

THRESHOLD KEY EXCHANGE

TECHNICAL FIELD

- 5 The present disclosure relates to a method of generating shared cryptographic keys.

BACKGROUND

Public-key cryptography is a type of cryptographic system that uses pairs of keys: private
10 keys which are known only to the owner of the private key, and public keys which are generated based on the corresponding private key and which may be disseminated without compromising the security of the private key.

Public-key cryptography enables a sender to encrypt a message using a recipient's public
15 key (i.e. the public key corresponding to a private key known only to the recipient). The encrypted message can then only be decrypted using the recipient's private key. This type of encryption is known as asymmetric encryption.

An alternative to asymmetric encryption is symmetric encryption. Here, only one key (a
20 secret key) is used to both encrypt and decrypt a message (any data). The entities communicating via symmetric encryption must exchange the key so that it can be used in the decryption process.

Returning to public-key cryptography, a sender can use their own private key to sign a
25 message, e.g. to prove that the message is being sent by the sender, and/or to indicate that the sender agrees with the message. The signer (i.e. the party generating the signature) uses their private key to create a digital signature on the message. Anyone with the signer's corresponding public key can use the same message and the digital signature on the message to verify whether the signature was validly created, i.e. whether the signature was
30 indeed made using the signer's private key.

A digital signature scheme typically involves three procedures, i.e. algorithms. A key generation algorithm is used to generate a random private key and a corresponding public key. A signing algorithm is used to generate a signature based on a message and the private key. A verification algorithm is used to verify, given a public key and the message, whether the signature has been generated using the corresponding private key and according to the signing algorithm.

Threshold cryptography refers to a cryptographic system that distributes shares (sometimes called slices) of a private key between a plurality of participants. A threshold (i.e. minimum) number of those shares are required in order to re-create the private key. Depending on the particular system that is used, the private key may first be generated and then split into shares, or the shares of the private key may be generated without the private key ever existing. A message may be encrypted with the corresponding public key. A threshold number of participants must cooperate in order to decrypt the message.

A threshold signature scheme allows a threshold number of participants in a group to create a digital signature on (i.e. of) a message using individual shares of a shared private key. Here, a digital signature is a signature which is generated based on the message to be signed. In such a scheme, the signature can only be created if the threshold number of participants agree to generate the signature on the message. Any attempt to generate a signature using a smaller number of participants will not generate a valid signature. Therefore, a valid signature by the group (i.e. one generated using the message and the shared private key) provably had the threshold number of people agree to generate the signature. This also implies that any adversary needs to obtain the threshold number of shares of the private key to forge a signature with that private key. A common feature of threshold signature shares is that if any of the private key shares are lost, the private key can still be recoverable provided that the threshold number of shares are still available.

Diffie-Hellman key exchange refers to a protocol for generating a shared cryptographic key. The protocol can be summarised as follows. A first party and a second party each have a respective private-public key pair. The public keys are known to each party. The first party can calculate the shared key based on the elliptic curve multiplication of the first party's

private key and the second party's public key. The second party can calculate the same shared key based on the elliptic curve multiplication of the second party's private key and the first party's public key. This allows both parties to calculate the same shared key without divulging private information. Only the two parties can calculate the shared key because
5 only they know their respective private keys.

SUMMARY

A problem with the existing Diffie-Hellman (DH) key exchange method is that it introduces a
10 single point of failure. That is, if the private key of one of the parties is compromised by an attacker (or even accidentally disclosed) then the shared key can be obtained (assuming the other party's public key is known, which is often the case). Therefore any message that has been encrypted with the shared key can be decrypted by the attacker. This is particularly problematic if the message contains sensitive data such as, for example, personal identifying
15 information, financial data, medical data, etc.

It would therefore be desirable to increase the security of the DH key exchange method, or other similar methods that involve the generation of keys based on a secret (e.g. a private key).

20 According to one aspect disclosed herein, there is a computer-implemented method of generating a shared cryptographic key based on at least one shared secret, wherein each participant belonging to a first group has a respective share of a first secret, the first secret having a first threshold and a corresponding first public key, wherein a second coordinator
25 has a second public key corresponding to a second secret, and wherein the method is performed by a first coordinator of the first group and comprises: obtaining, from at least the first threshold number of participants of the first group, respective shares of the shared cryptographic key, where each respective share of the shared cryptographic key is based on
30 i) a respective share of the first secret or a respective zeroth order coefficient of a respective private polynomial used to calculate the respective share of the first secret, and ii) the second public key; and generating the shared cryptographic key based on the obtained

respective shares of the cryptographic key, wherein the second coordinator is configured to generate the same shared cryptographic key.

Embodiments of the present invention increase the security of shared keys (shared in the sense that the shared key is known to two different parties). Instead of generating a shared key based on a complete secret (e.g. a private key), the shared key is instead generated based on a shared secret (e.g. a shared private key). The shared secret is shared in the sense that each participant of a group has a share of a shared secret (e.g. a share of a shared private key). This shared secret is not known to two different parties. Moreover, the shared secret may not exist as a whole secret (e.g. key) in the sense that no individual knows that key. The shared secret will be referred to below as a shared Diffie-Hellman (DH) key to more easily distinguish it from the group's shared secret. However it should be appreciated that this is merely used as convenient label. The shared secret may also be referred to as a common secret, i.e. a secret common to (known to) more than one party. The private key shares may have been generated such that the private key never existed. The skilled person will be familiar with such techniques. Therefore no single party has access to the shared secret. The shared secret has a threshold, meaning that at least a threshold number of shares of the shared secret are required to reconstruct the shared secret.

The present invention generates the shared DH key based on one party's (or entity's) public key and at least a threshold number of shares of the shared secret. The complete shared DH key can only be constructed if enough different shares of the shared DH key are made available. Now, given that the shared secret does not exist (i.e. it is not stored by any one participant), it cannot be compromised, making it more difficult for an attacker to obtain the shared DH key. Moreover, in the event that an attacker compromises a participant's share of the shared secret, that share does not give enough information away in order to reconstruct the shared secret nor the shared DH key. An attacker would have to compromise the threshold number of shares, which is much more difficult task than compromising a single private key.

30

To further improve security of the shared DH key, both parties may use shared secrets. That is, one group of participants may have shares of a first shared secret and another group of

participants may have shares of a second shared secret. Each group generates respective shares of the shared DH key using the other group's public key and shares of their own group's shared secret.

- 5 In some embodiments, each and every participant of a group may be required to calculate a share of the shared DH key in order for that group to construct the shared DH key. This is particularly useful for use cases whereby collaboration of all participants is required.

BRIEF DESCRIPTION OF THE DRAWINGS

10

To assist understanding of embodiments of the present disclosure and to show how such embodiments may be put into effect, reference is made, by way of example only, to the accompanying drawings in which:

- 15 Figure 1 schematically illustrates an example system for generating shared keys according to some embodiments of the present invention,

Figure 2 schematically illustrates another example system for generating shared keys according to some embodiments of the present invention,

20

Figure 3 shows an example method for generating shared keys according to some embodiments of the present invention, and

Figure 4 schematically illustrates an example blockchain transaction protocol.

25

DETAILED DESCRIPTION OF EMBODIMENTS

CRYPTOGRAPHIC PRELIMINARIES

- 30 Elliptic curve groups

An elliptic curve E satisfies the equation:

$$y^2 = x^3 + ax + b \bmod p$$

where $a, b \in \mathbb{Z}_p$ and a, b are constants satisfying $4a^3 + 27b^2 \neq 0$. The group over this elliptic curve is defined to be the set of elements (x, y) satisfying this equation along with the point at infinity $\mathcal{O}$, which is the identity element. The group operation on the elements in this group is called elliptic curve point addition and denoted by $+$. This group is denoted by $E(\mathbb{Z}_p)$ and its order by n .

This group operation can be used to define another operation on the elements called point multiplication denoted by $\cdot$. For a point $G \in E(\mathbb{Z}_p)$ and a scalar $k \in \mathbb{Z}_n^*$, the point $k \cdot G$ is defined to be the point G added to itself k times.

In elliptic curve cryptography, a private key is defined to be a scalar $k \in \mathbb{Z}_n \setminus \{0\}$ where $\mathbb{Z}_n \setminus \{0\}$ is notation for the set $\{1, \dots, n-1\}$, and the corresponding public key is the point $k \cdot G$ on an elliptic curve. For instance, in some blockchain protocols, the elliptic curve is chosen to be the secp256k1 elliptic curve, and the values a, b , and p are completely specified by this curve. The order n of this group has been calculated given these values, which in the case of this curve is a prime, and the secp256k1 standard also specifies a point G which is to be used as the generator of this group.

Elliptic Curve Digital Signature Algorithm

In order to create a signature on a message msg , with the private key a , the following steps are taken:

1. Calculate the message digest $e = \text{hash}(msg)$, where may be any hash function. For instance, in some examples $\text{hash}(msg) = \text{SHA256}(\text{SHA256}(msg))$ where $\text{SHA256}(\blacksquare)$ is the SHA-256 hash function. Note that instead the message may be hashed only once, or more that two times with the same or different hash functions.
2. Chose a random integer $k \in \{1, \dots, n-1\}$, where n is the order of the elliptic curve, e.g. the secp256k1 curve. In the following, k is referred to as the ephemeral private key.

3. Calculate the ephemeral public key corresponding to this ephemeral private key $k \cdot G = (R_x, R_y)$.
4. Calculate $r = R_x \bmod n$. If $r = 0$, return to step 2.
5. Calculate the multiplicative inverse of the ephemeral key $k^{-1} \bmod n$.
- 5 6. Calculate $s = k^{-1}(e + ar) \bmod n$. If $s = 0$, return to step 2.
7. The signature on the message msg is (r, s) .

The ephemeral key must be kept secret, otherwise the private key can be calculated, given a message and signature. Additionally, each time a signature is generated, a different
 10 ephemeral key must be used. If this is not the case, it is possible to derive the private key a given two different signatures and their corresponding messages.

Given a message msg , a public key $P = a \cdot G$, and corresponding signature (r, s) , then one can verify the signature by completing the following steps:

- 15 1. Calculate the message digest $e = \text{hash}(msg)$, e.g. $e = \text{SHA256}(\text{SHA256}(msg))$.
2. Calculate the multiplicative inverse s^{-1} of s modulo n .
3. Calculate $j_1 = es^{-1} \bmod n$ and $j_2 = rs^{-1} \bmod n$.
4. Calculate the point $Q = j_1 \cdot G + j_2 \cdot P$.
5. If $Q = \mathcal{O}$, the point at infinity, the signature is invalid.
- 20 6. If $Q \neq \mathcal{O}$, then let $Q := (Q_x, Q_y)$, and calculate $u = Q_x \bmod n$. If $u = r$, the signature is valid.

In threshold signature schemes, this private key a is split into key shares that are distributed amongst participants in a threshold scheme group.

25

Joint Verifiable Random Secret Sharing

Assume that N participants want to create a joint secret that can only be regenerated by at least $(t + 1)$ of the participants in the scheme. To create the shared secret, the following steps are taken:

- 30 1. The participants agree on the unique label i for each participant. Each participant i generates $(t + 1)$ random numbers

$$a_{ij} \in_R \mathbb{Z}_n \setminus \{0\}, \forall j = 0, \dots, t,$$

where $\in_R$ means a randomly generated element of the set $\mathbb{Z}_n \setminus \{0\}$ where $\mathbb{Z}_n \setminus \{0\}$ is notation for the set $\{1, \dots, n-1\}$. Then each participant has a secret polynomial of order t

$$f_i(x) = a_{i0} + a_{i1}x + \dots + a_{it}x^t \bmod n,$$

for $i = 1, \dots, N$. Note that we omit the $\bmod n$ notation from now on, and it is assumed that all arithmetic operations over integers are done modulo n .

2. Each participant i sends the value $f_i(j)$ to participant j e.g. using a secure communication channel with participant j only.
3. Each participant i calculates their own private secret share of a shared secret polynomial as

$$a_i := \sum_{j=1}^N f_j(i).$$

- 15 A shared secret share is a point with the form (i, a_i) , where i is the participants label in the scheme. This method for creating a secret share of a , as described in steps 1-3, is denoted herein by $a_i = JVRSS(i)$ for participant i . Note that "JVRSS" typically stands for "Joint verification random secret sharing" and includes steps 4 and 5 as well. However, throughout this document JVRSS is taken to mean performing at least steps 1 to 3, where steps 4 and 5 are optional steps.

Now that the participants have generated a shared polynomial, they can each verify that the other participants have shared the correct information to all participants, and that all participants have the same shared polynomial. This is done in the following way.

25

4. Each participant i broadcasts to all participants the obfuscated coefficients

$$a_{ik} \cdot G,$$

for $k = 0, \dots, t$.

5. Each participant i checks that each participant j has correctly calculated the polynomial point $f_j(i)$ by calculating $f_j(i) \cdot G$ and verifying that

30

$$f_j(i) \cdot G \stackrel{?}{=} \sum_{k=0}^t i^k (a_{jk} \cdot G) \quad \forall j = 1, \dots, N.$$

If all participants find that this equation holds for each polynomial, then the group can collectively be sure that they have all created the same shared polynomial.

5

Reconstructing a shared secret

Assume a participant wants to reconstruct a shared secret a which is the zeroth order of a shared polynomial. Given $(t + 1)$ points on this polynomial of the form

$$(1, a_1), \dots, ((t + 1), a_{t+1}),$$

10 then to find the shared secret a , one calculates

$$\text{interpolate}(a_1, \dots, a_{t+1}) = \left(\sum_{l=1}^{t+1} a_l \prod_{\substack{1 \leq j \leq (t+1), \\ j \neq l}} (-j) (l - j)^{-1} \right) = a,$$

which is derived from a general formula known as “Lagrange Interpolation”.

15 Public Key calculation

Given the N zeroth-order private polynomial coefficient public keys $a_{i0} \cdot G$ for $i = 1, \dots, N$ shared in step 4 of JVRSS, each participant calculates the shared public key P using

$$P = a \cdot G = \sum_{j=1}^N a_{j0} \cdot G,$$

corresponding to the shared secret a .

20

Addition of shared secrets

To calculate the addition of two shared secrets that are shared amongst a group of N participants, where each secret polynomial has order t , without any entity knowing the individual secrets, the following steps are taken:

- 25 1. Generate the first shared secret a , where participant i 's share is given by $a_i = JVRSS(i)$ for $i = 1, \dots, N$ with a threshold of $(t + 1)$.

2. Generate the second shared secret b , where participant i 's share is given by $b_i = JVRSS(i)$, with a threshold of $(t + 1)$.

3. Each participant i calculates their own additive share

$$v_i = a_i + b_i \bmod n .$$

4. All participants broadcast their additive share v_i to all other participants.
5. Each participant interpolates over at least $(t + 1)$ of the shares v_i to calculate

$$v = \text{interpolate}(v_1, \dots, v_{t+1}) = a + b .$$

This method for the addition of shared secrets is denoted by $ADDSS(i)$ for participant i , which results in each participant i knowing $v = (a + b)$.

Product of shared secrets

To calculate the product of two shared secrets that are both shared amongst a group of N participants, where each secret polynomial has order t , the group takes the following steps:

1. Generate the first shared secret a , where participant i 's share is given by $a_i = JVRSS(i)$ for $i = 1, \dots, N$. The shared secret polynomial has order t , meaning $(t + 1)$ participants are required to recreate it.

2. Generate the second shared secret b , where participant i 's share is given by $b_i = JVRSS(i)$, and the shared secret polynomial again has order t .

3. Each participant calculates their own multiplicative share μ_i using

$$\mu_i = a_i b_i .$$

4. All participants broadcast their multiplicative share μ_i to all other participants.
5. Each participant interpolates over at least $(2t + 1)$ of the shares μ_i to calculate

$$\mu = \text{interpolate}(\mu_1, \dots, \mu_{2t+1}) = ab .$$

This method for calculating the product of two shared secrets is denoted herein by $\mu = ab = PROSS(i)$ for participant i .

Inverse of a shared secret

In order to calculate the inverse of a shared secret a , the following steps are taken:

1. All participants calculate the product of shared secrets $PROSS(i)$, the result of which is $\mu = ab \bmod n$.

2. Each participant calculates the modular inverse of μ which results in

$$\mu^{-1} = (ab)^{-1} \bmod n .$$

- 5 3. Each participant i calculates their own inverse secret share by calculating

$$a_i^{-1} = \mu^{-1} b_i .$$

This method for calculating the inverse of shared secrets is denoted by $a_i^{-1} = INVSS(i)$ for participant i .

10

Shared private key generation and verification

To calculate a shared private key a between $N \geq 2t + 1$ participants, $t + 1$ of which are required to create a signature, the participants execute JVRSS with a threshold of $t + 1$ and public key calculation as described above. The result is that every participant $i = 1, \dots, N$ has a private key share a_i and the corresponding shared public key $P = (a \cdot G)$.

15

Ephemeral key shares generation

To generate ephemeral key shares and the corresponding r , as is required in a signature, a group of size N with a shared private key a of threshold $(t + 1)$ execute the following steps:

20

1. Generate the inverse share of a shared secret $k_i^{-1} = INVSS(i)$, where $(t + 1)$ shares are required to recreate it.
2. Each participant calculates

$$(x, y) = \sum_{i=1}^N (k_{i0} \cdot G) ,$$

using the obfuscated coefficients shared in the verification of k_i , then they

25

calculate

$$r = x \bmod n .$$

3. Each participant i stores (r, k_i^{-1}) .

GENERATING SHARED KEYS

30

Embodiments of the present invention enable two parties to generate a shared cryptographic key, where at least one of those parties comprises a group of participants (e.g. users or machines). Figure 1 illustrates an example system 100 for generating a shared key. As shown, the system 100 comprises a first party which comprises a first coordinator 101 and a first group of participants 102. Only three participants 102 are shown in Figure 1, but it will be appreciated that in general the first group may comprise any number of participants. Furthermore, in Figure 1 the first coordinator 101 is shown as being distinct from the participants 102, but in some embodiments the first coordinator 101 may also be one of the participants 102, e.g. the first participant 102a. Also shown in Figure 1 is a second party comprising a second coordinator 103.

As shown in Figure 2, in some embodiments the second party may also comprise a second group of participants 104. As is the case for the first group of participants, the second group may in general comprise any number of participants 104.

The first coordinator 101, the second coordinator 103, each of the first group of participants, and each of the second group of participants, operate respective computing equipment. Each of the respective computing equipment comprises respective processing apparatus comprising one or more processors, e.g. one or more central processing units (CPUs), accelerator processors (GPUs), application specific processors and/or field programmable gate arrays (FPGAs). The respective computing equipment may also comprise memory, i.e. computer-readable storage in the form of a non-transitory computer-readable medium or media. The memory may comprise one or more memory units employing one or more memory media, e.g. a magnetic medium such as a hard disk; an electronic medium such as a solid-state drive (SSD), flash memory or EEPROM; and/or an optical medium such as an optical disk drive. The respective computing equipment may comprise at least one user terminal, e.g. a desktop or laptop computer, a tablet, a smartphone, or a wearable device such as a smartwatch. Alternatively or additionally, the respective computing equipment may comprise one or more other networked resources, such as cloud computing resources accessed via the user terminal (the cloud computing resources comprising resources of one or more physical server devices implemented at one or more sites). It will be appreciated that any act described as being performed by a party

(e.g. coordinator, participant, etc.) may be performed by the respective computing apparatus operated by that party.

Referring back to Figure 1, the first coordinator 101 (on behalf of the first group of participants) and the second coordinator 103 wish to generate a shared key. This key will be referred to as a shared Diffie-Hellman (DH) key. This label is used merely for convenience and does not place any limitations on the key other than those imposed by the present description.

10 In the embodiment of Figure 1, the second coordinator 103 has a public key corresponding to a private key. "Having" a key means storing that key in memory, or being otherwise able to access and retrieve that key. For example, a key may be written down on paper and then input to a device, e.g. via a user interface. In this embodiment, the private key is a complete (i.e. full, whole, etc.) key. A private key may be said to be a complete key if it directly maps,
15 by way of a generator point, to the corresponding public key.

Each participant 102 of the first group has a share (a "private key share") of a private key. This private key will be referred to as the first private key. The private key associated with the second coordinator 103 will be referred to as the second private key. The first private
20 key and the second private key are different keys (e.g. different integers). In some examples, the first coordinator 101 also has a share of the first private key.

The first private key is a threshold private key. This means that at least a threshold number of the first private key shares are required to reconstruct the first private key. Techniques
25 for generating shares of a threshold private key per se will be familiar to the skilled person. One such technique is known as joint verifiable random secret sharing (JVRSS) and is described above. The first group of participants may each use JVRSS to generate their respective shares of the first private key. Another technique is known as Shamir's secret sharing scheme (SSSS). The first group of participants may each use SSSS to obtain their
30 respective shares of the first private key. Note that SSSS requires a coordinator 101, whereas JVRSS does not. Another technique for generating and distributing private key shares is described in WO2017145010A1.

The first private key has a corresponding public key - a first public key. The first public key may be known to the first coordinator 101 and/or one, some or all of the participants 102.

- 5 In embodiments, the first coordinator 101 may obtain a public key (a second public key) corresponding to the second private key. For example, the second coordinator 103 may transmit the second public key to the first coordinator 101. Alternatively, the second public key may be obtained from a publicly accessible source such as a website or a blockchain. Alternatively, the first coordinator 101 may already have access to the second public key,
10 e.g. it may be stored in memory.

- The first coordinator 101 may send the second public key to some or all of the participants 102 of the first group. The first coordinator 101 may send the second the second public key to each participant 102 separately, or the first coordinator 101 may broadcast the second
15 public key to the first group. It is also not excluded that some or all of the participants 102 may already have access to the second public key, in which case the first coordinator 101 does not have to send the second public key to those participants 102, although the first coordinator 101 may still choose to do so.

- 20 The first coordinator 101 obtains at least a threshold number of respective shares of a shared DH key. For example, the first coordinator 101 may send (e.g. broadcast) a request to the participants 102 for a respective share of the shared DH key. As shown in Figure 1, in some embodiments not all of the participants are required to generate a share of the shared DH key. This will depend on the threshold of the first private key. Each share of the
25 shared DH key is generated based on (i.e. is a function of) a different respective share of the first private key, i.e. each share of the shared DH key is generated by a different participant 102. Each share of the shared DH key is also generated based on the second public key. Recall from above that each participant either already has the second public key or it may be obtained from the first coordinator 101. In some examples, one of the shares is
30 generated by the first coordinator 101.

Each share of the shared DH key may be generated by performing elliptic curve multiplication of the respective first private key share and the second public key. It is not excluded that the shares of the shared DH key are calculated using alternative mathematical operations.

5

Each of the participants 102 that generates a share of the shared DH key may transmit their share directly to the first coordinator 101, or via one or more other participants. The shares may be transmitted over a secure communication channel.

10 Having obtained the required number of different shares of the shared DH key, the first coordinator 101 generates the complete shared DH key. The shared DH key is a function of the obtained shares. For example, the shared DH key may be obtained by performed elliptic curve interpolation of the shares of the shared DH key.

15 The second coordinator 103 is configured to generate the same shared DH key. In some embodiments, the second coordinator 103 generates the shared DH key based on the first public key and the second private key. In this case the second coordinator has access to the full second private key, i.e. not just a share of the second private key. The second coordinator 101 may already have access to the first public key, e.g. it may be obtained from
20 a publicly accessible source such as the blockchain. Alternatively, the first coordinator 101 may send the first public key to the second coordinator 103. In some examples, each participant 102 may have a share of the first public key corresponding to their respective share of the first private key. The first coordinator 101 may require a threshold number of shares of the first public key in order to generate the first public key, before sending it to
25 the second coordinator 103.

As mentioned above, the first private key shares may be generated using JVRSS or an equivalent scheme. In these embodiments, each participant of the first group has a respective zeroth order coefficient of a private polynomial (e.g. see step 1 of the JVRSS
30 method described above). Now instead of calculating their respective share of the shared DH key based on their respective share of the first private key, each participant generates their respective share based on their respective zeroth order coefficient. Each share of the

shared DH key is also based on the second public key. In these examples, a respective share of the shared DH key must be obtained from each participant 102.

Referring to Figure 2, the second private key may be a threshold private key and each participant 104 of the second group may have a share of the second private key. In these embodiments, the second coordinator 103 may perform equivalent operations to the first coordinator in order to generate the shared DH key based on respective shares of the shared DH key generated by the participants 104 of the second group. E.g. the participants of the second group may generate shares of the shared DH key based on the first public key and either their respective share of the second private key, or their respective zeroth order coefficient of their respective private polynomial.

Regardless of the method chosen by the second coordinator 103, both the first coordinator 101 and the second coordinator 103 will have the same shared DH key. The shared DH key may then be used to encrypt messages, i.e. any type of data. For example, the first coordinator 101 may encrypt a message and send the encrypted message to the second coordinator 103, or vice versa. The message may comprise personal and/or confidential information, e.g. financial data, medical data, prescription data, etc. The message may comprise a contract or other type of document.

The shared DH key may be used as a symmetric encryption key, in which case it can be used to both encrypt and decrypt messages. Any suitable symmetric encryption scheme may be used, e.g. data encryption standard (DES), triple DES, Blowfish, advanced encryption standard (AES), Rivest Cipher 4 (RC4), RC5, or RC6. For example, the first coordinator 101 may receive an encrypted message from the second coordinator 103 which has been encrypted with the shared DH key. The first coordinator 101 may then use the shared DH key to decrypt the message (i.e. decrypt the ciphertext to obtain the plaintext message).

In some examples, the shared DH key may be used as an asymmetric key, e.g. a public key. In this case, a message may be encrypted with the shared DH key such that it can only be decrypted with the corresponding private key. Neither the first coordinator 101 nor the second coordinator 103 has enough information to compute the corresponding private key.

In order to do so, the first coordinator 101 and the second coordinator 103 may perform threshold computation (e.g. interpolation over the private key shares) to compute the private key corresponding to the shared DH key.

- 5 In some examples, having obtained the private key corresponding to the shared DH key, the first coordinator 101 may generate a digital signature based on a message to be signed and the private key. The second coordinator 103 may also generate a digital signature using the private key. In some examples, a blockchain transaction may be created (e.g. by the first coordinator 101 or the second coordinator) that comprises an output locked to the shared
- 10 DH key. The first coordinator 101 or the second coordinator 102 may unlock the output by generating a blockchain transaction that comprises an input referencing the output of the earlier blockchain transaction, and comprising a digital signature generated using the private key and a message comprising at least part of the transaction.
- 15 The first coordinator 101 may generate one or more additional cryptographic keys based on the shared DH key. For example, the shared DH key may be input to a hash function (e.g. SHA256, SHA512, etc.). The hash digest may be used as an additional cryptographic key. The shared DH key may be hashed multiple times to generate multiple additional cryptographic keys. Additional cryptographic keys may be generated in alternative ways. For instance, the
- 20 shared DH key may be combined with a different key (e.g. a different public key) using point addition. The second coordinator 103 can perform the same operation(s) as the first coordinator to generate the same additional cryptographic keys. This allows many keys to be generated from the same shared DH key, e.g. to prevent key re-use.
- 25 Figure 3 shows an example method 300 for generating a shared DH key. It will be appreciated that some of the steps are optional. At step S301, the first coordinator 101 obtains a second public key, e.g. from the second coordinator 103. At step S302, the first coordinator sends a request to the participants 102 of the first group, requesting shares of the shared DH key (the common secret). At step S303, the first coordinator 101 obtains at
- 30 least a threshold number of shared DH key shares. At step S304, the first coordinator 101 computes the shared DH key. The shared DH key may then be used for, among other things, encrypting messages.

The following provides further examples of the described embodiments. In the following examples, Bob is equivalent to the first coordinator 101 and Alice is equivalent to the second coordinator 103.

5

In order to calculate a shared DH key where at least one of the keys is a shared secret between a group of N participants, the following steps may be taken. Assume that Alice wants to create a secret channel with Bob's threshold group. That is, Bob's group has N participants, each with a share b_i of the private key b and assume that $(t + 1)$ of them would need to cooperate to calculate the private key. The corresponding public keys of Alice and Bob's group are assumed to be public.

10

In order to calculate a shared DH key requiring $(t + 1)$ of the group to cooperate, the following steps may be taken.

15

1. Alice 103 receives the public key of Bob's group $b \cdot G$ and calculates $a(b \cdot G)$ using point multiplication.
2. A coordinator Bob 101 of Bob's group requests at least $(t + 1)$ shares of a shared DH key using $a \cdot G$.
3. At least $(t + 1)$ participants in Bob's group calculate $b_i(a \cdot G)$ using Alice's public key.
4. The $(t + 1)$ participants send their share to Bob 101 using a secure communication channel.
5. Bob 101 calculates the shared DH key using

20

25

$$b(a \cdot G) = ECinterpolate(b_1(a \cdot G), \dots, b_{t+1}(a \cdot G)),$$

where *ECinterpolate* is the same as the equation for normal interpolation but using point addition instead of the usual addition over the natural numbers.

30

Now both Alice 103 and Bob's group have a shared key $ab \cdot G$ that they can use to encrypt messages. The shared key may be used for symmetric or asymmetric encryption. If used for the latter, the corresponding private key ab may be calculated using threshold computation.

- 5 If Alice's private key is a shared secret as well, her group would also execute the same steps as in steps 2-5.

There is also another way to calculate the shared DH key avoiding elliptic curve interpolation, but each participant must contribute. The participants may take the following
10 steps.

1. Alice 103 receives the public key of Bob's group $b \cdot G$ and calculates $a(b \cdot G)$.
2. A coordinator Bob 101 of Bob's group requests all N participants to calculate a share of a DH key using $a \cdot G$.
- 15 3. The N participants in Bob's group calculate $b_{i0}(a \cdot G)$ using Alice's public key, where b_{i0} is the zeroth order of participant i 's private polynomial.
4. The N participants send their share to Bob 101 using a secure communication channel or broadcasting to the scheme participants only.
5. Bob 101 calculates the shared DH key using

$$20 \quad b(a \cdot G) = \sum_{i=1}^N b_{i0}(a \cdot G) .$$

In this equation, the summation is point addition.

If Alice 101 is part of a threshold scheme as well, her group would also execute steps 2-5.

25

In step 4 of both schemes, the participants should preferably send their shares via a secure communication channel with Bob 101. If they send their share on a public network and they are obtained, then anyone may be able to calculate the shared DH key. This contradicts that the shared DH key should be a shared secret between Alice 103 and Bob's group.

30

The two methods may be combined. For example, Bob's group may calculate the shared DH key using the first method whilst Alice's group calculates the shared DH key using the second method, or vice versa. The first method is slightly slower than the second method, but it allows for the shared DH key to be recovered if lost. Conversely, the second method is faster than the first method, but the shared DH key is not recoverable if a participant of Bob's group loses their respective share of their shared private key. Of course the shared DH key may still be calculated by Alice. The second method is desirable if the use case benefits from contribution by all participants.

10 EXAMPLE USE CASES

Embodiments of the present invention may be used for a variety of use cases, and in particular ones where it is desirable to remove the single point of failure associated with a private key. For instance, if a full private key is lost then a shared key generated based on the private key is not recoverable. Furthermore, if a shared key is generated based on a full private key and that private key is stolen or otherwise compromised, then any message encrypted with the shared key may be decryptable (depending on how the shared key is generated). Embodiments of the present invention are also particularly advantageous where increased security of a message is important such that the message is not easily decryptable. For instance, it is important to keep sensitive data, such as a patient's medical history, safe and secure.

The shared DH key may be used to encrypt and share a patient's medical data. For example, a patient may choose to share his/her medical data with a third party service provider or data company. E.g. the patient may sell his/her data to a data analytics company. In this example the patient (equivalent to the second coordinator 103) may have a private-public key pair. The service provider, the patient and the patient's doctor (equivalent to the first group of participants 102, with the service provider also being the first coordinator 101) may each have shares of a shared private key. At least two of the three parties in this example are required to construct the shared DH key. The patient and the service provider use embodiments of the present invention to construct the shared DH key. The service provider will only be able to construct the same shared DH key if the patient and/or their

doctor provide a share of the shared DH key to the service provider. The patient may then encrypt his/her medical data with the shared DH key and send the encrypted data to the service provider. The service provider may then use the shared DH key to decrypt and gain access to the patient's medical data.

5

As another example, a shared DH key may be used to share media content, e.g. as part of a streaming service. For example, a content provider (second coordinator 103) may want to stream a movie to a user. The content provider has a private-public key pair. The content provider and the user have shares of a shared private key. In this example the user is
10 equivalent to the first coordinator 101. The content provider generates a shared DH key using its private key and the public key corresponding to the shared private key. The content provider encrypts part or all of the movie and sends the encrypted data to the user. The user may generate a share of the shared DH key using the user's share of the private key and the content provider's public key. In this example all owners of a private key share are
15 required to participate to generate the shared DH key. Therefore the user will only be able to generate the full shared DH key if the content provider also provides a share of the shared DH key. The content provider may choose to provide a share of the shared DH key in return for payment from the user.

20 In general, the present invention can be used to encrypt any message. As a particular example use case, the message may be part or all of a blockchain transaction. Additionally or alternatively, the encrypted message may be included in a blockchain transaction.

Threshold signature schemes have been briefly discussed above. With such a scheme,
25 embodiments of the present invention may be used to store an encrypted share of a shared secret, i.e. a share of a private key used to generate a share of the threshold signature. Then, if the share of the shared secret is lost, the share can be recovered with a threshold number of participants agreeing to decrypt the share for the participant who has lost their share.

30

Figure 4 illustrates an example transaction protocol for use as part of a blockchain protocol. Example blockchain protocols are well documented in the literature, but a description of an

example protocol transaction is provided here for completeness. This is an example of a UTXO-based protocol. A transaction 152 (abbreviated “Tx”) is the fundamental data structure of the blockchain (each block of the blockchain comprising one or more transactions 152). The following will be described by reference to an output-based or
5 “UTXO” based protocol. However, this not limiting to all possible embodiments.

In a UTXO-based model, each transaction (“Tx”) 152 comprises a data structure comprising one or more inputs 202, and one or more outputs 203. Each output 203 may comprise an unspent transaction output (UTXO), which can be used as the source for the input 202 of
10 another new transaction (if the UTXO has not already been redeemed). The UTXO includes a value specifying an amount of a digital token, e.g. representing an amount of a digital asset. This represents a set number of tokens on the (distributed) ledger. The UTXO may also contain the transaction ID of the transaction from which it came, amongst other information. The transaction data structure may also comprise a header 201, which may
15 comprise an indicator of the size of the input field(s) 202 and output field(s) 203. The header 201 may also include an ID of the transaction. In embodiments the transaction ID is the hash of the transaction data (excluding the transaction ID itself) and stored in the header 201 of the raw transaction 152 submitted to the miners.

20 Say a first user, e.g. Alice, wishes to create a transaction 152j transferring an amount of the digital token in question to a second user, e.g. Bob. In Figure 4 Alice’s new transaction 152j is labelled “ T_{X_I} ”. It takes an amount of the digital token that is locked to Alice in the output 203 of a preceding transaction 152i in the sequence, and transfers at least some of this to Bob. The preceding transaction 152i is labelled “ T_{X_O} ” in Figure 4. T_{X_O} and T_{X_I} are just
25 arbitrary labels. They do not necessarily mean that T_{X_O} is the first transaction in the blockchain, nor that T_{X_I} is the immediate next transaction in the pool. T_{X_I} could point back to any preceding (i.e. antecedent) transaction that still has an unspent output 203 locked to Alice.

30 The preceding transaction T_{X_O} may already have been validated and included in the blockchain at the time when Alice creates her new transaction T_{X_I} , or at least by the time she sends it to the network 106. It may already have been included in one of the blocks at

that time, or it may be still waiting in the pool 154 in which case it will soon be included in a new block. Alternatively T_{x0} and T_{x1} could be created and sent to the blockchain network together, or T_{x0} could even be sent after T_{x1} if the node protocol allows for buffering “orphan” transactions. The terms “preceding” and “subsequent” as used herein in the context of the sequence of transactions refer to the order of the transactions in the sequence as defined by the transaction pointers specified in the transactions (which transaction points back to which other transaction, and so forth). They could equally be replaced with “predecessor” and “successor”, or “antecedent” and “descendant”, “parent” and “child”, or such like. It does not necessarily imply an order in which they are created, sent to the network, or arrive at any given node. Nevertheless, a subsequent transaction (the descendent transaction or “child”) which points to a preceding transaction (the antecedent transaction or “parent”) will not be validated until and unless the parent transaction is validated. A child that arrives at a node before its parent is considered an orphan. It may be discarded or buffered for a certain time to wait for the parent, depending on the node protocol and/or miner behaviour.

One of the one or more outputs 203 of the preceding transaction T_{x0} comprises a particular UTXO, labelled here $UTXO_o$. Each UTXO comprises a value specifying an amount of the digital token represented by the UTXO, and a locking script which defines a condition which must be met by an unlocking script in the input 202 of a subsequent transaction in order for the subsequent transaction to be validated, and therefore for the UTXO to be successfully redeemed. Typically the locking script locks the amount to a particular party (the beneficiary of the transaction in which it is included). I.e. the locking script defines an unlocking condition, typically comprising a condition that the unlocking script in the input of the subsequent transaction comprises the cryptographic signature of the party to whom the preceding transaction is locked.

The locking script (aka scriptPubKey) is a piece of code written in the domain specific language recognized by the node protocol. A particular example of such a language is called “Script” (capital S). The locking script specifies what information is required to spend a transaction output 203, for example the requirement of Alice’s signature. Unlocking scripts appear in the outputs of transactions. The unlocking script (aka scriptSig) is a piece of code

written the domain specific language that provides the information required to satisfy the locking script criteria. For example, it may contain Bob's signature. Unlocking scripts appear in the input 202 of transactions.

- 5 So in the example illustrated, $UTXO_0$ in the output 203 of TX_0 comprises a locking script [Checksig P_A] which requires a signature Sig P_A of Alice in order for $UTXO_0$ to be redeemed (strictly, in order for a subsequent transaction attempting to redeem $UTXO_0$ to be valid). [Checksig P_A] contains the public key P_A from a public-private key pair of Alice. The input 202 of TX_1 comprises a pointer pointing back to TX_0 (e.g. by means of its transaction ID,
- 10 $TXID_0$, which in embodiments is the hash of the whole transaction TX_0). The input 202 of TX_1 comprises an index identifying $UTXO_0$ within TX_0 , to identify it amongst any other possible outputs of TX_0 . The input 202 of TX_1 further comprises an unlocking script <Sig P_A > which comprises a cryptographic signature of Alice, created by Alice applying her private key from the key pair to a predefined portion of data (sometimes called the "message" in
- 15 cryptography). What data (or "message") needs to be signed by Alice to provide a valid signature may be defined by the locking script, or by the node protocol, or by a combination of these.

- When the new transaction TX_1 arrives at a node, the node applies the node protocol. This
- 20 comprises running the locking script and unlocking script together to check whether the unlocking script meets the condition defined in the locking script (where this condition may comprise one or more criteria). In embodiments this involves concatenating the two scripts:

<Sig P_A > < P_A > || [Checksig P_A]

25

- where "||" represents a concatenation and "<...>" means place the data on the stack, and "[...]" is a function comprised by the unlocking script (in this example a stack-based language). Equivalently the scripts may be run one after the other, with a common stack, rather than concatenating the scripts. Either way, when run together, the scripts use the
- 30 public key P_A of Alice, as included in the locking script in the output of TX_0 , to authenticate that the locking script in the input of TX_1 contains the signature of Alice signing the expected portion of data. The expected portion of data itself (the "message") also needs to be

included in TX_0 in order to perform this authentication. In embodiments the signed data comprises the whole of TX_0 (so a separate element does not need to be included specifying the signed portion of data in the clear, as it is already inherently present).

- 5 The details of authentication by public-private cryptography will be familiar to a person skilled in the art. Basically, if Alice has signed a message by encrypting it with her private key, then given Alice's public key and the message in the clear (the unencrypted message), another entity such as a node of the blockchain network is able to authenticate that the encrypted version of the message must have been signed by Alice. Signing typically
- 10 comprises hashing the message, signing the hash, and tagging this onto the clear version of the message as a signature, thus enabling any holder of the public key to authenticate the signature. Note therefore that any reference herein to signing a particular piece of data or part of a transaction, or such like, can in embodiments mean signing a hash of that piece of data or part of the transaction.

15

- If the unlocking script in TX_1 meets the one or more conditions specified in the locking script of TX_0 (so in the example shown, if Alice's signature is provided in TX_1 and authenticated), then the blockchain node deems TX_1 valid. If it is a mining node, this means it will add it to the pool of transactions awaiting proof-of-work. If it is a forwarding node, it will forward the
- 20 transaction TX_1 to one or more other nodes in the blockchain network, so that it will be propagated throughout the network. Once TX_1 has been validated and included in the blockchain, this defines $UTXO_0$ from TX_0 as spent. Note that TX_1 can only be valid if it spends an unspent transaction output 203. If it attempts to spend an output that has already been spent by another transaction, then TX_1 will be invalid even if all the other
- 25 conditions are met. Hence the node 104 also needs to check whether the referenced UTXO in the preceding transaction TX_0 is already spent (has already formed a valid input to another valid transaction). This is one reason why it is important for the blockchain 150 to impose a defined order on the transactions. In practice a given node 104 may maintain a separate database marking which UTXOs 203 in which transactions have been spent, but
- 30 ultimately what defines whether a UTXO has been spent is whether it has already formed a valid input to another valid transaction in the blockchain.

If the total amount specified in all the outputs 203 of a given transaction is greater than the total amount pointed to by all its inputs 202, this is another basis for invalidity in most transaction models. Therefore such transactions will not be propagated nor mined into blocks.

5

Note that the script code is often represented schematically (i.e. not the exact language). For example, one may write [Checksig P_A] to mean [Checksig P_A] = OP_DUP OP_HASH160 <H(P_A)> OP_EQUALVERIFY OP_CHECKSIG. "OP_..." refers to a particular opcode of the Script language. OP_CHECKSIG (also called "Checksig") is a Script opcode that takes two inputs (signature and public key) and verifies the signature's validity using the Elliptic Curve Digital Signature Algorithm (ECDSA). At runtime, any occurrences of signature ('sig') are removed from the script but additional requirements, such as a hash puzzle, remain in the transaction verified by the 'sig' input. As another example, OP_RETURN is an opcode of the Script language for creating an unspendable output of a transaction that can store metadata within the transaction, and thereby record the metadata immutably in the blockchain. E.g. the metadata could comprise a document which it is desired to store in the blockchain.

10

15

20

The signature P_A is a digital signature. In embodiments this is based on the ECDSA using the elliptic curve secp256k1. A digital signature signs a particular piece of data. In embodiments, for a given transaction the signature will sign part of the transaction input, and all or part of the transaction output. The particular parts of the outputs it signs depends on the SIGHASH flag. The SIGHASH flag is a 4-byte code included at the end of a signature to select which outputs are signed (and thus fixed at the time of signing).

25

30

The locking script is sometimes called "scriptPubKey" referring to the fact that it comprises the public key of the party to whom the respective transaction is locked. The unlocking script is sometimes called "scriptSig" referring to the fact that it supplies the corresponding signature. However, more generally it is not essential in all applications of a blockchain that the condition for a UTXO to be redeemed comprises authenticating a signature. More generally the scripting language could be used to define any one or more conditions. Hence the more general terms "locking script" and "unlocking script" may be preferred.

According to some embodiments of the present invention, a transaction may be a pay-to-public-key-hash (P2PKH) output which is locked to a hash of the shared DH key. In order to be unlocked, an input of a later transaction that references the P2PKH output needs to include the (unhashed) shared DH key and a signature generated based on the private key
5 corresponding to the shared DH key. Represented in script, the “locking script” and “unlocking script” may take the following forms:

Locking script = OP_DUP OP_HASH160 <Public KeyHash> OP_EQUAL OP_CHECKSIG

Unlocking script = <Signature> <Public Key>

10

It will be appreciated that the above embodiments have been described by way of example only. More generally there may be provided a method, apparatus or program in accordance with any one or more of the following Statements.

15 Statement 1. A computer-implemented method of generating a shared cryptographic key based on at least one shared secret, wherein each participant belonging to a first group has a respective share of a first secret, the first secret having a first threshold and a corresponding first public key, wherein a second coordinator has a second public key corresponding to a second secret, and wherein the method is performed by a first
20 coordinator of the first group and comprises:

obtaining, from at least the first threshold number of participants of the first group, respective shares of the shared cryptographic key, where each respective share of the shared cryptographic key is based on i) a respective share of the first secret or a respective zeroth order coefficient of a respective private polynomial used to calculate the respective
25 share of the first secret, and ii) the second public key; and

generating the shared cryptographic key based on the obtained respective shares of the cryptographic key, wherein the second coordinator is configured to generate the same shared cryptographic key.

30 Statement 2. The method of statement 1, wherein the coordinator is one of said participants belonging to the first group.

Statement 3. The method of statement 1 or statement 2, comprising:

obtaining the second public key; and

transmitting the second public key to each participant of the first group.

- 5 Statement 4. The method of any preceding statement, comprising transmitting the first public key to second coordinator.

Statement 5. The method of statement 4, comprising:

- 10 obtaining, from at least the first threshold number of participants of the first group, respective shares of the first public key, wherein each respective share of the first public key is based on the respective share of the first secret and a public key generator.

- Statement 6. The method of statement 3 or any statement dependent thereon, wherein said obtaining of the second public key comprises receiving the second public key from the
15 second coordinator.

- Statement 7. The method of any preceding statement, wherein said respective share of the shared DH key is generated based on the respective share of the first secret, and wherein said generating of the shared cryptographic key comprises performing elliptic curve
20 interpolation over the obtained respective shares of the shared cryptographic key.

- Statement 8. The method of any of statements 1 to 6, wherein said respective share of the shared cryptographic key is generated based on the respective zeroth order coefficient of a respective private polynomial used to calculate the respective share of the first secret,
25 wherein said obtaining of the respective shares of the shared cryptographic key comprises obtaining a respective share of the shared cryptographic key from each of the first group of participants, and wherein said generating of the shared cryptographic key comprises performing point addition of the obtained respective shares of the shared cryptographic key.

30

- Statement 9. The method of any preceding statement, comprising encrypting a first message with the shared cryptographic key.

Statement 10. The method of statement 9, comprising transmitting the encrypted first message to the second coordinator and/or to a different party.

- 5 Statement 11. The method of statement 9 or statement 10, comprising:
- generating a blockchain transaction, wherein the blockchain transaction comprises the encrypted message; and
 - making the blockchain transaction available to one or more nodes of a blockchain network.

10

Statement 12. The method of any preceding statement, wherein the shared cryptographic key is a symmetric key, wherein a second message has been encrypted with the shared cryptographic key, and wherein the method comprises decrypting the second message using the shared cryptographic key.

15

- Statement 13. The method of any preceding statement, comprising:
- obtaining a private key corresponding to the shared cryptographic key; and
 - generating a digital signature using the corresponding private key, and/or using the corresponding private key to decrypt a third message that has been encrypted with the
- 20 shared cryptographic key.

25

Statement 14. The method of statement 13, wherein a first blockchain transaction comprises an output locked to the shared cryptographic key, and wherein the method comprises generating a second blockchain transaction having an input that references the output of the first blockchain transaction and comprises the digital signature for unlocking said output.

30

Statement 15. The method of any preceding statements, comprising generating one or more additional cryptographic keys based on the shared cryptographic key.

Statement 16. The method of statement 15, wherein said generating of the one or more additional cryptographic keys comprises applying a hash function to the shared cryptographic key.

- 5 Statement 17. The method of any preceding statement, wherein the second secret is a shared secret, wherein a second group comprises a plurality of participants, and wherein each participant of the second group has a respective share of a second secret, the second secret having a second threshold.
- 10 Statement 18. The method of statement 17, wherein each participant of the second group has a respective zeroth order coefficient of a respective private polynomial used to calculate the respective share of the second secret.

Statement 19. Computer equipment comprising:

- 15 memory comprising one or more memory units; and
processing apparatus comprising one or more processing units, wherein the memory stores code arranged to run on the processing apparatus, the code being configured so as when on the processing apparatus to perform the method of any of statements 1 to 18.
- 20 Statement 20. A computer program embodied on computer-readable storage and configured so as, when run on computer equipment, to perform the method of any of statements 1 to 18.

- According to another aspect disclosed herein, there may be provided a method comprising
- 25 the actions of the first participant and the key generator.

According to another aspect disclosed herein, there may be provided a system comprising the computer equipment of the first participant and the key generator.

- 30 Other variants or use cases of the disclosed techniques may become apparent to the person skilled in the art once given the disclosure herein. The scope of the disclosure is not limited by the described embodiments but only by the accompanying claims.

CLAIMS

1. A computer-implemented method of generating a shared cryptographic key based on at least one shared secret, wherein each participant belonging to a first group has a respective share of a first secret, the first secret having a first threshold and a corresponding first public key, wherein a second coordinator has a second public key corresponding to a second secret, and wherein the method is performed by a first coordinator of the first group and comprises:

obtaining, from at least the first threshold number of participants of the first group, respective shares of the shared cryptographic key, where each respective share of the shared cryptographic key is based on i) a respective zeroth order coefficient of a respective private polynomial used to calculate the respective share of the first secret, and ii) the second public key; and

generating the shared cryptographic key based on the obtained respective shares of the cryptographic key, wherein the second coordinator is configured to generate the same shared cryptographic key.

2. The method of claim 1, wherein the coordinator is one of said participants belonging to the first group.

3. The method of claim 1 or claim 2, comprising:

obtaining the second public key; and

transmitting the second public key to each participant of the first group.

4. The method of any preceding claim, comprising transmitting the first public key to second coordinator.

5. The method of claim 4, comprising:

obtaining, from at least the first threshold number of participants of the first group, respective shares of the first public key, wherein each respective share of the first public key is based on the respective share of the first secret and a public key generator.

6. The method of claim 3 or any claim dependent thereon, wherein said obtaining of the second public key comprises receiving the second public key from the second coordinator.
7. The method of any preceding claim, wherein said respective share of the shared cryptographic key is generated based on the respective zeroth order coefficient of a respective private polynomial used to calculate the respective share of the first secret, wherein said obtaining of the respective shares of the shared cryptographic key comprises obtaining a respective share of the shared cryptographic key from each of the first group of participants, and wherein said generating of the shared cryptographic key comprises performing point addition of the obtained respective shares of the shared cryptographic key.
8. The method of any preceding claim, comprising encrypting a first message with the shared cryptographic key.
9. The method of claim 8, comprising transmitting the encrypted first message to the second coordinator and/or to a different party.
10. The method of claim 8 or claim 9, comprising:
 - generating a blockchain transaction, wherein the blockchain transaction comprises the encrypted message; and
 - making the blockchain transaction available to one or more nodes of a blockchain network.
11. The method of any preceding claim, wherein the shared cryptographic key is a symmetric key, wherein a second message has been encrypted with the shared cryptographic key, and wherein the method comprises decrypting the second message using the shared cryptographic key.
12. The method of any of claims 1 to 10, comprising:
 - obtaining a private key corresponding to the shared cryptographic key; and

generating a digital signature using the corresponding private key, and/or using the corresponding private key to decrypt a third message that has been encrypted with the shared cryptographic key.

13. The method of claim 12, wherein a first blockchain transaction comprises an output locked to the shared cryptographic key, and wherein the method comprises generating a second blockchain transaction having an input that references the output of the first blockchain transaction and comprises the digital signature for unlocking said output.

14. The method of any preceding claims, comprising generating one or more additional cryptographic keys based on the shared cryptographic key.

15. The method of claim 14, wherein said generating of the one or more additional cryptographic keys comprises applying a hash function to the shared cryptographic key.

16. The method of any preceding claim, wherein the second secret is a shared secret, wherein a second group comprises a plurality of participants, and wherein each participant of the second group has a respective share of a second secret, the second secret having a second threshold.

17. The method of claim 16, wherein each participant of the second group has a respective zeroth order coefficient of a respective private polynomial used to calculate the respective share of the second secret.

18. Computer equipment comprising:
memory comprising one or more memory units; and
processing apparatus comprising one or more processing units, wherein the memory stores code arranged to run on the processing apparatus, the code being configured so as when on the processing apparatus to perform the method of any of claims 1 to 17.

19. A computer program embodied on computer-readable storage and configured so as, when run on computer equipment, to perform the method of any of claims 1 to 17.