sent directly to each participant device, or preferably, be sent via a communication-facilitating subsystem (e.g., a web portal or a dedicated server).

[0078] At 334, the identifier and the commitment of the information holder device are communicated to participant devices.

[0079] At 336, each participant device verifies that the information holder device's identifier is whitelisted for encryption requests.

[0080] At 338, each participant device calculates an evaluation and a proof to send to the information holder device over an end-to-end secure channel.

[0081] At 340, the information holder device verifies the evaluation and the proof received from each participant device using the secret share commitments stored in a public storage.

[0082] At 342, the information holder device combines the proofs and the evaluations received from each participant device to form a mask;

[0083] At 344, the information holder device encrypts information using the mask.

[0084] Referring to FIG. 4C, shown therein is a flow diagram of a decryption phase 350 for a threshold symmetric encryption method. The encryption phase 350 may be performed by the system 120 shown in FIG. 3.

[0085] At 352, an information requesting device sends a decryption request including its identifier and an indication of the information to be retrieved from a public storage. The decryption request may be sent directly to each participant device, or preferably be sent through a communication-facilitating subsystem (e.g., a web portal or a dedicated server).

[0086] At 354, the identifier and commitments of the information holder device are retrieved from the public storage along with the encrypted information. The retrieval may be performed by the information requesting device directly or by the communication-facilitating subsystem.

[0087] At 356, the identifier and commitments of the information requesting device along with the identifier of the information holding device are sent to the participant devices either directly or via the communication-facilitating subsystem.

[0088] At 358, each participant device verifies that the identifier of the information requesting device is whitelisted for decryption requests and that the identifier of the information holding device is whitelisted for encryption requests. According to some embodiments, each participant device locally verifies that the information holding device and the information requesting device are whitelisted for encryption and decryption, respectively. According to other embodiments, the verification may be performed in a distributed fashion across more than one device.

[0089] At 360, each participant device sends a verification including a second evaluation and a second proof to the information requesting device over an end-to-end secure channel.

[0090] At 362, the information requesting device verifies a second hash, the second evaluation and the second proof received from each participant device using the secret share commitments stored in the public storage.

[0091] At 364, the second evaluations and the second proofs received from each participant device are combined to generate a mask.

[0092] At 366, the information requesting device uses the mask to decrypt the encrypted information/message.

[0093] EXAMPLES

[0094] The exemplary implementations shown in FIGS. 5-7 of the setup phase (FIG. 5), the encryption phase (FIG. 6) and the decryption phase (FIG. 7) are provided as possible embodiments of the invention. All examples involve a system of 4 participants/devices/systems: an end user 202, a service provider 204, a system provider 206 and a backup 208.

[0095] The end user 202, the service provider 204, the system provider 206 and the backup 208 may be a server computer, desktop computer, notebook computer, tablet, PDA, smartphone, or another computing device. One or more of the devices 202, 204, 206, 208 may be virtual devices/machines. The devices 202, 204, 206, 208 may include a connection with a network such as a wired or wireless connection to the Internet. In some cases, the network may include other types of computer or telecommunication networks.

[0096] The devices 202, 204, 206, 208 may include one or more of: a memory, a secondary storage device, processor(s), an input device, a display device, and an output device. Memory may include random access memory (RAM) or similar types of memory. Also, memory may store one or more applications for execution by processor. The applications, computer readable instructions or programs may be stored in memory or in secondary storage or may be received from the Internet or other network.

[0097] Applications may correspond to software modules comprising computer executable instructions to perform processing for the functions described below. Secondary storage device may include a hard disk drive, floppy disk drive, CD drive, DVD drive, Blu-ray drive, or other types of non-volatile data storage. Processors may execute the applications, computer-readable instructions or programs.

[0098] The devices 202, 204, 206, 208 may include an input device for entering information into device 202, 204, 206, 208. For example, input device may be a keyboard, keypad, cursor-control device, touchscreen, camera, or microphone. The devices 202, 204, 206, 208 may include a display device for presenting visual information. For example, display device may be a computer monitor, a flat-screen display, a projector or a display panel.

[0099] Although the devices 202, 204, 206, 208 are described with various components, one skilled in the art will appreciate that the devices 202, 204, 206, 208 may in some cases contain fewer, additional or different components. In addition, although aspects of an implementation of the devices 202, 204, 206, 208 may be described as being stored in memory, one skilled in the art will appreciate that these aspects can also be stored on or read from other types of computer program products or computer-readable media, such as secondary storage devices, including hard disks, floppy disks, CDs, or DVDs; a carrier wave from the Internet or other network; or other forms of RAM or ROM. The computer-readable media may include instructions for controlling the devices 202, 204, 206, 208 and/or processor to perform a particular method.

[0100] Each device 202, 204, 206, 208 may be associated with a user/company account. Any suitable mechanism for associating a device with an account is expressly contemplated, including by the use of cryptographic storage devices 203, 205, 207, 209 as described below.

[0101] In the description of the exemplary implementations that follows, the devices/participants 202, 204, 206, 208 are described performing certain acts. It will be appreciated that any one or more of these devices may perform an act automatically or in response to an interaction by a user of that device. That is, the user of the device 202, 204, 206, 208 may manipulate one or more input devices (e.g., a touchscreen, a mouse, or a button) causing the device to perform the described act. In many cases, this aspect may not be described below, but it will be understood.

[0102] Each participant 202, 204, 206, 208 is associated to a cryptographic storage device, for example, a portable cryptographic module 203 (e.g., Multi-Pass™, Yubikey™, or the like, or hardware security modules 205, 207, 209. The cryptographic storage devices 203, 205, 207, 209 can only be accessed by the participant 202, 204, 206, 208 they are linked to.

[0103] A communication-facilitating subsystem (comm. system) 212, created in collaboration between various service providers and the system provider 206, is used to connect the participants 202, 204, 206, 208 when they are online. The comm. system 212 is preferably a web portal, but may be a different communication-facilitating subsystem

such as a dedicated server. Using the comm. system 212, end users 202 may request the encryption of various documents that have been created jointly with a service provider. According to other embodiments, the participants 202, 204, 206, 208 may communicate directly, without the comm. system 212 intermediary.

[0104] It is assumed the service provider 204, the system provider 206 and the backup provider 208 are always online and the end user 202 is online only when it initiates an operation. A cloud-based public storage 214 (e.g., AWS®) is always online.

[0105] Referring to FIG. 6, consider an individual who wishes to protect a digital copy of a notarized house purchase agreement and a company specialized in digital security, able and willing to provide data protection using the above-described invention. The individual, the notary, and the company are respectively the end user 202, the service provider 204 and the system provider 206. A separate entity called the backup 208, which is managed by the system provider 206 independently from its core systems, is also provided as a participant. The end user 202, the service provider 204, the system provider 206 and the backup 208 form a group of four participants.

[0106] The end user 202, acting as the initiator, creates a request 220 through the comm. system 212. The request 220 includes the total number of participants, $n=4$, and the required number of participants involved in an operation of encryption or decryption, $k=3$. In this example, the list of entities allowed to make encryption requests will only contain the end user 202; the list of entities allowed to make decryption requests will contain the end user 202, the service provider 204 and the system provider 206. Note that in this example, the list of allowed entities only contains entities that are also participants. This is not necessarily the case in other embodiments of the invention.

[0107] The comm. system 212 generates information 222 based on the request 220. The information 222 includes a cyclic group e.g., the group of points on the elliptic curve secp256k1, defined by the following values:

- the prime defining the field $p = 0xfffeffffc2f$;
- the curve equation is $y^2 = x^3 + 7$;

- the base point $G =$
(0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798
,
0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8);
- the order of the group $q =$
0xfffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141; and
- the cofactor of the group $h = 1$.

[0108] The first generator $gen1$ is set to be the base point G . The second generator $gen2$ can be calculated as follows:

- generate 32 random bytes, called $init_{rand}$;
- hash $init_{rand}$ to the above-described curve using an elliptical hashing method, for example, as described by Faz-Hernandez, et al., ("Hashing to Elliptical Curves," Internet Engineering Task Force: Crypto Forum Research Group accessible at <<https://www.ietf.org/archive/id/draft-irtf-cfrg-hash-to-curve-10.html>>) using the domain separation tag 'PEDERSEN_SECP256K1_COMMIT_PRNG-v1.13.0-hash_to_curve' and SHA256 as hashing algorithm; and
- in the unlikely case that the result $gen2$ is the same as $gen1$, try again with fresh randomness.

[0109] Both $init_{rand}$ and $gen2$ will be published so that every participant can verify that $gen2$ has been generated from $init_{rand}$, which constitutes a satisfying proof that the relationship between the two generators is not known.

[0110] In this example, the following values are set:

- Seed:
0xe5699fe804e189bc320b93ababcb2d39175d9f688ca53b66c9d07bfa32a772d;
and
- Generator:
(0x262ee3413bbb3beb89139998678f5d22e571e1548b4c445b313cd870987f3f8

3,
 0x39bae3a78b57176e15d244f77144cbfda8e047f2bad04bf3b0ec6f162b081f70)

[0111] In this example, the ordering of the participants is set to be the following:

- 1. end user 202;
- 2. service provider 204;
- 3. system provider 206; and
- 4. backup 208.

[0112] The identifier of each participant is a random Universally Unique Identifier, as follows:

- end user 202: e44838bd-a5c1-4f1a-8746-1c9e87b504c4;
- service provider 204: 106d8052-e68b-4763-b4fc-f233d224cedc;
- system provider 206: c538806e-4073-4b12-bc4a-bde52adb6c40; and
- backup 208: 3ffca6d1-8a14-4b6a-a723-47e76fa07fd9

[0113] The comm. system 212 creates the information 222 detailed above, and communicates them to all involved participants 204, 206, 208. The comm. system 212 also provides participants with a means to communicate with one another and with the public storage 214 which may be done through the intermediary of the comm. system 212 itself.

[0114] Locally, each participant 204, 206, 208 verifies the information 222 and creates a copy of the received whitelist.

[0115] The participants 204, 206, 208 then create partial secrets using Shamir's secret sharing scheme, and split the partial secret shares 224 amongst themselves as follows:

[0116] End user 202:

- degree-2 polynomial:
 0x17b2f0c7dd1bf1f77a7f2a3af1b4ac55a73eb436a82350a0f114e73c3239954 +

$0x9ad8dc4fbff7873500a6c477ce0dc749620dee3cda5d6b257d6091c0dec7b2a3$
 $\cdot x +$
 $0x45883af57eabdbccdd4c4c3b3f9f02d881ba7b258885fffe490fa07d1f9cb7d06 \cdot$
 $x^2;$

- partial secret:
 $0x17b2f0c7fdd1bf1f77a7f2a3af1b4ac55a73eb436a82350a0f114e73c3239954;$
- Partial secret shares 224:
 - for itself (end user 202): (1,
 $0xf814080d3c7522224d137acf77193f96d8298bd8cd3fa0141d6be8069bb$
 $6c8fd);$
 - for service provider 204: (2,
 $0x6385953d78703cc0cc088a6332f78f7b17d0d751e22bca6fce15d423c77$
 $47030);$
 - for system provider 206: (3,
 $0x5a079858b1c30efaf487215ee2b63a6f8ec7877c07d7f494a0b3cfe4e6c9$
 $116f);$ and
 - for backup 208: (4,
 $0xdb9a115ee86d98d0c68f3fc2865540743d0d9c573e441e829545db49f9b$
 $4acba).$

[0117] Service provider 204:

- degree-2 polynomial:
 $0xfd4b14b951844a6062e8e04b8be59bab669f4a20a0f7830e89cdb66cff9850a7$
 $+$
 $0x6382f59a1bb82d40ae8cd8f4ab973b1d854ce27828ee7f50b45bae69ef66eee2$
 $\cdot x +$
 $0xf082184567d7d88c015fc3ab73f5467945a863b8039a7e072d9c17ce2e07482c$
 $\cdot x^2;$

- partial secret:
0xfd4b14b951844a6062e8e04b8be59bab669f4a20a0f7830e89cdb66cff9850a7;
- partial secret shares 224:
 - for end user 202: (1,
0x51502298d514502d12d57cebab721d44bc36d6836eef3feec20bf8b7c9a0533);
 - for itself (service provider 204): (2,
0x8659610328540711c581a0e2b2e92bd1e2704d6f94d358a1e9d999b9857408d6);
 - for system provider 206: (3,
0x9c66cff84b436f0e7aed4c30a24ac7541e9cd1fe635b2cebc325e66a49f01a4f); and
 - for backup 208: (4,
0x93786f783de2882333187ed57996efcb70bc642fda86bcc7805a59dca0e399e).

[0118] System provider 206:

- degree-2 polynomial:

$$0x9fb7db5b29b356970eb4059fd9f36a4aaacb30b16585a22d0d1ccf5feb67c748 +$$

$$0xab146c8adadaeb0f4165b9a72a204a6b278d320f932a80b2d0a49173b07e23e$$

$$9 * x +$$

$$0x6e43678cc44b4555bd7ad9d97511d6e33e9825ce136645f013b1eb8238864de$$

$$5 * x^2;$$
- partial secret:
0x9fb7db5b29b356970eb4059fd9f36a4aaacb30b16585a22d0d1ccf5feb67c748;
- partial secret shares 224:
 - for end user 202: (1,
0x9fb7db5b29b356970eb4059fd9f36a4aaacb30b16585a22d0d1ccf5feb67c748)

- for service provider 204: (2,
0xae52a3f096420c876ae054027b5ab1c4399554cb99da9fbd684a9bd
da82eb)
- for itself (system provider 206): (3,
0x8153c4eea0e987c87c36db3a75f4d790f4b2edb6b1e9d84fb15d9402185
5688a); and
- for backup 208: (4,
0x30400652d9d3582feb889d3d3920237e7adb4ce0fbdc1a40c961bd213a
6a8b2).

[0119] Backup 208:

- degree-2 polynomial:
0xd51c9ffba7c8089f574934d3302b786ce9edc3f27e91d3d614efc78ef0902ba +
0xf7f205b0455e23601ae73a4b1cf469065f262b957be07cb40d207375f3e6b820 *
x +
0x911ce83fe862ecb5b222c40a484e5fccb52e8f687684aa832418d1c780044287
* x²;
- partial secret:
0xd51c9ffba7c8089f574934d3302b786ce9edc3f27e91d3d614efc78ef0902ba;
- partial secret shares 224:
 - for end user 202: (1,
0xd51c9ffba7c8089f574934d3302b786ce9edc3f27e91d3d614efc78ef090
2ba);
 - for service provider 204: (2,
0x9744c5fe6c47a20f3ce180c8e2508ccbc3b208a8d519f874cd751c2c5d8
36d1);
 - for system provider 206: (3,
0xd6f6db4fb6113d0d8963268b14a150d6753d82764bef9017cefcc277c93
177d1); and

- for itself (backup 208): (4,
0xc6b33abf5623d965833dbd1e2bba587c233f49659905953719af19a22c2
6bb5d).

[0120] It should be noted that the partial secret shares 224 listed above are not points on an elliptic curve, but points in two-dimensional cartesian space. All participants 202, 204, 206, 208 will discard the degree-2 polynomial and the partial secret and will distribute the partial secret shares 224 with one another. Note that while FIG. 5 shows direct communication between the participants 204, 206, 208, this may be done through the intermediary of the comm. system 212 (as is done between the end user 202 and the other participants 204, 206, 208). It should also be noted that the communications between the participants 204, 206, 208 are end-to-end secured, so that the comm. system 212 can direct the traffic but cannot see the contents of the communications.

[0121] The participants 202, 204, 206, 208 add their partial secret shares 224 (specifically, the y-coordinate of their share, which is the second value of the pair listed below) to obtain a secret share 226 using modular addition:

- end user 202 secret share: (1,
0xbe38a600f5b94970641190a82f817eaf986cb57f6b0c5ef5f853b651525515b0);
- service provider 204 secret share: (2,
0xa2419544781f00000cc323a676811eccc006fdb20a1fcfd02aae5bd0064f181);
- system provider 206 secret share: (3,
0x4eb9088f540142df750e6f550f972a2da1f70fda0a7b4970648f4faf71d38997);
- backup 208 secret share: (4,
0x6605c1e95647528968de0589ff388af6435944ed62fcf1b2b3ebf9426323c7e5);
and
- the shared secret, which remains unknown by all participants, is: (0,
0x89d280dc3385e0a0deb96bdc47f70844c51f88873a57370b87a6139ffcc0317b).

[0122] From the two generators provided by the comm. system 212, the participants 202, 204, 206, 208 can calculate the following commitments 228 (using fresh randomness r and the equation $\gamma = gen1 * s + gen2 * r$):

- End user 202:
 - r :
0x79f407f06c570b29005aff824d46b5a39bfe8220aa90be4466ddbb159c8705c0
 - γ :
(0x57f34d6d89556b51c797b6af94b1d5795e708742d73e63c4cde0429a06f3546c,
0x530b772755bcee3583c01bf82cb3674af0d1f5a2e303687cbb375df77f5c9f6e)
- Service provider 204:
 - r :
0xd619de2b4bb87cc36ae36153863a21d5c46890b1e45691e5de974476f114e279
 - γ :
(0xcb9f96da6e04c629d1a951815bfd43631c6ce97e77473b4da7aa0b63521cbb0d,
0xe03c738ad0c609f1eb96eebdea24db93c844e1cc23fc0d648f5d7737ae58261d)
- System provider 206:
 - r :
0xa437ae2c673f24ac7d3d9459acafa6290a5892c031730651ddc9ea17197b4453
 - γ :
(0x868edba0cc40b1e5e1d583dc81cb6f61ac5535436d7b464b708877191e1fbc3a,

0xebac13434a2d5c110d56d0f652dcaed5c15005f3113bd8c6321104d3acac2d9c)

- Backup 208:

- r :

0x87c745559349f1c301295412fce8742fda9d62debee347890f8fca06fe025637

- γ :

(0xe3920fda26117206c98e20b5f09fd4c7ad1ebe662e41229eb931011d11e32935,
0xcda4530999a3cc1bc8ea36a0627bcab82100070d9deee96f93dbfedb61131a4f).

[0123] The secret shares 226 are stored in the respective cryptographic storage devices 203, 205, 207, 209.

[0124] The participants 202, 204, 206, 208 calculate, then publish their commitments 228. Each participant 202, 204, 206, 208 will then keep their randomness r and secret share s secret, and publish γ . Note that while the drawing represents direct communication with the public storage 214, this may be done through the intermediary of the comm. system 212.

[0125] The participants 202, 204, 206, 208 gather the commitments 230 that will be used to validate future encryption and decryption operations.

[0126] The participants 202, 204, 206, 208 also go through a single encryption phase (not shown) using a random value to validate the data generated during the setup phase. After having obtained the response from other participants for the encryption of the random value, each participant will validate the following:

- The value obtained by combining the response of any 3 participants (possibly including itself) is the same as the value obtained by combining the response of all 4 participants;

- The value obtained by combining the response of any 2 participants (possibly including itself) differs from the value obtained by combining the response of any 3 participants; and
- The value obtained by combining the response of any 3 participants (possibly including itself) is not the point at infinity.

[0127] If any of the validations fail, the participant will claim that the setup phase failed, and it will be necessary to start over with fresh values. Otherwise, the setup phase concludes successfully, and an arbitrary number of encryption and decryption *phases* can begin for this group of participants.

[0128] Referring to FIG. 6, shown therein is an exemplary implementation of the encryption phase. Here, the end user 202 is the information holder. The end user 202 connects to the comm. system 212 to provide an encryption request 240 including their own identifier. The encryption request 240 may be linked with the identifier of the end user's portable cryptographic device 203 or may be linked with an identifier attributed to the end user 202 by the comm. system 212, but in either case, matches the identifier listed in the whitelist created during the setup phase.

[0129] To alleviate the calculation time and storage space, the information to be encrypted by the user will first be encrypted locally, using AES-256. The resulting ciphertext will be uploaded to the designated public storage 214, and the encryption phase will use only the 32-bytes AES key as message, m . For example, the AES key is the following:
0xece39589748cb37d62c68fe58ed5e39e8c5cca51093729adbff138804351c298.

[0130] The end user will then calculate a commitment, α , to the message, m , as follows:

- fresh randomness p =
0x6cf931d678a0c886462b288814e36099ccdb5ca5681704a821f9da150719e664
- α =
SHAKE256(0xece39589748cb37d62c68fe58ed5e39e8c5cca51093729adbff1388

04351c2986cf931d678a0c886462b288814e36099ccdb5ca5681704a821f9da150
 719e664) =
 0x53058f73c1d4607c5c53064f1e7bc5c88fff1c62e0fa462aed86f0fa3470a6cabdc
 5bb7a1efef5989164d242cf6004c60d70d30235a8f2d51a93a0ab5449cf795b1636
 4b3ec9cd9ba2a20a38bee2760550a8c76b228e64076772413c96a23d6e

[0131] Note that α is three times the length of the message to respect security parameters.

[0132] In this example, the comm. system 212 will then send this information 242, along with its identifier, to 3 among 4 participants: the service provider 204, the system provider 206 and the end user 202 (here, the backup 208 is not involved). Note that the end user 202, while being the initiator of the encryption request 240, is handled like any other participant.

[0133] The participants 202, 204, 206 verify that the identifier is whitelisted then calculate an evaluation, h , and a proof 244 as follows:

[0134] Service provider 204:

- hash the concatenation of the id and α (right below) to the above-described curve using the method described by Faz-Hernandez, et al., using the domain separation tag 'DISE_SECP256K1_DST-v1-hash_to_curve' and SHA256 as hashing algorithm;
- $\omega =$
 $H(0xe44838bda5c14f1a87461c9e87b504c453058f73c1d4607c5c53064f1e7bc5c$
 $88fff1c62e0fa462aed86f0fa3470a6cabdc5bb7a1efef5989164d242cf6004c60d70$
 $d30235a8f2d51a93a0ab5449cf795b16364b3ec9cd9ba2a20a38bee2760550a8c7$
 $6b228e64076772413c96a23d6e) =$
 $(0xbe6da964023e176f14730782e86a45bc1236fc28a440ef6fdfbbe919bc2398e8,$
 $0x981713afc6894875a136f844b372e357b34fd1cf1394d30b0861b7645bd81d6a)$
 ;
- calculate $h = \omega * s =$
 $(0xed4f1ae19e7d0791ed78dd2b8721aab379509bc1e2ef7533939507d0740f2eb$

- 6,
 0x51f2454480894506e1d3bdf349cdc78aa3a7522e6b4deb8acf0498c75d7c0b53)
 ;
- randomly sample $mask1$, $mask2$ in the range $(1, q-1)$:
 - $mask1 =$
 0xf06ac15c2438270044c0d920ca4838b12abb9b80b7a18d15ffdd6efae80
 b861b;
 - $mask2 =$
 0xf82106d5fb3c801a2ac3c64864dd7e9ac27b6bcc16edb18725b6d8b754d
 6fa47;
 - set $t = \omega * mask1 =$
 (0xf7a44bebefc5e4e203d965ae66dc735202552a09e48d19259419ad3a005d7a7
 3,
 0x61d317c4b48015f85786441c192ad32b80dbd510d1d2337236201c61a77b808)
 ;
 - set $t' = gen1 * mask1 + gen2 * mask2 =$
 (0x8e69f5ed58273f82ad7ee38df698d46b9e1b5c0b3b4b178875aacb609b3b758
 0, 0xe25e7eaf2adfd6b249942d519e10ef60399bf5fcd9e3efdb8cbe4d9d5e2fc82);
 - set $challenge = H'(h, \omega, \gamma, gen1, gen2, t, t')$ to the above-described curve using
 the method described by Faz-Hernandez, et al., using the domain separation tag
 'PUBLICLY_VERIFIABLE_NIZK-v2-hash_to_field' and SHA256 as hashing
 algorithm;
 - $challenge =$
 0xba66345e352db119f7734ec4eb8811760d6b10133dd85fbd0912c908df3
 85c81;
 - set $sig1 = (mask1 - challenge * s) \% q =$
 0xb9a327f10eecbb4e0c531cc8f30d478c70e26d94b6a9a532af3c5b0d7263f2f5;

- set $sig2 = (mask2 - challenge * r) \% q$
0x3293241eb1bebbf3339520024eb17b270ddd5b0377b547f4cb96d306aa7db60
b; and
- call (challenge, sig1, sig2) the proof.

[0135] System provider 206:

- hash the concatenation of the id and α (right below) to the above-described curve using the method described by Faz-Hernandez, A., et al., using the domain separation tag 'DISE_SECP256K1_DST-v1-hash_to_curve' and SHA256 as hashing algorithm;
- $\omega =$
H(0xe44838bda5c14f1a87461c9e87b504c453058f73c1d4607c5c53064f1e7bc5c
88fff1c62e0fa462aed86f0fa3470a6cabdc5bb7a1efef5989164d242cf6004c60d70
d30235a8f2d51a93a0ab5449cf795b16364b3ec9cd9ba2a20a38bee2760550a8c7
6b228e64076772413c96a23d6e) =
(0xbe6da964023e176f14730782e86a45bc1236fc28a440ef6dfdbbe919bc2398e8,
0x981713afc6894875a136f844b372e357b34fd1cf1394d30b0861b7645bd81d6a)
;
- calculate $h = \omega * s =$
(0xffab47e3a59f3d83e0071e43faa97b70ffe12ef2152208f3bb533b0a5cfc49fc,
0x2d74d2848f0698cf629f95921586596825d1c3f853c6411d77a2052eb527b215);
- randomly sample $mask1$, $mask2$ in the range (1, q-1):
 - $mask1 =$
0xbbb716500b726b84e00cc624cfad876628754b6f931dd3e0278333e0def
c4d0f;
 - $mask2 =$
0x5275e357521b4587b1a102e9563bd398f4027663f34a781d75bfc6dff0dd
2bf8;

- set $t = \omega * mask1 =$
 $(0xbb3ee393435b695b314fd8adc2fc0b27d744622149cfa9e1763c9cf5c49d2f81,$
 $0x722c6f46ee563e6b7126a6ea4346d1d4e88be6b63c74f108a860daaf1f723b25)$
 ;
- set $t' = gen1 * mask1 + gen2 * mask2 =$
 $(0xa65cea98a731ae016227388b747cc97dde95c7f1644cdf3790edf6aae02b3f51,$
 $0x40b3f9ad7b1292148e60552ab6dbe0aab5d19dc9448417386a967d07df73999$
 $d);$
- set $challenge = H'(h, \omega, \gamma, gen1, gen2, t, t')$ to the above-described curve using the method described by Faz-Hernandez, et al., using the domain separation tag 'PUBLICLY_VERIFIABLE_NIZK-v2-hash_to_field' and SHA256 as hashing algorithm;
 - $challenge =$
 $0xa5b0e8448b0488b6fab6512ec098fc9dc8814cdeb94402d22e758e5051$
 $127463;$
- set $sig1 = (mask1 - challenge * s) \% q =$
 $0xe852faac7504992e83aade90c2a306d8d5c815f6f628df778462ca257734343a;$
- set $sig2 = (mask2 - challenge * r) \% q$
 $0xc20e6ced665013d09c99a6003c327b454047ac49b5d518fc10cd5a37da9510e6$
 ; and
- call (challenge, sig1, sig2) the proof.

[0136] Since the end user 202 also needs its own share, it verifies that it is whitelisted and does the following locally:

- hash the concatenation of the id and α (right below) to the above-described curve using the method described by Faz-Hernandez, et al., using the domain separation tag 'DISE_SECP256K1_DST-v1-hash_to_curve' and SHA256 as hashing algorithm;

- $\omega =$
 $H(0xe44838bda5c14f1a87461c9e87b504c453058f73c1d4607c5c53064f1e7bc5c$
 $88fff1c62e0fa462aed86f0fa3470a6cabdc5bb7a1efef5989164d242cf6004c60d70$
 $d30235a8f2d51a93a0ab5449cf795b16364b3ec9cd9ba2a20a38bee2760550a8c7$
 $6b228e64076772413c96a23d6e) =$
 $(0xbe6da964023e176f14730782e86a45bc1236fc28a440ef6fdffbbe919bc2398e8,$
 $0x981713afc6894875a136f844b372e357b34fd1cf1394d30b0861b7645bd81d6a;$
- calculate $h = \omega * s =$
 $(0xac992d539e7c8708b4503a44f56a4f51e6bff0734f2261513c1cc64eab96f818,$
 $0xf419b3412f799ff5e03c098492248777edffe0b3ab62539605a0b98774297e02);$
- randomly sample $mask1, mask2$ in the range $(1, q-1)$:
 - $mask1 =$
 $0xc5ae57854208a16e69628496da1dc5d53cdc1a64509f3d73bfce1655ee$
 $b1ca0d;$
 - $mask2 =$
 $0xbc7f7590ee430cb0cc578321817609cdb649d4ff3beb7f612f7574c127a5$
 $838c;$
- set $t = \omega * mask1 =$
 $(0xa3ff332261d0bb5540c0b7d5020fb8570dfbe82d9e285fe312c5541aca3cf700,$
 $0x851bd9c57639631637781b44c10b4300f14b83403bb2b7675b20b183719358fd$
 $);$
- set $t' = gen1 * mask1 + gen2 * mask2 =$
 $(0xa09f62292090b490f2796ea5739203541991334f012cf67d2fa783c12d43be71,$
 $0x9e6c855ff4fe0ac7b205ee4d7a600e3a2a907e6d4394b37004885c9c9a804c83)$
 $;$
- set $challenge = H'(h, \omega, \gamma, gen1, gen2, t, t')$ to the above-described curve using the method described by Faz-Hernandez, et al., using the domain separation tag 'PUBLICLY_VERIFIABLE_NIZK-v2-hash_to_field' and SHA256 as hashing algorithm;

- $challenge =$
 $0xe1d9edcd245147a39aefefaec987fa59d1a8c90e7580fde934573bdf7cef$
 $c72;$
- set $sig1 = (mask1 - challenge * s) \% q =$
 $0x73adc9e688291c744a47145600ee2676c8affedbc3d4dde279f7fa5d02fe14a8;$
- set $sig2 = (mask2 - challenge * r) \% q$
 $0x81c3428e7dd79a216ff1062bba4431efd5b96ca9587f20ad1b826a7890a5ea9;$
and
- call (challenge, sig1, sig2) the proof.

[0137] The participants 202, 204, 206 send the resulting data 246 including their respective h and $proof$ to the information holder/end user 202 over an end-to-end secured channel. Note that while FIG. 6 shows direct communication, this may be done through the intermediary of the comm. system 212.

[0138] Once the end user 202 has all three shares (including its own), it does the following, for each response:

- calculate ω locally as a basis);
- compute $t = \omega * sig1 + h * challenge;$
- compute $t' = gen1 * sig1 + gen2 * sig2 + \gamma * challenge;$ and
- verify that $challenge = H'(h, \omega, \gamma, gen1, gen2, t, t').$

[0139] If all verifications are successful, the end user will use Lagrange Interpolation on the following values obtained by taking the participants' ordering and the h value they provided:

[0140] End user 202:

- (1,
 $(0xac992d539e7c8708b4503a44f56a4f51e6bff0734f2261513c1cc64eab96f818,$
 $0xf419b3412f799ff5e03c098492248777edffe0b3ab62539605a0b98774297e02))$

[0141] Service provider 204:

- (2,
(0xed4f1ae19e7d0791ed78dd2b8721aab379509bc1e2ef7533939507d0740f2eb
6,
0x51f2454480894506e1d3bdf349cdc78aa3a7522e6b4deb8acf0498c75d7c0b53)
)

[0142] System provider 206:

- (3, (0xffab47e3a59f3d83e0071e43faa97b70ffe12ef2152208f3bb533b0a5cfc49fc,
0x2d74d2848f0698cf629f95921586596825d1c3f853c6411d77a2052eb527b215)
)

[0143] The Lagrange interpolation of those values results in the value *comb* = (0xe4c8737c190f22b5a5a4771ee486bb4676146aa6f65f19f516311d9c6f7d8caa,
0x872a5bbddf2420e8d1badc9f056ebf1ad8bc8d726490ae5aaebcd7c80ef3b79e). The end user 202 then uses *comb* as seed for a pseudo-random number generator:

- $serialized_{comb} = comb.x || comb.y =$
0x04e4c8737c190f22b5a5a4771ee486bb4676146aa6f65f19f516311d9c6f7d8caa
a872a5bbddf2420e8d1badc9f056ebf1ad8bc8d726490ae5aaebcd7c80ef3b79e;
- calculate $csprng_{seed} = SHA256(serialized_{comb})$
0x886041f507cd49e99429c7d888ed2ac1d0983abeebcf2bff012fd1eaca60c906;
and
- return AES(0x00 * 64) with $csprng_{seed}$ as the key, using CTR mode and a random nonce:
 - $nonce = 0xc22912f781a7b1d1$
 - $PRG(comb)$ =
0x421a6408e86f9418d0faaa668a11a8e64e0ec27dd6ad75cb9df7a552800
528d43e85e2ac5369bf155357f310396850a5cce84c1ac13f0e8c371e2f13
5bdbb202

[0144] Finally, the end user 200 calculates $e = PRG(comb) \text{ xor } (m || \rho) =$
0xae9f1819ce32765b23c258304c44b78c252082cdf9a5c6622069dd2c354ea4c527cd3
7a2bc97793157cdb982d8b303c003310bfa9280a2416e7f5065cc25466.

[0145] The end user's ID, α , e , *nonce* and the original ciphertext 248 will be published in the designated public storage 214.

[0146] Referring to FIG. 7, shown therein is an exemplary implementation of the decryption phase. Here, the end user 202 is the information requester. The end user 202 connects to the comm. system 212 to send a decryption request 260 including their own identifier and indicates the information they wish to retrieve.

[0147] The comm. system 212 fetches the relevant information from the public storage 214, namely, the end user's ID, α , e , *nonce* and the original ciphertext 262. Here, having the end user 202 retrieve its own ID may seem redundant, but in other examples the ID is not necessarily readily available.

[0148] The comm. system 212 communicates this information 264 with 3 among 4 participants: the service provider 204, the system provider 206 and the end user 202 (here, the backup 208 is not involved). Note that the end user 202, while being the initiator of the decryption request 260, is handled like any other participant.

[0149] The following information 264 is sent to the other participants involved in the decryption:

- id' = e44838bd-a5c1-4f1a-8746-1c9e87b504c4, the identifier of the entity requesting the decryption;
- id = e44838bd-a5c1-4f1a-8746-1c9e87b504c4, the identifier of the entity having done the encryption; and
- α =
0x53058f73c1d4607c5c53064f1e7bc5c88fff1c62e0fa462aed86f0fa3470a6cabdc
5bb7a1efef5989164d242cf6004c60d70d30235a8f2d51a93a0ab5449cf795b1636
4b3ec9cd9ba2a20a38bee2760550a8c76b228e64076772413c96a23d6e, as per
retrieved from the public storage.

[0150] As in the encryption phase (FIG. 6), the end user 202 will choose itself as the third participant involved in the decryption and will therefore perform some computations locally.

[0151] The participants 202, 204, 206 verify that the identifier (id') of the information requester is whitelisted for decryption and the identifier (id) of the information holder is

whitelisted for encryption; then calculate a hash and a proof 266 as described for the *encryption phase*, above, using freshly generated random values for *mask1*, *mask2*.

[0152] The participants 202, 204, 206 send the resulting data including ω , h and *proof* 268 to the information requester/end user 202 over an end-to-end secured channel. Note that while FIG. 7 shows direct communication, this may be done through the intermediary of the comm. system 212.

[0153] The end user 202 verifies the data they received using the commitments that were stored in the public storage during the setup phase, then use the received data to recover the initial information. The end user 202 does the same verification it did during the encryption phase, namely by computing t , t' , and verifying that *challenge* matches the hash of h , ω , γ , *gen1*, *gen2*, t and t' .

[0154] If all verifications are successful, the end user 202 will calculate the Lagrange Interpolation of all received responses as it did in the encryption phase, to obtain *comb*. The end user 202 will then calculate $m \parallel \rho = PRG(comb) \text{ xor } e$, verify that $\alpha = XOF(m \parallel \rho)$ using the same XOF as in the encryption phase, and if this last verification is successful, the end user 202 can recover the message, which is the AES key required to recover the plaintext.

Claims:

1. A computer-implemented method for symmetric encryption without a trusted third party, the method comprising:

initiating, by an end user device, a setup request;

creating and sending setup information based on the setup request, to a plurality of participant devices, wherein the setup information includes identifiers of encryption requesters and decryption requesters;

verifying, by each participant device, the setup information and creating whitelists of the identifiers of encryption requesters and the identifiers of decryption requesters;

creating partial secrets by each participant device;

communicating partial secret shares amongst the participant devices;

adding the partial secret shares to obtain a secret share, by each participant device, for storing the secret share in a cryptographic storage device;

calculating and publishing secret share commitments to a public storage, by each participant device; and

gathering the secret share commitments from the public storage, by each participant device, for validating future encryption and decryption operations.

2. The computer-implemented method of claim 1, further comprising:

validating data generated in a test encryption, by each participant device, using the identifiers in the whitelists and the secret share commitments.

3. The computer-implemented method of claim 1, further comprising:

splitting a partial secret into n partial secret shares, by each participant device, where n equals a total number of participant devices.

4. The method of claim 1, wherein communicating the partial secret shares amongst the participant devices comprises:

each participant device retaining one partial secret share and sending a partial secret share to every other participant device over an end-to-end secure channel.

5. The computer-implemented method of claim 1, further comprising:

initiating, by an information holder device, an encryption request including an identifier and a commitment;

communicating, the identifier and the commitment, to at least a first subset of the participant devices;

verifying, by each participant device in the first subset, that the identifier is whitelisted for encryption;

calculating, by each participant device in the first subset, an evaluation and a proof to send to the information holder device over an end-to-end secure channel;

verifying, by the information holder device, the evaluation and the proof received from each participant device in the first subset using the secret share commitments stored in the public storage;

combining the evaluations and the proofs, by the information holder device, to form a mask;

encrypting information, by the information holder device, using the mask; and

publishing, by the information holder device, the identifier, the commitment and encrypted information to the public storage.

6. The computer-implemented method of claim 4, wherein encrypting the information, by the information holder device, using the mask comprises:

encrypting the information using the mask directly.

7. The computer-implemented method of claim 4, wherein encrypting the information, by the information holder device, using the mask comprises:

encrypting the information using a symmetrical cipher key; and

encrypting the symmetrical cipher key using the mask.

8. The computer-implemented method of claim 4, further comprising:

initiating, by an information requesting device, a decryption request including an identifier of the information requesting device and an indication of the encrypted information;

retrieving, by the information requesting device, the encrypted information, the identifier, the commitments of the information holder device from the public storage;

sending, by the information requesting device, the identifier and the commitment of the information requesting device and the identifier of the information holder device to at least a second subset of the participant devices;

verifying, by each participant device in the second subset, that the identifier of the information requesting device is whitelisted for decryption requests and that the identifier of the information holder device is whitelisted for encryption requests;

calculating, by each participant device in the second subset, a second evaluation and a second proof to send to the information requesting device over an end-to-end secure channel;

validating, by the information requesting device, the second proof and the second evaluation received from each participant device in the second subset using the secret share commitments stored in the public storage;

combining, by the information requesting device, the second evaluations and the second proofs to generate the mask; and

decrypting, by the information requesting device, the encrypted information using the mask.

9. The computer-implemented method of claim 7, wherein decrypting the encrypted information using the mask comprises:

decrypting the information using the mask directly.

10. The computer-implemented method of claim 7, wherein decrypting the encrypted information using the mask comprises:

decrypting a symmetrical cypher key using the mask; and

decrypting the encrypted information using the symmetrical cypher key.

11. A system for symmetric encryption without a trusted third party, the system comprising:

a plurality of participant devices, each participant device configured to:

verify setup information to create whitelists of identifiers of encryption requesters and identifiers of decryption requesters;

create and communicate partial secret shares to all other participant devices;

add the partial secret shares to obtain a secret share for the participant device;

publish a secret share commitment to a public storage;

verify that an identifier in an encryption request is whitelisted for encryption requests;

send an evaluation and a proof to an information holder device;

verify that an identifier in a decryption request whitelisted for decryption requests and that the information holder device is whitelisted for encryption requests; and

send a verification to an information requesting device;

the information holder device, configured to:

send the encryption request; and

verify the evaluation and the proof received from each participant device using the commitments stored in the public storage to encrypt information;

the information requesting device, configured to:

send the decryption request; and

decrypt the information using the commitments stored in the public storage.

12. The system of claim 11, further comprising:

a communication-facilitating subsystem configured to:

communicate the secret shares between the participant devices;

communicate the commitments between the participant devices, the public storage and the information holder device;

communicate the setup request and setup information from the user device to the participant devices;

communicate the encryption request from the information holder device to the participant devices; and

communicate the decryption request from the information requesting device to the participant devices.

13. The system of claim 12, wherein the communication-facilitating subsystem is a web portal or a dedicated server.

14. The system of claim 11, further comprising:

a user device configured to:

initiate a setup request to send the setup information for encryption and decryption of the original message to the plurality of participant devices, the setup information including the identifiers of the encryption requesters and the decryption requesters.

15. The system of claim 14, wherein the user device is the information holder device.
16. The system of claim 14, wherein the user device is the information requesting device.
17. The system of claim 11, wherein the setup information includes:
 - a total number of participant devices, n ;
 - a required number of participant devices for encryption or decryption operations, k ;
 - a list of entities authorized to initiate encryption; and
 - a list of entities authorized to initiate decryption.
18. The system of claim 11, wherein each secret share is represented as a cartesian coordinate.
19. The system of claim 18, wherein the cartesian coordinate comprises:
 - an x-coordinate that is a public value corresponding to a given participant device's ordering in a cyclic group, G , having an order, q ; and
 - a y-coordinate that is the partial secret share of the given participant device.