(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) **EP 3 935 779 B9**

(12) **CORRECTED EUROPEAN PATENT SPECIFICATION**

(15) Correction information:
**Corrected version no 1 (W1 B1)**
**Corrections, see**
**Description Paragraph(s) 4, 14, 16, 34, 59, 92, 93, 98, 111, 115**
**Claims EN 1, 4**
**Claims FR 1, 4**

(48) Corrigendum issued on:
**02.08.2023 Bulletin 2023/31**

(45) Date of publication and mention of the grant of the patent:
**07.06.2023 Bulletin 2023/23**

(21) Application number: **20702673.3**

(22) Date of filing: **07.02.2020**

(51) International Patent Classification (IPC):
***H04L 9/08*** *(2006.01)* ***H04L 9/30*** *(2006.01)*
***H04L 9/32*** *(2006.01)* ***H04L 9/00*** *(2022.01)*

(52) Cooperative Patent Classification (CPC):
**H04L 9/3013; H04L 9/008; H04L 9/085; H04L 9/3252**

(86) International application number:
**PCT/EP2020/053101**

(87) International publication number:
**WO 2020/177977 (10.09.2020 Gazette 2020/37)**

# (54) A METHOD FOR PROVIDING A DIGITAL SIGNATURE TO A MESSAGE

VERFAHREN ZUM BEREITSTELLEN EINER DIGITALEN SIGNATUR FÜR EINE NACHRICHT

PROCÉDÉ PERMETTANT DE FOURNIR UNE SIGNATURE NUMÉRIQUE À UN MESSAGE

(84) Designated Contracting States:
**AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO RS SE SI SK SM TR**

(30) Priority: **05.03.2019 EP 19160731**

(43) Date of publication of application:
**12.01.2022 Bulletin 2022/02**

(73) Proprietor: **SEPIOR ApS**
**8000 Aarhus C (DK)**

(72) Inventors:
- **JAKOBSEN, Thomas Pelle**
  **8320 Måslet (DK)**
- **DAMGÅRD, Ivan Bjerre**
  **8230 Åbyhøj (DK)**
- **ØSTERGAARD, Michael Bæksvang**
  **8361 Hasselager (DK)**
- **NIELSEN, Jesper Buus**
  **8000 Aarhus C (DK)**

(74) Representative: **Inspicos P/S**
**Agern Allé 24**
**2970 Hørsholm (DK)**

(56) References cited:
**US-A1- 2015 188 713**

- **Rosario Gennaro ET AL: "Robust Threshold DSS Signatures" In: "Serious Games", 1 January 2001 (2001-01-01), Springer International Publishing, Cham 032682, XP055564585, ISSN: 0302-9743 ISBN: 978-3-540-37274-5 vol. 1070, pages 354-371, DOI: 10.1007/3-540-68339-9_31, the whole document**



EP 3 935 779 B9

## Description

### FIELD OF THE INVENTION

**[0001]** The present invention relates to a method for providing a digital signature to a message in accordance with a digital signature algorithm (DSA) or an elliptic curve digital signature algorithm (ECDSA). According to the invention, the digital signature is generated using a multiparty threshold DSA or ECDSA protocol.

### BACKGROUND OF THE INVENTION

**[0002]** Digital signatures may be used for ensuring integrity of transmitted data online, for authentication of data and/or entities online, etc. By using secret signature keys which are generated as secret sharings among a number of parties, each party holding a share of the secret signature key, instead of allowing a single party to hold the entire secret signature key, the risk of the signature system being compromised is reduced. Such a scheme is sometimes referred to as a 'multiparty signature scheme'. In multiparty signature schemes it may be possible to generate a digital signature, even though some of the parties are unavailable, corrupt or compromised. The maximum number of corrupted parties that can be tolerated without violating security is sometimes referred to as the scheme's security threshold, and may be denoted 't'.

**[0003]** Digital signature algorithms have previously been used for generating digital signatures. One example is the DSA standard. It devises a signature scheme parameterised by a cyclic group G of prime order q and generator $g \in G$ as well as two functions $H: \{0,1\}^* \rightarrow Z_q$ and $F: G \rightarrow Z_q$.

**[0004]** The secret signature key x is chosen at random from $Z_q$ and the corresponding public verification key is computed as $y=g^x$. To sign a message M one first chooses a random nonce $k \in Z_q$ and then computes the signature as (r, s), where $r=F(g^k)$ and $s=k^{-1} \cdot (H(M)+r \cdot x)$. Given the public verification key y and a signature (r, s) one can verify the signature by computing m=H(M) and checking that $r=F(g^{m/s} \cdot y^{r/s})$.

**[0005]** Another example is the ECDSA standard which defines signature schemes that essentially work in the same way as DSA, but where G is instead generated by a point g on an elliptic curve. Digital signature algorithms applying elliptic curve cryptography, normally referred to as elliptic curve digital signature algorithms or ECDSAs, have previously been used for generating digital signatures. Such algorithms are known to be suitable for providing reliable digital signatures.

**[0006]** It is common to use multiplicative notation when describing DSA, but additive notation for ECDSA. Here we will use multiplicative notation even though our method can be applied to both DSA and ECDSA. I.e., for elements a, b in the group G we will use $a \cdot b$, or just ab, to denote the group operation applied to the two elements. Computations on elements in the field $Z_q$ are assumed to take place within the field, i.e., when we write $r \cdot x$ or m/s reduction modulo q is implicit.

**[0007]** WO 2015/160839 A1 discloses a system and a method for generation of elliptic curve digital signature algorithm (ECDSA) based digital signatures in a distributed manner, where a secret-share protocol is initialized between a client and a set of n servers to share a set of shares of a private key among the set of n servers. The set of servers initializes a protocol to generate a digital signature on a message using the set of shares of the private key without reconstructing or revealing the private key. A threshold, t, of up to n/2 (i.e. $t \leq n/2$) of the n servers can be maliciously and completely corrupted or compromised, without compromising the confidentiality of the private key or the correctness of the generated signature.

**[0008]** The system and method disclosed in WO 2015/160839 A1 requires a significant number of rounds of interaction in order to generate the digital signature. Furthermore, the system and method requires high processing power and communication bandwidth.

**[0009]** Rosario Gennaro, et al.: "Robust Threshold DSS Signatures", Information and Computation, vol. 164, pages 54-84 (2001), discloses a method for providing digital signatures using a multiparty threshold signature scheme. The protocol described in this article applies polynomial interpolation, and is robust and unforgeable against up to t malicious adversaries when the number of parties, n, is larger than or equal to 3t+1. Thus, in order to tolerate 1 malicious adversary, the number of parties should be at least 4. Furthermore, the method requires a significant number of rounds of interaction in order to generate the digital signature.

### DESCRIPTION OF THE INVENTION

**[0010]** It is an object of embodiments of the invention to provide a method for providing a multiparty DSA or ECDSA based digital signature in which the number of rounds of interaction required in order to generate the digital signature is reduced without compromising confidentiality.

**[0011]** It is a further object of embodiments of the invention to provide a method for providing a multiparty DSA or ECDSA based digital signature in which requirements to processing power is reduced without compromising confidentiality.

**[0012]** It is an even further object of embodiments of the invention to provide a method for providing a multiparty DSA or ECDSA based digital signature in which requirements to communication bandwidth is reduced without compromising confidentiality.

**[0013]** It is an even further object of embodiments of the invention to provide a method for providing a multiparty DSA or ECDSA based digital signature which can tolerate up to t of the parties being maliciously corrupted, where $t>n/3$.

**[0014]** The invention provides a method for providing

a digital signature to a message, M, in accordance with a digital signature algorithm DSA or an elliptic curve digital signature algorithm ECDSA, the method comprising the steps of:

- providing a generator, g, for a cyclic group, G, of order q, where $g \in G$, a function, F, and a function, H, where g, G, F and H are specified by the DSA or ECDSA,
- generating a secret key, x, as a random secret sharing [x] among at least two parties,
- generating random secret sharings, [a] and [k], among the at least two parties and computing [w]=[a][k],
- computing a value, R, as $R=g^k$, without revealing k,
- ensuring that R is correct by verifying that $R=g^k$ is computed from at least t+1 shares of [k] originating from honest parties,
- computing an authenticator, W, as $W=g^{ak}$, by computing $R^a$, without revealing a or k,
- ensuring that W is correct by verifying that $W=R^a$ is computed from at least t+1 shares of [a] originating from honest parties,
- verifying [w] by checking whether or not $g^w=W$, and
- signing the message, M, by computing $[k^{-1}]=[a]\cdot w^{-1}$, computing $[x\cdot k^{-1}]=[x]\cdot[k^{-1}]$, and generating a sharing, [s], among the at least two parties, as a function of M, R, $[k^{-1}]$ and $[x\cdot k^{-1}]$.

**[0015]** Thus, the method of the invention is a method for providing a DSA based or an ECDSA based digital signature to a message, M. The digital signature applied in accordance with the method of the invention may, e.g., be used for ensuring authenticity of a sender, for ensuring integrity of transmitted data, e.g. during online financial transactions, etc.

**[0016]** According to the method of the invention, a generator, g, for a cyclic group, G, of order q, a function, F, and a function, H, are initially provided. The generator, g, is an element of the cyclic group, G, i.e. $g \in G$. The cyclic group, G, the generator g, and the functions, F and H, are all specified by the DSA or ECDSA, and thereby these are defined once it has been determined which specific DSA or ECDSA is to be used.

**[0017]** Next, a secret key, [x], is generated as a random secret sharing among at least two parties, such as among at least three parties. In the present context the term 'secret sharing' should be interpreted to mean that the secret key is distributed among the at least two parties in such a manner that each of the parties holds a share of the secret key, and none of the parties is in the possession of the entire secret key. Accordingly, the method of the invention applies a multiparty system. Thereby none of the parties constitutes a single point of trust, and several of the parties, i.e. at least t+1 parties, need to collude maliciously in order to gain access to the entire secret key. This improves the security of the system, in particular in terms of confidentiality.

**[0018]** An example of a secret sharing scheme is additive sharing. A secret x is said to be additively shared among n parties $P_1, P_2, ..., P_n$ when each party $P_i$ holds a random share $x_i$ such that $x=x_1+x_2+...+x_n$. Additive sharing has threshold t=n-1 since all n shares are required to reconstruct the secret, i.e., reconstructing x requires the collaboration of all n parties, while possession of up to all but one share does not allow reconstruction of x. Another example is Shamir's sharing scheme. Here a secret x is shared when each party $P_i$ holds as its share the value f(i) for a random polynomial f subject to f(0)=x. The degree of a Shamir sharing is defined as the degree of the polynomial f. If this degree is t then any t+1 parties can recombine their shares into the secret using polynomial interpolation, whereas the shares held by any subset of t or less parties reveal nothing about x.

**[0019]** In the present context the term 'random secret sharing' should be interpreted to be a secret sharing where the shares are randomly chosen, such that the shared secret is also a random value. For an additive sharing, this means that the shares are completely random. For a Shamir sharing with threshold t it means that the shares are points on a random degree t polynomial.

**[0020]** Throughout this disclosure, letters arranged in square brackets, '[]', represent sharings of an element among a number of parties. The same letters without the square brackets represent the entire element. Thus, for instance 'x' represents the entire secret key, and [x] represents the secret sharing of the secret key, x.

**[0021]** It is possible to carry out computation on secret sharings without revealing the secrets. This is the subject of the research field 'secure multiparty computation'. For most kinds of secret sharing the parties can easily compute the sum of two secret sharings, i.e., given [x] and [y] they can compute the sharing [x+y] without revealing x or y. Similarly, if all parties agree on a public constant c, they can easily compute from c and [x] the sharing [c·x] without revealing x. We will use the notation [x]+[y] and c·[x] for these operations.

**[0022]** It is also possible to compute from [x] and [y] the product [xy]. Doing so without revealing x and y, when up to a given security threshold (t) of the parties may collude maliciously, is possible, but is often difficult and inefficient. For this reason, for two random sharings [x] and [y] a 'weak' notion of multiplication is sometimes used. A weak multiplication guarantees confidentiality in the sense that no information about x and y leaks as long as at most t parties are malicious. But weak multiplication does not guarantee correctness of the result. Rather, it only computes the correct result [xy] if all parties behave

correctly. A single malicious party could, e.g., spoil the result such that the resulting sharing [z] is not a sharing of xy. In the following, for simplicity, we use the notation [z] = [x] [y] to denote weak multiplication unless otherwise stated.

**[0023]** Weak multiplication can e.g. be constructed for Shamir sharings as long as the security threshold t satisfies t<n/2. For larger thresholds, i.e., up to the maximal threshold t=n-1, weak multiplication can e.g. be obtained for additive sharings by combining an additive homomorphic encryption scheme such as Paillier's encryption scheme with so-called zero knowledge proofs. Examples of such constructions of weak multiplication can be found in the secure multiparty computation research literature.

**[0024]** The parties may, e.g., be in the form of separate servers, which may be physically separated from each other. The parties may preferably be separate or independent in the sense that information being available to one party is unavailable to the other parties. It is, however, not ruled out that some of the parties share some information, as long as no subset of t+1 or more parties is in the possession of all information.

**[0025]** Next, random secret sharings, [a] and [k], are generated among the at least two parties. Accordingly, each of the at least two parties holds a share of a and a share of k. Furthermore, [w]=[a][k] is computed, also as a secret sharing among the at least two parties. This could be performed by each of the at least two parties computing a share of w from its shares of a and k. Furthermore, [w] may be 'opened', i.e. the secret w revealed to each of the at least two parties. This could, e.g., be obtained by each of the at least two parties revealing its share of w to each of the other parties. It should be noted that [w] can be opened without opening [a] and/or [k], i.e. while keeping the values of a as well as k secret.

**[0026]** Next, a value, R, is computed as $R=g^k$, without revealing k. This could, e.g., include each of the at least two parties computing a share of R, based on their share of k. This will be described in further detail below. As an alternative, R may be computed in another manner, as long as it is ensured that k is not revealed. However, R may be revealed to each of the at least two parties. Note that due to the commonly accepted cryptographic assumption known as the 'discrete log' assumption, on which standard DSA/ECDSA is also based, it is not possible to deduce any information about k by seeing the value $R=g^k$.

**[0027]** It is further ensured that R is correct. In the present context this should be interpreted to mean that R equals $g^k$ where k is the unique value k defined by the secret sharing [k]. The step of ensuring that R is correct is performed by verifying that $R=g^k$ is computed from at least t+1 shares of [k] originating from honest parties, where t denotes the threshold of the signature scheme, i.e. up to t malicious, corrupt, dishonest or unavailable parties can be tolerated. Thus, when ensuring the correctness of R, the honesty of the participating parties is investigated, and it is only concluded that R is correct, if it can be demonstrated that at least t+1 participating parties are honest. This will be described in further detail below.

**[0028]** If it turns out that R is incorrect, the signing process may be aborted. For instance, each honest party may be guaranteed to either (1) obtain the value $R=g^k$ or (2) to output abort at this point. I.e., even if up to t parties are corrupted, the remaining n-t parties will either obtain $R=g^k$ or abort. This will be described in further detail below.

**[0029]** Next, an authenticator, W, is computed as $W=g^{ak}$. This is done by computing $R^a$ instead of by computing $g^{ak}$ directly, and without revealing a or k. This may be done in the same manner as the manner in which R was computed, except R is used as a base instead of g. Also in this case, it is ensured that W is correct by verifying that $W=R^a$ is computed from at least t+1 shares of [a] originating from honest parties, and the signing process may be aborted if this is not the case.

**[0030]** Thus, at this stage in the process, a correct $R=g^k$ and a correct $W=g^{ak}$ have been computed, while the values k and a remain secret.

**[0031]** Next, [w] is verified by checking whether or not $g^w=W$. This may include opening [w] to each party. Note that since weak multiplication of sharings was used to compute [w], w only equals the product ak if all parties were honest and followed the protocol until this point. Since $W=g^{ak}$ is already known to be correct, verifying that $g^w=W$ ensures that w=ak. At this point W, g and w are known by all parties, and thereby any party can verify the correctness of its value w in this manner. Moreover, this can be done without revealing a, as well as without revealing k.

**[0032]** Thus, at this stage in the process, it has been ensured that a correct value of w has been computed by each honest party among the at least two parties.

**[0033]** All of the steps described above may be performed without knowledge of the message, M, to be signed. Accordingly, these steps may be performed in a pre-processing process, e.g. during non-peak periods. This may level loads on processing equipment and increase the number of transactions which can be performed during peak periods, and it may reduce response time from a message, M, is presented and until the signature on M is computed.

**[0034]** Finally, the message, M, is signed by computing $[k^{-1}]=[a]\cdot w^{-1}$, computing $[x\cdot k^{-1}]=[x]\cdot[k^{-1}]$, and generating a sharing, [s], among the at least two parties, as a function of M, R, $[k^{-1}]$ and $[x\cdot k^{-1}]$.

**[0035]** A DSA or ECDSA digital signature of M normally consists of a pair, (r, s), where $r=F(g^k)$ and $s=k^{-1}(H(M)+rx)$. The pair, (r, s), can be revealed and verified using the public key y. The generated sharing, [s], is a sharing of the latter part, s, of the signature pair. The first part of the pair, r, may be generated as r=F(R), where F is one of the functions which were initially provided and specified by the ECDSA. The sharing, [s] is generated as follows: First a sharing of the inversion of k is computed

as $[k^{-1}]=w^{-1}[a]$. Since w has been verified to be equal to ak, this yields a correct sharing of $k^{-1}$, because $w^{-1}[a]=[aw^{-1}]=[a(ak)^{-1}]=[(aa^{-1})k^{-1}]=[k^{-1}]$. Furthermore, the sharing $[x\cdot k^{-1}]$ is computed as $[x]\cdot[k^{-1}]$. The sharing [s] can then e.g. be computed by first computing $[A]=H(M)[k^{-1}]$, $[B]=[k^{-1}][x]$, and finally computing $[s]=[A]+r[B]$.

**[0036]** Furthermore, in the method according to the invention, abort is allowed in the case that there is any doubt regarding the correctness of the process, i.e. termination guarantee is not provided, even though this is a requirement in many prior art methods. Such doubt could, e.g., be due to one or more of the parties being compromised or malicious, or it could simply be due to package loss during communication among the parties. Aborting the process would simply result in the process being restarted in order to attempt to provide a correct signature.

**[0037]** Signing methods where termination guarantee is provided usually achieve the termination guarantee by assuming a synchronous network, i.e. a network which guarantees an upper limit on package delay. Applying such methods using an open communication network, such as the Internet, being inherently asynchronous, requires that each party must be provided with a local clock, and if a party does not receive an expected message within a certain fixed timeout, according to the local clock, it will simply treat the sender as corrupt.

**[0038]** This results in a dilemma. If the timeout is small, then there is a high risk that the package delays, which occur frequently on the Internet, will quickly cause more parties to be treated as corrupt than what the protocol can tolerate, e.g., more than n/2 or n/3 of the n parties, and consequently the secret signing key may leak. To avoid this, the timeout must be set very high. But this has the drawback that it allows a single malicious party to introduce a very high latency in the system by intentionally delaying each message that he sends to the other parties, i.e., he can hold back the message until close until the timeout happens. The method according to this invention avoids this dilemma by allowing the protocol to abort. This allows the use of quite small timeouts, yielding only a small overall latency in the system. This is possible because the method is designed to ensure that the worst thing that can happen, even when all messages are delayed, is that the protocol aborts, and not, as in some prior disclosed methods, that the secret signing key leaks.

**[0039]** The step of computing a value, R, may comprise the steps of:

- each of the at least two parties computing a share, $R_j$, of the value, R, as $R_j=g^{k\_j}$, and distributing the share to each of the other parties, and
- computing the value, R, from the shares, $R_j$, and

the step of ensuring that R is correct may comprise the step of:

- each of the parties checking that R is correct, based on the shares, $R_j$, received from the other parties.

**[0040]** Recall that each party, $P_j$, holds a share, $k_j$, of the secret k. According to this embodiment, the value, R, is computed by the at least two parties in the following manner. Each of the at least two parties computes a share, $R_j$, of R, where $R_j$ denotes the share of R which is computed by party j. $R_j$ is computed as $R_j=g^{k\_j}$, where k_j denotes the share of [k] which is held by party j. Thus, each party calculates a share of $R_j$, based on its own share of [k].

**[0041]** Each of the at least two parties then distributes its share, $R_j$, of R to each of the other parties, i.e. the shares, $R_j$, are revealed. However, due to the discrete log assumption, the shares k_j remain secret, and hence the secret nonce k remains secret.

**[0042]** The value, R, is then computed from the revealed shares, $R_j$. This could, e.g., include interpolating 'in the exponent' the shares, $R_j$. For instance, the value, R, may be computed by the at least two parties and/or it may be computed centrally. In the present context the term 'interpolating in the exponent' should be interpreted to mean polynomial interpolation where a secret 'in the exponent' is reconstructed, for instance calculating $g^k$ from $g^{k\_1}$, $g^{k\_2}$, etc.

**[0043]** Finally, each of the at least two parties checks that R is correct, based on the shares, $R_j$, which were received from the other parties.

**[0044]** According to one embodiment of the method, the shares may be Shamir shares, and there are at least three parties (i.e., $n\geq 3$) and the security threshold t satisfies $t<n/2$. In this embodiment, each of the at least three parties may verify that R is correct by checking that each of the shares, $R_j$, received from the other parties is consistent with a degree t polynomial, f, that is uniquely defined by the first t+1 shares. This can be done, e.g., by comparing each of the shares $R_{t+2}$, $R_{t+3}$, ..., $R_n$ to the expected share which can be computed from the first t+1 shares $R_1$, $R_2$, ..., $R_{t+1}$ using e.g. Lagrange interpolation in the exponent. If in all cases the received share equals the expected share, then it can be concluded that all received shares, $R_j$, are correct. This can be concluded since it is known that at least t+1 of the shares, $R_j$, are received from honest parties and hence are correct. If any party finds that some of the shares, $R_j$, are missing or are inconsistent with the polynomial determined by the first t+1 received shares, then the party may abort the protocol.

**[0045]** According to an alternative embodiment of the method, the shares may be additive shares, and the security threshold may be up to t=n-1. In this embodiment the share $g^{k\_i}$ of each party $P_i$ may include a zero knowledge proof, e.g., a non-interactive zero-knowledge proof, whereby $P_i$ proves to the other parties that the share $R_i$ was correctly computed, without revealing any informa-

tion about $k_i$. In this alternative embodiment, verifying correctness of R may include that a party that receives a share $R_j$ from another party verifies the zero knowledge proof and aborts if the proof is invalid.

**[0046]** According to one embodiment, the correctness of R may be ensured in the following manner. Assume that there are three participating parties, n=3, and that one compromised or unavailable party can be tolerated, i.e. t=1. In this case a correct value of R can be computed from shares, $R_j$, originating from two or more honest parties. Each of the three participating parties may then computes its share of R, based on their respective shares of k, and distribute the computed share of R to each of the other two parties, as described above. Each party will then be in the possession of three shares of R, i.e. the share which was computed by itself and the two shares received from the other two participating parties. If all three parties are honest, then R can be correctly computed from any combination of two of these shares. Accordingly, the parties compute R based on any possible combination of two of the available shares, in this example amounting to three combinations, i.e. $(R_1,R_2)$, $(R_1,R_3)$, and $(R_2,R_3)$. If all three combinations result in the same value of R, then it can be concluded that all three parties are honest, and that R is correct. If the value of R computed based on at least one of the combinations differs from the value of R computed based on any of the other combinations, it can be concluded that at least one of the parties is dishonest. However, it can not be determined which of the parties is dishonest, and therefore it is not possible to decide which of the computed values of R is correct. In this case it is simply determined that R is incorrect, and the signing process may be aborted. This might delay the signing process, but no secrets are revealed.

**[0047]** The step of computing the value, R, and the step of checking that R is correct may be performed in a reversed order, i.e. the parties may check that the received shares, $R_j$, are correct before computing the value, R. In this case the value, R, may be computed only if it is found to be correct.

**[0048]** The step of computing a value, R, and the step of computing an authenticator, W, may be performed using the same protocol. For instance, the step of computing an authenticator, W, may be performed essentially in the manner described above relating to the value, R, but by using R as a base instead of g. Thus, in this case each of the at least two parties computes a share, $W_j$, of the authenticator, W, as $W_j=R^{a\_j}$, and distributes the share to each of the other parties, the authenticator, W, is computed from the shares, $W_j$, and each of the parties checks that W is correct, based on the shares, $W_j$, received from the other parties, e.g. in the manner described above.

**[0049]** The method may further comprise the step of aborting the signing process in the case that it is revealed that R or W is incorrect. Thereby it is ensured that the secret key is not revealed to a malicious party. However, in the case that the signing process is aborted, it can be proved that it is safe to restart the process in order to try again to obtain a valid signature.

**[0050]** The method may further comprise the step of aborting the signing process in the case that the step of verifying w reveals that $g^w \neq W$. If it turns out that $g^w \neq W$, then it can be concluded that $w \neq ak$, and consequently a valid signature can not be obtained based on [w]. Attempting to compute and reveal a signature value, s, in this case could potentially harm confidentiality by revealing information about the secret key, x, to a malicious party. Therefore, if this is the case, then the signing process is aborted, similarly to the situation described above, and the signing process may be restarted.

**[0051]** The step of signing a message, M, may be performed by opening [w] and computing $[s]=mw^{-1}[a]+rw^{-1}[a][x]+[d]+m[e]$, where r=F(R), m=H(M), and [d] and [e] are random sharings of zero (so-called "blinder sharings").

**[0052]** With some kinds of secret sharing schemes, once a product [ax]=[a][x] has been computed, it is not safe to open up the sharing [ax], since the shares of [ax] may reveal too much information about the secrets a and x.

**[0053]** This is e.g. the case in a first embodiment of our method, where the shares are Shamir shares, and where the parties' shares of [ax] are obtained by each party multiplying its shares of [a] and [x]. In this case the shares of [ax] are no longer random and may leak information about a or x when opening [ax]. To avoid this, the parties first compute a sharing of zero [0] that is known to have random shares, and instead open [ax]+[0]. Adding [0] does not change the result, but ensures that the value ax is the only information revealed when opening [ax].

**[0054]** So [s] is first computed as $mw^{-1}[a]+rw^{-1}[a][x]$. Note that since w is known to be correct at this point, and assuming that [a][x] is correct, this implies that $[s]=[k^{-1}(m+rx)]$, i.e., [s] is then a sharing of the correct signature value s as defined by DSA or ECDSA. But before opening up [s], a zero sharing [d] is added, i.e., [s]+[d] is opened. Adding d to s does not change the value of s, since d=0.

**[0055]** Given that correctness of w is enforced, as explained above, it can in some embodiments be mathematically proved that revealing the signature (r, s) as computed above does not leak anything more about the secret key x than a correctly computed signature, even in the case where the multiplication [a][x] is a weak multiplication that may not be correct. The only thing that happens if [a][x] is incorrect is that the resulting value (r, s) will not be a valid signature. This can be determined by each party by simply verifying the signature (r, s) using the message M and the public verification key y. If invalid, the parties may restart the signing process.

**[0056]** In some embodiments, opening [s]+[d] can be proved to be secure, but the proof assumes that the honest parties initially agree on the message M to sign. If they disagree on M, unintended information about the secret key x may leak. Therefore, in some embodiments,

such as a first embodiment described herein, the parties may instead compute and open [s]+[d]+m·[e] where both [d] and [e] are random sharings of zero, and where m=H(M). In these embodiments, opening this sharing can be shown to be secure even if the honest parties do not all use the same value M. If they agree on M, [d]+m·[e] will turn into a zero sharing that ensures that the result [s] can be opened without revealing information about x. Conversely, if some of the parties hold different values of M, then adding [d]+m·[e] to [s] turns [s] into a completely random sharing which makes the protocol abort, but without leaking any information about x. Thereby we achieve the property that the parties do not have to first ensure that they use the same value of M before they open [s]. This may be important in practice, since this means that the parties do not have to spend an additional round of interaction to ensure that they agree on the value of M before they open [s].

**[0057]** In an alternative embodiment of the method, the shares may be additive shares and an additive homomorphic encryption scheme, such as Paillier's encryption scheme, is used to implement the multiplication [ax]=[a][x]. Here it may be necessary in order to be able to securely reveal the shares of a product [ax]=[a][x] without leaking information about a or x, to include zero-knowledge proofs, e.g., non-interactive zero-knowledge proofs, in the process of computing [ax]. Including such zero-knowledge proofs in the process, e.g., when Paillier encryption is used, is a well-known technique that can be applied by persons skilled in the art.

**[0058]** At least the steps of generating a secret key, x, generating random secret sharings, [a] and [k], computing a value, R, and computing an authenticator, W, may be performed by pre-processing, prior to generation of the message, M. According to this embodiment, some of the steps, e.g. including generating a key pair ([x], y), computing R and W, ensuring correctness of R or W, generating zero sharings [d] and [e], computing the sharing [w], opening [w], verifying that $g^w$=W, and/or computing [ax]=[a][x] may be performed before the message, M, to be signed is known to the parties. As described above, this reduces the number of steps to be performed, and thereby the processing requirements and response time, at the time where the digital signature is provided.

**[0059]** Many kinds of secret sharing, such as e.g. Shamir secret sharing, has the following property: Given two sharings, e.g. [a] and [x], a sharing of the sum of the two secrets, [a+x], can be computed without any interaction between the parties. Furthermore, if all parties know a given public value c, the parties can compute a sharing of the product [cx] without interaction. Such secret sharing schemes are called 'linear' secret sharing schemes.

**[0060]** According to some embodiments of the present invention where such linear secret sharing is used, by generating [x], [a], y, R, W, [ax] and w, and by verifying correctness of R and w, in a pre-processing stage, prior to the knowledge of the message M to be signed, then once M becomes known to the parties, the parties can generate r and [s] without any interaction with each other. This is because by using the sharings and values already generated and verified in the pre-processing, r can be computed locally by each party as r=F(R) and furthermore, [s] can be computed only by addition of sharings and multiplication of sharings with public constants.

**[0061]** Furthermore, according to some embodiments, by performing some steps by pre-processing, once M is known to the parties, the final signature (r, s) can be revealed to a receiving party in a single round of interaction where each party sends R and its share of [s] to the receiving party.

**[0062]** Furthermore, according to some embodiments, by performing some steps by pre-processing, once M becomes known to the parties, the final signature (r, s) can be revealed to a receiving party in a single round of interaction where only some of the parties send their share of [s] to the receiving party. For example, in a first embodiment, [s] is a degree 2t Shamir sharing, and hence it is sufficient that 2t+1 parties send their shares to the receiving party in a single round of interaction.

**[0063]** According to some embodiments, where linear secret sharing is used, by performing additional pre-processing steps, given M, the final signature (r, s) can be revealed to a receiving party in a single round of interaction where only t+1 of the parties send R and their share of [s] to the receiving party. For example, in some embodiments where [ax] is a degree 2t Shamir sharing, the degree of [ax] can be reduced to t in a pre-processing step using standard techniques known from the field of secure multiparty computation. By doing this, the degree of [s] also becomes t, and hence s can be reconstructed by the receiving party based only on t+1 shares of [s].

**[0064]** Furthermore, according to an embodiment of the invention, additional pre-processing steps may be performed to ensure that the sharing [ax] is a correct sharing of the value ax. This can be done with standard techniques from the field of secure multiparty computation. By doing this, the property is achieved, that if the process does not abort during the pre-processing steps, then a certain number of honest parties, e.g., 2t+1 or t+1 honest parties, can be guaranteed to be able to open up a valid signature (r, s), even if up to t malicious parties try to prevent the opening of a valid signature.

**[0065]** Simplifying the communication like this, e.g., by minimising the number of interaction rounds, that are required once the message M to be signed is known to the parties, as well as the number of shares needed to reconstruct s, can provide benefits in practical applications of a multiparty signature scheme.

**[0066]** The method may further comprise the step of computing a public key, y, as $y=g^x$, and revealing y to each of the at least two parties. According to this embodiment, a key pair in the form of a secret key, [x], and a public key, $y=g^x$, is generated. This key pair is used for generating the digital signature.

**[0067]** The method may further comprise the step of

verifying the signature, using the public key, y. This could, e.g., comprise checking whether or not $r=F(g^{m/s}\cdot y^{r/s})$. Alternatively or additionally, this step may comprise checking whether or not $R^s=g^m\cdot y^r$.

**[0068]** This could, e.g. include each of the at least two parties verifying the correctness of the signature, using the public verification key y. This could, e.g., comprise checking that $R^s=g^m\cdot y^r$. If a party finds this to be the case, the party accepts the signature (r, s) and outputs it as the result. If not, it may abort the signing process.

**[0069]** In another embodiment of the method, it is not the parties themselves, but an external party that should receive the resulting signature (r, s). In this case each party $P_j$ computes R and its share $s_j$ of the secret sharing [s] as described above. Each party then sends R and $s_j$ to the external party. The external party compares all the received values R and aborts if they are not equal. It then computes r=F(R) and computes s from the shares $s_j$ via polynomial interpolation, and verifies using the public key y, that $R^s=g^m\cdot y^r$. If so, he accepts (r, s) as the resulting signature.

**[0070]** The method may further comprise the step of checking correctness of y. According to an embodiment of the invention, this step may comprise the steps of first generating a random sharing [x] and then opening $y=g^x$ and checking correctness of y in the same manner as generating the random sharing [k] and opening and verifying correctness of $R=g^k$.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0071]** The invention will now be described in further detail with reference to the accompanying drawings in which

Fig. 1 is a block diagram illustrating key generation and signature generation using a method according to a first embodiment of the invention,

Fig. 2 is a block diagram illustrating key generation using a method according to a second embodiment of the invention,

Fig. 3 is a block diagram illustrating signature generation using a method according to a third embodiment of the invention,

Fig. 4 is a block diagram illustrating signature generation using a method according to a fourth embodiment of the invention,

Fig. 5 is a block diagram illustrating key generation and signature generation using a method according to a fifth embodiment of the invention, and

Fig. 6 is a flow chart illustrating a method according to an embodiment of the invention.

## DETAILED DESCRIPTION OF THE DRAWINGS

**[0072]** Fig. 1 is a block diagram illustrating key generation and signature generation using a method according to a first embodiment of the invention. The method involves the use of three parties, P1, P2 and P3, being individual or separate in the sense that information being available to one of the parties P1, P2, P3 may not be available to the other parties P1, P2, P3. The parties P1, P2, P3 may, e.g., be in the form of physically separated hardware servers. In this example the security threshold (t) is 1, i.e., if one of the parties P1, P2, P3 is malicious, that party will not be able to learn any information about x or otherwise be able to sign a message M with the secret key x unless the other (honest) parties agree to sign M.

**[0073]** A client 1 sends a request, KeyGen, to each of the parties P1, P2, P3, requesting that the parties P1, P2, P3 generate a key pair, ([x], y), comprising a secret key, [x], and a public key, y. The client 1 is arranged in the environment surrounding a system including the three parties P1, P2, P3, i.e. the client does not form part of the system which is to generate the key and the signature.

**[0074]** In response to receipt of the request KeyGen, the three parties P1, P2, P3 generate a secret key, [x], as a random secret sharing among the three parties P1, P2, P3. This may include several rounds of interaction between the parties P1, P2, P3, and it may, e.g., be performed in the manner described below with reference to Fig. 2. As a result, each of the parties P1, P2, P3 holds a share, x1, x2, x3 of the secret key x, but none of the parties P1, P2, P3 is in the possession of the entire secret key x.

**[0075]** The three parties P1, P2, P3 further compute a public key, y, as $y=g^x$. The public key, y, is made public in the sense that each of the parties P1, P2, P3 is in the possession of y, and in the sense that y is communicated to the client 1 by each of the parties P1, P2, P3. Thus, $y=g^x$ is made public or 'opened', but x remains secret.

**[0076]** In the case that none of the parties P1, P2, P3 is malicious or corrupted, the public key, y, communicated to the client 1 by each of the parties P1, P2, P3 will be the same, i.e. the client 1 will receive three identical version of y from the three parties P1, P2, P3. Thus, if the client 1 receives three identical versions of y, it can conclude that none of the parties P1, P2, P3 is malicious or corrupted, i.e. that all of the parties P1, P2, P3 have acted correctly so far. On the other hand, in the case that the three versions of y received from the three parties P1, P2, P3 differ from each other, it can be concluded that at least one of the parties P1, P2 or P3 is malicious or corrupted. In that case the client 1 causes the process to abort.

**[0077]** When the key pair, ([x], y), has been generated, a signature process, which applies the generated key pair, ([x], y), is initiated by the client 1 sending a request, Sign, and a message, M, to be signed to each of the

parties, P1, P2, P3.

**[0078]** In response to receipt of the request, Sign, and the message, M, the parties P1, P2, P3 engage in a signature generation process which may require several rounds of interaction among the parties and in which each party P1, P2, P3 applies its share x1, x2, x3 of [x] without revealing the share x1, x2, x3 to the other parties. At the end of the signature generation process, each party P1, P2, P3 is in the possession of a value R and a share, s1, s2, s3 of [s]. The signature generation process could, e.g., be performed in the manner described below with reference to Fig. 3.

**[0079]** Each of the parties P1, P2, P3 then returns R and its share, s1, s2 or s3, of [s] to the client 1. In response thereto the client 1 computes s from the received shares s1, s2, s3, e.g. using interpolation. Furthermore, the client 1 computes r=F(R), and may accept (r, s) as a valid signature only if identical versions of R are received from at least two of the three parties P1, P2, P3. Furthermore, an additional verification check, e.g. validating that $R^s=g^m \cdot y^r$, may be performed by the client 1, in order for it to accept the signature, (r, s) as a valid signature.

**[0080]** Fig. 2 is a block diagram illustrating key generation using a method according to a second embodiment of the invention. The key generation illustrated in Fig. 2 may, e.g., be applied as part of the method illustrated in Fig. 1.

**[0081]** In the embodiment illustrated in Fig. 2, three parties P1, P2, P3 cooperate in computing a key pair ([x], y), where [x] is a secret key in the form of a secret sharing among the three parties P1, P2, P3, and y is a public key. According to this embodiment the shares are Shamir secret shares, and the security threshold is one, i.e., t=1.

**[0082]** In a first round of interactions between the parties P1, P2, P3, the secret key, [x], is generated as a random degree t Shamir secret sharing among the parties P1, P2, P3. To this end, each party P1, P2, P3 generates three random values, one for itself and one for each of the other parties P1, P2, P3, and forwards the generated values to the respective other parties P1, P2, P3. Thus, party P1 generates value x1,1 and keeps it for itself, generates value x1,2 and forwards it to party P2, and generates value x1,3 and forwards it to party P3. Similarly, party P2 generates value x2,1 and forwards it to party P1, generates value x2,2 and keeps it for itself, and generates value x2,3 and forwards it to party P3. Finally, party P3 generates value x3,1 and forwards it to party P1, generates value x3,2 and forwards it to party P2, and generates value x3,3 and keeps it for itself.

**[0083]** Thus, at the end of the first round of interaction among the parties P1, P2, P3, each party P1, P2, P3 is in the possession of three random values, i.e. a value generated by itself and a value received from each of the other parties P1, P2, P3. Based on these three values, each of the parties P1, P2, P3 generates a share, x1, x2, x3, of [x].

**[0084]** In a second round of interaction among the parties P1, P2, P3, the parties P1, P2, P3 compute a public key, y. To this end, each party computes $y_i=g^{x_i}$, where g is a generator for a cyclic group, G, $y_i$ is a value generated by party Pi, and $x_i$ is the share of [x] being held by party Pi. Furthermore, each of the parties P1, P2, P3 communicates the value $y_i$ to each of the other parties P1, P2, P3. Thus, party P1 computes $y_1=g^{x_1}$ and communicates $y_1$ to parties P2 and P3, etc. Accordingly, each of the parties P1, P2, P3 is now in the possession of each of the three values, $y_1$, $y_2$ and $y_3$.

**[0085]** Each of the parties P1, P2, P3 then checks that the values $y_i$ received from the other two parties are trustworthy. This may, e.g., include performing interpolation in the exponent. In the case that one of the parties P1, P2, P3 concludes that the value $y_i$ received from at least one of the other parties is not trustworthy, that party P1, P2, P3 outputs an 'abort' signal, and the signing process is consequently aborted.

**[0086]** In the case that none of the parties P1, P2, P3 outputs an 'abort' as described above, the signing process is allowed to continue, and each of the parties P1, P2, P3 generates a public key, y, based on the values $y_1$, $y_2$ and $y_3$, and using interpolation. According to the DSA/ECDSA standard, y=1 is an illegal public key, and hence the process should be aborted if this is the case. It is noted that if the protocol aborts, it may be restarted, resulting in another key pair being generated. If it was found that $y \neq 1$, each of the parties P1, P2, P3 then forwards an 'OK' signal to each of the other parties P1, P2, P3, as part of a third round of interaction among the parties P1, P2, P3. Each of the parties P1, P2, P3 accepts its own version of y only if it receives an 'OK' signal from each of the other parties P1, P2, P3. Otherwise the signing process will be aborted.

**[0087]** In the case that 'OK' signals are received from the other parties P1, P2, P3 as described above, the public key, y, is output.

**[0088]** Fig. 3 is a block diagram illustrating signature generation using a method according to a third embodiment of the invention. As in Fig. 2, the secret sharings are Shamir sharings and the security threshold is one (t=1). The process starts in Fig. 3a and continues in Fig. 3b. The signature generation illustrated in Fig. 3 may, e.g., be applied as a part of the method illustrated in Fig. 1.

**[0089]** In the embodiment illustrated in Fig. 3, three parties P1, P2, P3 cooperate in generating a signature (r, s) for a message, M, using a secret key, [x], in the form of a secret sharing among the three parties, P1, P2, P3. The secret key, [x], could, e.g., be generated as a degree t Shamir sharing in the manner described above with reference to Fig. 2.

**[0090]** In a first round of interaction among the parties P1, P2, P3, illustrated in Fig. 3a, the parties generate random degree t secret sharings, [k] and [a]. This is performed essentially in the manner described above with reference to Fig. 2 with regard to the generation of [x]. The parties also generate three random degree 2t zero sharings, [b], [d], and [e], i.e. blinding sharings. Thus, at the end of the first round of interaction among the parties

P1, P2, P3, each party P1, P2, P3 holds a share of each of [k], [a], [b], [d] and [e], and none of the parties P1, P2, P3 is in the possession of any information about the secrets a and k.

**[0091]** Next, in a second round of interaction among the parties P1, P2, P3, also illustrated in Fig. 3a, each of the parties P1, P2, P3 computes a value Ri as $Ri=g^{ki}$, where g is a generator for a cyclic group, G, Ri is the value computed by party Pi, and ki is the share of [k] being held by party Pi. Thus, party P1 computes $R1=g^{k1}$, party P2 computes $R2=g^{k2}$, and party P3 computes $R3=g^{k3}$.

**[0092]** Furthermore, each of the parties P1, P2, P3 computes a value wi as $wi=ki\cdot ai+bi$, where wi is the value computed by party Pi, and ki, ai and bi are the shares of [k], [a] and [b], respectively, being held by party Pi. Thus, party P1 computes $w1=k1\cdot a1+b1$, party P2 computes $w2=k2\cdot a2+b2$, and party P3 computes $w3=k3\cdot a3+b3$. The shares w1, w2, w3 held by the parties form a degree 2t Shamir sharing [w] where w equals ak if all parties performed the prescribed actions correctly.

**[0093]** Each of the parties then reveals the computed values Ri and wi to each of the other parties. Thus, each of the parties P1, P2, P3 is now in the possession of each of the three values R1, R2 and R3, and each of the three shares w1, w2 and w3, and accordingly $R=g^k$ and $w=k\cdot a+b$ are now known by each of the parties P1, P2, P3, even though each of k and a remains secret. This may be referred to as 'opening' R and w. The sharing [b] may be referred to as a 'blinder sharing', since adding it to [ak] does not change the secret, but 'blinds' the individual shares of the sharing [ak], thereby making it impossible to derive any information about a or k from seeing the shares of [ak] except the product ak.

**[0094]** At the end of the second round of interaction among the parties P1, P2, P3, each of the parties P1, P2, P3 checks correctness of R. This is done based on the received values R1, R2 and R3, and using interpolation in the exponent. In the case that at least one of the parties P1, P2, P3 finds that R is incorrect, the process is aborted. Otherwise, the process continues.

**[0095]** Next, in a third round of interaction among the parties P1, P2, P3, illustrated in Fig. 3b, each of the parties P1, P2, P3 computes a value, Wi as $Wi=R^{ai}$, where Wi is the value computed by party Pi, and ai is the share of [a] being held by party Pi. Furthermore, each party P1, P2, P3 reveals the computed values Wi to each of the other parties P1, P2, P3. Accordingly, each of the parties P1, P2, P3 is now in the possession of each of the values W1, W2, W3, and by using interpolation in the exponent $W=R^a$ can be computed by each of the parties P1, P2, P3, while the value a remains secret. This may be referred to as 'opening' W.

**[0096]** At the end of the third round of interaction among the parties P1, P2, P3, each of the parties P1, P2, P3 checks correctness of W, based on the received values W1, W2, W3, and using interpolation in the exponent. This is done using the values Wi in the same manner as correctness of R was checked using the values Ri in the second round of interaction. In the case that at least one of the parties P1, P2, P3 finds that W is incorrect, the process is aborted. Otherwise, the process is continued.

**[0097]** Finally, in a fourth round of interaction among the parties P1, P2, P3, each of the parties P1, P2, P3 computes w, based on the values w1, w2 and w3, and using interpolation.

**[0098]** Furthermore, each of the parties P1, P2, P3 verifies w by checking that $W=g^w$. As described above, it is enforced that $W=R^a$ and $R=g^k$. If this was not the case, the process would have aborted previously. It follows from this that $W=g^{ka}$. So by checking that $g^w=W$, it is ensured that $w=a\cdot k$. It is noted that [w] was computed, not as [a][k], but as [a][k]+[b], but since [b] is a sharing of zero, i.e. [b]=[0], adding [b] makes no difference for this check.

**[0099]** In the case that at least one of the parties P1, P2, P3 finds that $W\neq g^w$, then the signature process is aborted. Otherwise the process is continued, and each of the parties P1, P2, P3 computes a share si of a sharing [s] as:

$$si=m\cdot hi+r\cdot hi\cdot xi+m\cdot di+ei$$

where:

$hi=ai\cdot w^{-1}$,
r=F(R), where F is a predefined function, and
m=H(M), where M is the message to be signed and H is a predefined function.

xi, di and ei are the shares of [x], [d] and [e], respectively, being held by party Pi. When each party computes its share si as described, it means that the parties collectively perform the computation $[s]=m[k^{-1}]+r[k^{-1}][x]+m[d]+[e]$. If all parties performs the actions as prescribed, this results in $[s]=[k^{-1}(m+rx)]$, i.e., $s=k^{-1}(m+rx)$ as required by DSA or ECDSA.

**[0100]** Each of the parties P1, P2, P3 reveals the share si to each of the other parties P1, P2, P3, and each of the parties P1, P2, P3 computes s using interpolation based on the received shares s1, s2, and s3 and then checks correctness of s by verifying the signature on the message M using the public key y, i.e., by checking that $R=g^{m}\cdot y^{r}$. If correctness of s is confirmed, the signature (r, s) is accepted as the final signature of the message, M.

**[0101]** Fig. 4 is a block diagram illustrating signature generation using a method according to a fourth embodiment of the invention. The process illustrated in Fig. 4 includes six rounds of interaction among three parties P1, P2, P3. The first three rounds of interaction among the parties P1, P2, P3 are identical to the first three rounds of interaction illustrated in Fig. 3 and described above.

**[0102]** In a fourth round of interaction among the parties P1, P2, P3, only party P1 computes s1 in the manner

described above with reference to Fig. 3, i.e.:

$$s1 = m \cdot h1 + r \cdot h1 \cdot x1 + m \cdot d1 + e1.$$

**[0103]** P1 then forwards s1 to party P2, and in a fifth round of interaction among the parties P1, P2, P3, party P2 computes s2 in the manner described above with reference to Fig. 3, and forwards s1 and s2 to party P3.

**[0104]** In a sixth round of interaction among the parties P1, P2, P3, party P3 computes s3 in the manner described above with reference to Fig. 3, and computes s based on the three shares s1, s2 and s3, and using interpolation. Party P3 then checks correctness of s, and if s is correct, the signature (r, s) is accepted by party P3 as the resulting signature (r, s) of the message M.

**[0105]** Thus, in the embodiment illustrated in Fig. 4, the signature (r, s) is only received by party P3, whereas each of the parties P1, P2, P3 receives the signature (r, s) in the embodiment illustrated in Fig. 3. An embodiment like this may be practical, since in the online phase, each party P1, P2, P3 needs only send a single message to one party, whereas in the embodiment of Fig. 3, each party has to send a message to each of the other parties.

**[0106]** Fig. 5 is a block diagram illustrating key generation and signature generation using a method according to a fifth embodiment of the invention. The embodiment illustrated in Fig. 5 is very similar to the embodiment illustrated in Fig. 1, and it will therefore not be described in detail here.

**[0107]** However, in the embodiment illustrated in Fig. 5, the process is initiated by one of the parties, party P1, rather than by an external client. Thus, party P1 performs the steps which are performed by the client in the embodiment of Fig. 1, as well as the steps performed by party P1 in the embodiment of Fig. 1.

**[0108]** Fig. 6 is a flow chart illustrating a method according to an embodiment of the invention. The process is started in step 2. In step 3, a cyclic group, G, and a generator, g, for the cyclic group, G, are defined. Furthermore, functions F and H are defined. G, g, F and H are all specified by a digital signature algorithm (DSA) or an elliptic curve digital signature algorithm (ECDSA) which is to be used for generating the digital signature.

**[0109]** In step 4 a random secret sharing [x] is generated among at least two parties, where x is a secret key.

**[0110]** In step 5 random secret sharings [a] and [k] are generated among the at least two parties.

**[0111]** In step 6 the parties compute $[w]=[a]\cdot[k]$, and in step 7 the parties compute a value, $R=g^k$. R may, e.g., be computed by each of the parties computing a share, $R_j$, as $R_j=g^{k\_j}$, where k_j is the share of [k] being held by party j, and computing the value, R, from the shares, $R_j$.

**[0112]** In step 8 it is investigated whether or not R is correct. If this is not the case, the process is forwarded to step 9, where the signing process is aborted, and the process is returned to step 5 in order to generate new secret sharings [a] and [k].

**[0113]** In the case that step 8 reveals that R is correct, the signing process is allowed to continue, and the process is forwarded to step 10, where the parties compute an authenticator, $W=g^{ak}$. This is done by computing $R^a$.

**[0114]** In step 11 it is investigated whether or not W is correct. If this is not the case the process is forwarded to step 9, where the signing process is aborted, and the process is returned to step 5 in order to generate new secret sharings [a] and [k].

**[0115]** In the case that step 11 reveals that W is correct, the process is forwarded to step 12, where it is investigated whether or not $g^w=W$. Since $W=R^a$ and $R=g^k$, then $W=g^{ak}$. Since $w=a\cdot k$, $g^w=W$ if w has been correctly computed. Thus, if it can be verified that $g^w=W$ it can be concluded that w has been correctly computed. Thus, in the case that step 12 reveals that $g^w\neq W$, the process is forwarded to step 9, where the signing process is aborted, and the process returns to step 5 in order to generate new secret sharings [a] and [k].

**[0116]** In the case that it is verified in step 12 that $g^w=W$, the process is forwarded to step 13, where a sharing [s] is generated among the parties, and a signature is applied to the message. Finally, the process is ended in step 14.

## Claims

1. A method for providing a digital signature to a message, M, in accordance with a digital signature algorithm, DSA, or an elliptic curve digital signature algorithm, ECDSA, the method comprising the steps of:

    - providing a generator, g, for a cyclic group, G, of order q, where $g \in G$, a function, F, and a function, H, where g, G, F and H are specified by the DSA or ECDSA,
    - generating a secret key, x, as a random secret sharing [x] among at least two parties,
    - generating random secret sharings, [a] and [k], among the at least two parties and computing $[w]=[a][k]$,
    - computing a value, R, as $R=g^k$, without revealing k, by performing the steps of:

        - each of the at least two parties computing a share, $R_j$, of the value, R, as $R_j=g^{k\_j}$, and distributing the share to each of the other parties, and
        - computing the value, R, from the shares, $R_j$,

    - ensuring that R is correct by verifying that $R=g^k$ is computed from at least t+1 shares of [k] originating from honest parties, by each of the parties checking that R is correct, based on the shares, $R_j$, received from the other parties,

- computing an authenticator, W, as $W=g^{ak}$, by computing $R^a$, without revealing a or k, by performing the steps of:

  - each of the at least two parties computing a share, $W_j$, of the authenticator, W, as $W_j=R^{a\_j}$, and distributing the share to each of the other parties, and
  - computing the authenticator, W, from the shares, $W_j$,

- ensuring that W is correct by verifying that $W=R^a$ is computed from at least t+1 shares of [a] originating from honest parties, by each of the parties checking that W is correct, based on the shares, $W_j$, received from the other parties,
- verifying [w] by checking whether or not $g^w=W$, and
- signing the message, M, by computing $[k^{-1}]=[a]\cdot w^{-1}$, computing $[x\cdot k^{-1}]=[x]\cdot[k^{-1}]$, and generating a sharing, [s], among the at least two parties, as a function of M, R, $[k^{-1}]$ and $[x\cdot k^{-1}]$, by computing $[s]=m\cdot w^{-1}\cdot[a] + r\cdot w^{-1}\cdot[a]\cdot[x] + [d]$, where r=F(R), m=H(M), and [d] is a random sharing of zero, where s forms part of a signature pair (r, s).

**2.** A method according to claim 1, further comprising the step of aborting the signing process in the case that it is revealed that R or W is incorrect.

**3.** A method according to any of the preceding claims, further comprising the step of aborting the signing process in the case that the step of verifying [w] reveals that $g^w \neq W$.

**4.** A method according to any of the preceding claims, wherein the step of signing a message, M, is performed by computing $[s]=m\cdot w^{-1}\cdot[a] + r\cdot w^{-1}\cdot[a]\cdot[x] + [d] + m\cdot[e]$, where r=F(R), m=H(M), and [d] and [e] are random sharings of zero.

**5.** A method according to any of the preceding claims, wherein at least the steps of generating a secret key, x, generating random secret sharings, [a] and [k], computing a value, R, and computing an authenticator, W, are performed by pre-processing, prior to generation of the message, M.

**6.** A method according to any of the preceding claims, further comprising the step of computing a public key, y, as $y=g^x$, and revealing y to each of the at least two parties.

**7.** A method according to claim 6, further comprising the step of verifying the signature, using the public key, y.

**8.** A method according to claim 7, wherein the step of verifying the signature comprises checking whether or not $r=F(g^{m/s}\cdot y^{r/s})$.

**9.** A method according to claim 7 or 8, wherein the step of verifying the signature comprises checking whether or not $R^s=g^m\cdot y^r$.

**10.** A method according to any of claims 6-9, further comprising the step of checking correctness of y.

**Patentansprüche**

**1.** Verfahren zum Bereitstellen einer digitalen Signatur für eine Nachricht, M, gemäß einem digitalen Signaturalgorithmus, DSA, oder einem digitalen Signaturalgorithmus mit elliptischer Kurve, ECDSA, wobei das Verfahren folgende Schritte umfasst:

- Bereitstellen eines Generators, g, für eine zyklische Gruppe, G, q. Ordnung, wobei $g \in G$, einer Funktion, F, und einer Funktion, H, wobei g, G, F und H durch den DAS oder den ECDSA vorgegeben werden,
- Generieren eines geheimen Schlüssels, x, als eine zufällige Geheimnisteilung [x] unter mindestens zwei Parteien,
- Generieren von zufälligen Geheimnisteilungen, [a] und [k], unter den mindestens zwei Parteien, und Berechnen von [w] = [a] [k],
- Berechnen eines Wertes, R, als $R = g^k$, ohne k preiszugeben, durch Ausführen folgender Schritte:

  - jede der mindestens zwei Parteien berechnet ein Teilgeheimnis, $R_j$, des Wertes, R, als $R_j = g^{k\_j}$, und verteilt das Teilgeheimnis an jede der anderen Parteien, und
  - Berechnen des Wertes, R, aus den Teilgeheimnissen, $R_j$,

- Sicherstellen, dass R richtig ist, durch Überprüfen, dass $R = g^k$ aus mindestens t+1 Teilgeheimnissen von [k] berechnet wird, die von vertrauenswürdigen Parteien stammen, indem jede der Parteien kontrolliert, dass R richtig ist, basierend auf den Teilgeheimnissen, $R_j$, die von den anderen Parteien empfangen werden,
- Berechnen eines Authentifikators, W, als $W = g^{ak}$, durch Berechnen von $R^a$, ohne a oder k preiszugeben, durch Ausführen folgender Schritte:

  - jede der mindestens zwei Parteien berechnet ein Teilgeheimnis, $W_j$, des Authentifikators, W, als $W_j = R^{a\_j}$, und verteilt das Teilgeheimnis an jede der anderen Parteien,

und
- Berechnen des Authentifikators, W, aus den Teilgeheimnissen, $W_j$,

- Sicherstellen, dass W richtig ist, durch Überprüfen, dass $W = R^a$ aus mindestens t+1 Teilgeheimnissen von [a] berechnet wird, die von vertrauenswürdigen Parteien stammen, indem jede der Parteien kontrolliert, dass W richtig ist, basierend auf den Teilgeheimnissen, $W_j$, die von den anderen Parteien empfangen werden,
- Überprüfen von [w] durch Kontrollieren, ob $g^w = W$ oder nicht, und
- Signieren der Nachricht, M, durch Berechnen von $[k^{-1}] = [a] \cdot w^{-1}$, Berechnen von $[x \cdot k^{-1}] = [x] \cdot [k^{-1}]$, und Generieren einer Teilung, [s], unter den mindestens zwei Parteien, als eine Funktion von M, R, $[k^{-1}]$ und $[x \cdot k^{-1}]$, durch Berechnen von $[s] = m \cdot w^{-1} \cdot [a] + r \cdot w^{-1} \cdot [a] \cdot [x] + [d]$, wobei r = F(R), m = H(M), und [d] eine Zufallsteilung von null ist, wobei s Teil eines Signaturpaars (r, s) ist.

**2.** Verfahren nach Anspruch 1, ferner umfassend den Schritt des Abbrechens des Signaturprozesses für den Fall, dass sich ergibt, dass R oder W nicht richtig ist.

**3.** Verfahren nach einem der vorhergehenden Ansprüche, ferner umfassend den Schritt des Abbrechens des Signaturprozesses für den Fall, dass der Schritt des Überprüfens von [w] ergibt, dass $g^w \neq W$.

**4.** Verfahren nach einem der vorhergehenden Ansprüche, wobei der Schritt des Signierens einer Nachricht, M, ausgeführt wird durch das Berechnen von $[s] = m \cdot w^{-1} \cdot [a] + r \cdot w^{-1} \cdot [a] \cdot [x] + [d] + m \cdot [e]$, wobei r = F(R), m = H(M), und [d] und [e] Zufallsteilungen von null sind.

**5.** Verfahren nach einem der vorhergehenden Ansprüche, wobei mindestens die Schritte des Generierens eines geheimen Schlüssels, x, des Generierens von zufälligen Geheimnisteilungen, [a] und [k], des Berechnens eines Wertes, R, und des Berechnens eines Authentifikators, W, durch eine Vorverarbeitung vor der Generierung der Nachricht, M, ausgeführt werden.

**6.** Verfahren nach einem der vorhergehenden Ansprüche, ferner umfassend den Schritt des Berechnens eines öffentlichen Schlüssels, y, als $y = g^x$, und des Preisgebens von y für jede der mindestens zwei Parteien.

**7.** Verfahren nach Anspruch 6, ferner umfassend den Schritt des Überprüfens der Signatur unter Verwendung des öffentlichen Schlüssels, y.

**8.** Verfahren nach Anspruch 7, wobei der Schritt des Überprüfens der Signatur das Kontrollieren umfasst, ob $r = F(g^{m/s} \cdot y^{r/s})$ oder nicht.

**9.** Verfahren nach Anspruch 7 oder 8, wobei der Schritt des Überprüfens der Signatur das Kontrollieren umfasst, ob $R^s = g^m \cdot y^r$ oder nicht.

**10.** Verfahren nach einem der Ansprüche 6 bis 9, ferner umfassend den Schritt des Kontrollierens der Richtigkeit von y.

**Revendications**

**1.** Procédé pour fournir une signature numérique à un message, M, conformément à un algorithme de signature numérique, DSA, ou à un algorithme de signature numérique à courbe elliptique, ECDSA, le procédé comprenant les étapes de :

- fourniture d'un générateur, g, pour un groupe cyclique, G, d'ordre q, où $g \in G$, une fonction, F, et une fonction, H, où g, G, F et H sont spécifiés par le DSA ou l'ECDSA,
- génération d'une clé secrète, x, comme partage secret aléatoire [x] entre au moins deux parties,
- génération de partages secrets aléatoires, [a] et [k], entre les au moins deux parties et calcul de [w]=[a][k],
- calcul d'une valeur, R, comme $R=g^k$, sans révélation de k, en effectuant les étapes de :

  - calcul par chacune des au moins deux parties d'une part, $R_j$, de la valeur, R, comme $R_j=g^{kJ}$, et distribution de la part à chacune des autres parties, et
  - calcul de la valeur, R, à partir des parts, $R_j$,

- fait de s'assurer que R est correct en vérifiant que $R=g^k$ est calculé à partir d'au moins t+1 parts de [k] provenant de parties honnêtes, par chacune des parties vérifiant que R est correct, sur la base des parts, $R_j$, reçues des autres parties,
- calcul d'un authentifiant, W, comme $W=g^{ak}$, en calculant $R^a$, sans révélation de a ou k, en effectuant les étapes de :

  - calcul par chacune des au moins deux parties d'une part, $W_j$, de l'authentifiant, W, comme $W_j=R^{aj}$, et distribution de la part à chacune des autres parties, et
  - calcul de l'authentifiant, W, à partir des parts, $W_j$,

- fait de s'assurer que W est correct en vérifiant que $W=R^a$ est calculé à partir d'au moins t+1

parts de [a] provenant de parties honnêtes, par chacune des parties vérifiant que W est correct, sur la base des parts, $W_j$, reçues des autres parties,

- vérification de [w] en vérifiant si oui ou non $g^w=W$, et
- signature du message, M, en calculant $[k^{-1}] =[a] \cdot w^{-1}$, en calculant $[x \cdot k^{-1}] = [x] \cdot [k^{-1}]$, et en générant un partage, [s], entre les au moins deux parties, en fonction de M, R, $[k^{-1}]$ et $[x \cdot k^{-1}]$, en calculant $[s]=m \cdot w^{-1} \cdot [a] + r \cdot w^{-1} \cdot [a] \cdot [x] + [d]$, où r=F(R), m=H(M), et [d] est un partage aléatoire de zéro, où s fait partie d'une paire de signatures (r, s).

**2.** Procédé selon la revendication 1, comprenant en outre l'étape d'abandon du processus de signature dans le cas où il serait révélé que R ou W est incorrect.

**3.** Procédé selon l'une quelconque des revendications précédentes, comprenant en outre l'étape d'abandon du processus de signature dans le cas où l'étape de vérification de [w] révélerait que $g^w \neq W$.

**4.** Procédé selon l'une quelconque des revendications précédentes, dans lequel l'étape de signature d'un message, M, est réalisée en calculant $[s]=m \cdot w^{-1} \cdot [a] + r \cdot w^{-1} \cdot [a] \cdot [x] + [d] + m \cdot [e]$, où r=F(R), m=H(M), et [d] et [e] sont des partages aléatoires de zéro.

**5.** Procédé selon l'une quelconque des revendications précédentes, dans lequel au moins les étapes de génération d'une clé secrète, x, de génération de partages secrets aléatoires, [a] et [k], de calcul d'une valeur, R, et de calcul d'un authentifiant, W, sont réalisées par pré-traitement, avant la génération du message, M.

**6.** Procédé selon l'une quelconque des revendications précédentes, comprenant en outre l'étape de calcul d'une clé publique, y, comme $y=g^x$, et de révélation de y à chacune des au moins deux parties.

**7.** Procédé selon la revendication 6, comprenant en outre l'étape de vérification de la signature, en utilisant la clé publique, y.

**8.** Procédé selon la revendication 7, dans lequel l'étape de vérification de la signature comprend le fait de vérifier si oui ou non $r=F ( g^{m/s} \cdot y^{r/s} )$.

**9.** Procédé selon la revendication 7 ou 8, dans lequel l'étape de vérification de la signature comprend le fait de vérifier si oui ou non $R^s=g^m \cdot y^r$.

**10.** Procédé selon l'une quelconque des revendications 6 à 9, comprenant en outre l'étape de vérification de l'exactitude de Y.

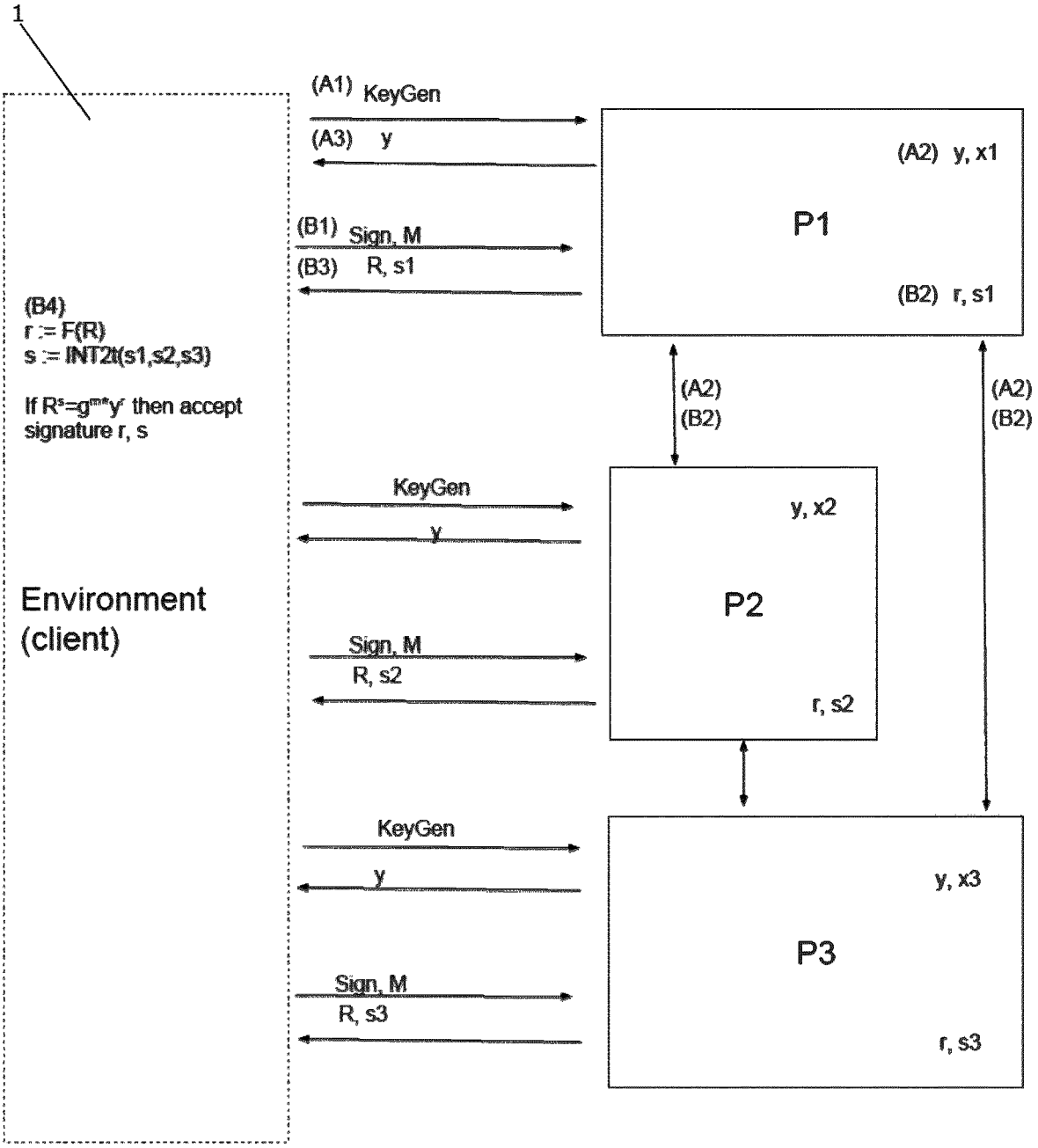
1
(A1) KeyGen
(A3) y
(A2) y, x1
P1
(B1) Sign, M
(B3) R, s1
(B2) r, s1
(B4)
r := F(R)
s := INT2t(s1,s2,s3)
If R^s=g^m*y^r then accept signature r, s
(A2)
(B2)
(A2)
(B2)
KeyGen
y
y, x2
P2
Environment (client)
Sign, M
R, s2
r, s2
KeyGen
y
y, x3
P3
Sign, M
R, s3
r, s3

Fig. 1

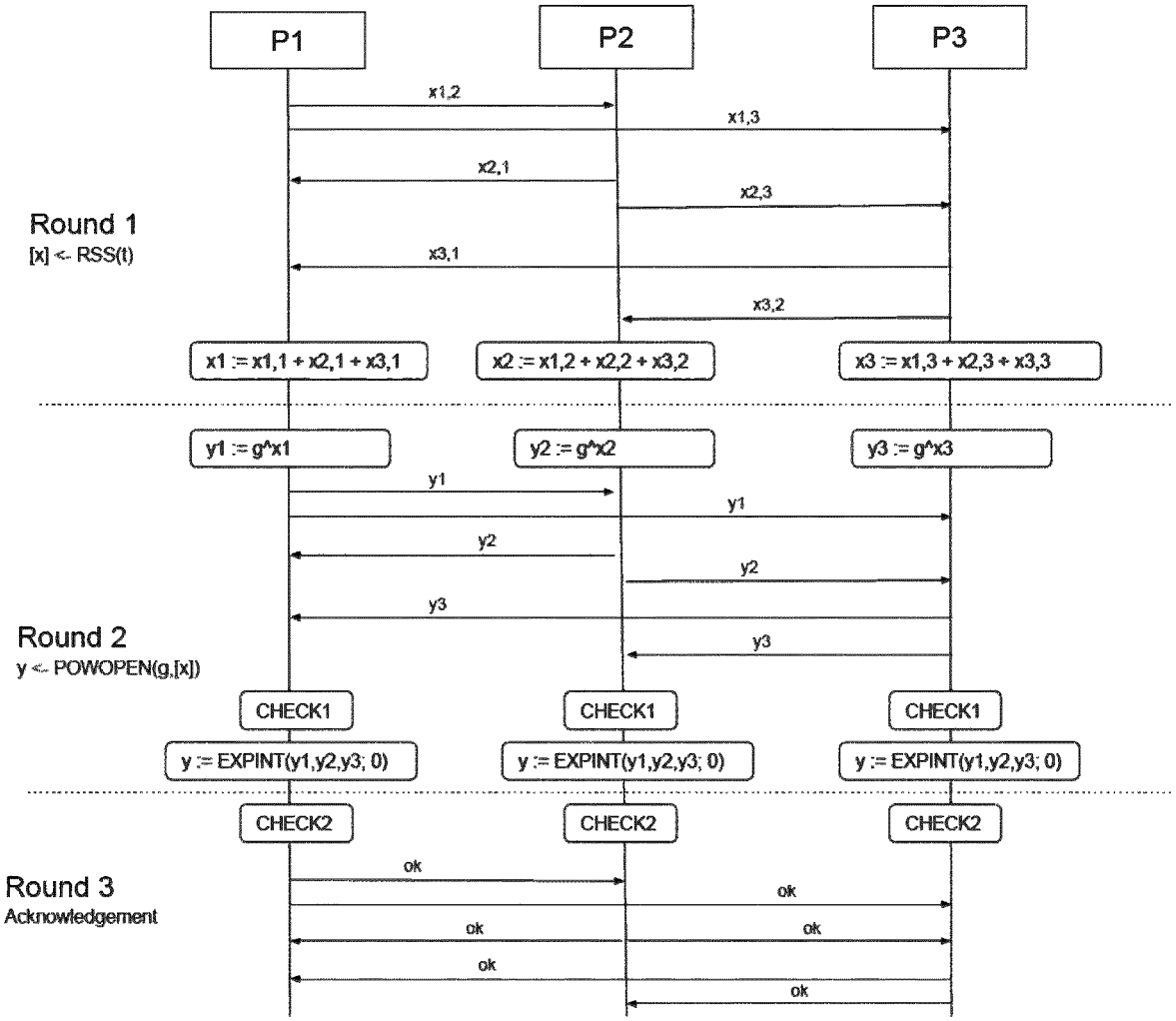
P1
P2
P3
x1,2
x1,3
x2,1
x2,3
Round 1
[x] <- RSS(t)
x3,1
x3,2
x1 := x1,1 + x2,1 + x3,1
x2 := x1,2 + x2,2 + x3,2
x3 := x1,3 + x2,3 + x3,3
y1 := g^x1
y2 := g^x2
y3 := g^x3
y1
y1
y2
y2
y3
Round 2
y <- POWOPEN(g,[x])
y3
CHECK1
CHECK1
CHECK1
y := EXPINT(y1,y2,y3; 0)
y := EXPINT(y1,y2,y3; 0)
y := EXPINT(y1,y2,y3; 0)
CHECK2
CHECK2
CHECK2
ok
ok
Round 3
Acknowledgement
ok
ok
ok
ok

Fig. 2

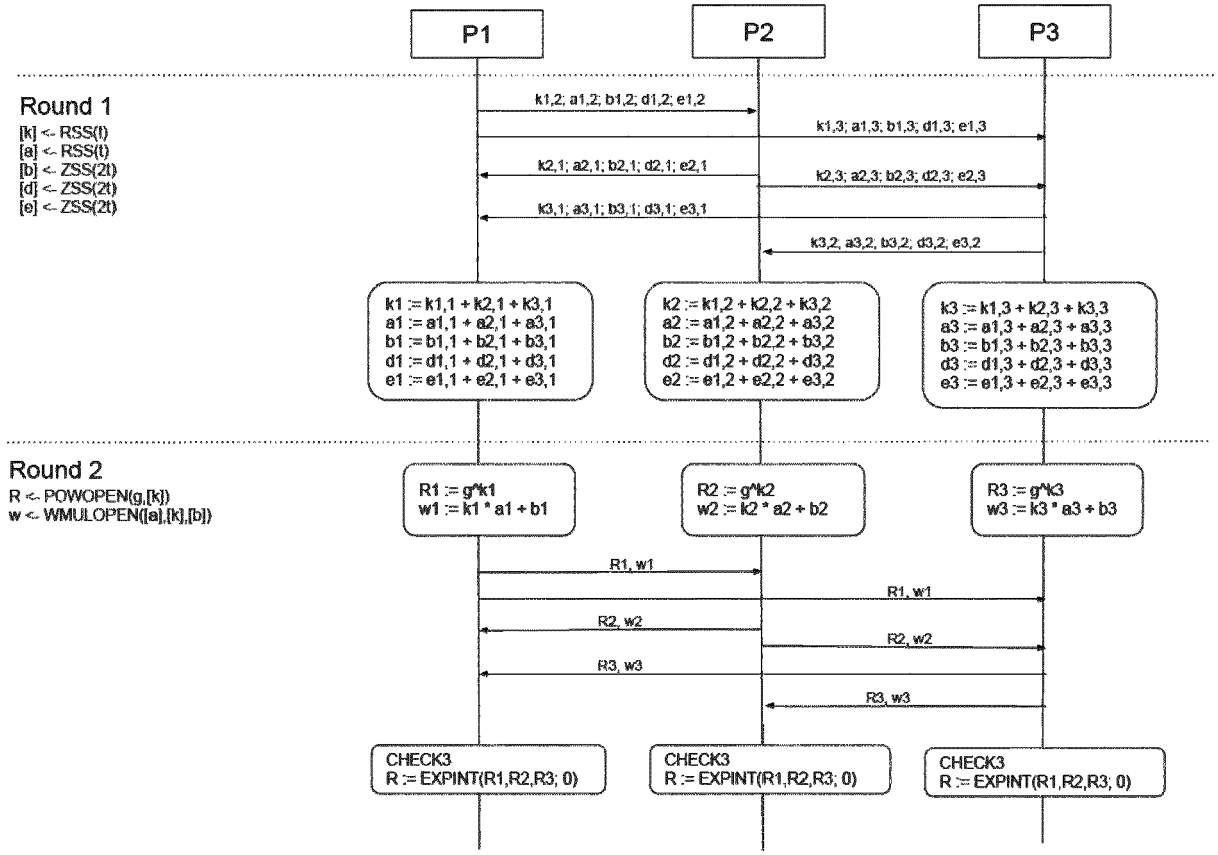
P1
P2
P3
Round 1
[k] <- RSS(t)
[a] <- RSS(t)
[b] <- ZSS(2t)
[d] <- ZSS(2t)
[e] <- ZSS(2t)
k1,2; a1,2; b1,2; d1,2; e1,2
k1,3; a1,3; b1,3; d1,3; e1,3
k2,1; a2,1; b2,1; d2,1; e2,1
k2,3; a2,3; b2,3; d2,3; e2,3
k3,1; a3,1; b3,1; d3,1; e3,1
k3,2; a3,2; b3,2; d3,2; e3,2
k1 := k1,1 + k2,1 + k3,1
a1 := a1,1 + a2,1 + a3,1
b1 := b1,1 + b2,1 + b3,1
d1 := d1,1 + d2,1 + d3,1
e1 := e1,1 + e2,1 + e3,1
k2 := k1,2 + k2,2 + k3,2
a2 := a1,2 + a2,2 + a3,2
b2 := b1,2 + b2,2 + b3,2
d2 := d1,2 + d2,2 + d3,2
e2 := e1,2 + e2,2 + e3,2
k3 := k1,3 + k2,3 + k3,3
a3 := a1,3 + a2,3 + a3,3
b3 := b1,3 + b2,3 + b3,3
d3 := d1,3 + d2,3 + d3,3
e3 := e1,3 + e2,3 + e3,3
Round 2
R <- POWOPEN(g,[k])
w <- WMULOPEN([a],[k],[b])
R1 := g^k1
w1 := k1 * a1 + b1
R2 := g^k2
w2 := k2 * a2 + b2
R3 := g^k3
w3 := k3 * a3 + b3
R1, w1
R1, w1
R2, w2
R2, w2
R3, w3
R3, w3
CHECK3
R := EXPINT(R1,R2,R3; 0)
CHECK3
R := EXPINT(R1,R2,R3; 0)
CHECK3
R := EXPINT(R1,R2,R3; 0)

Fig. 3a

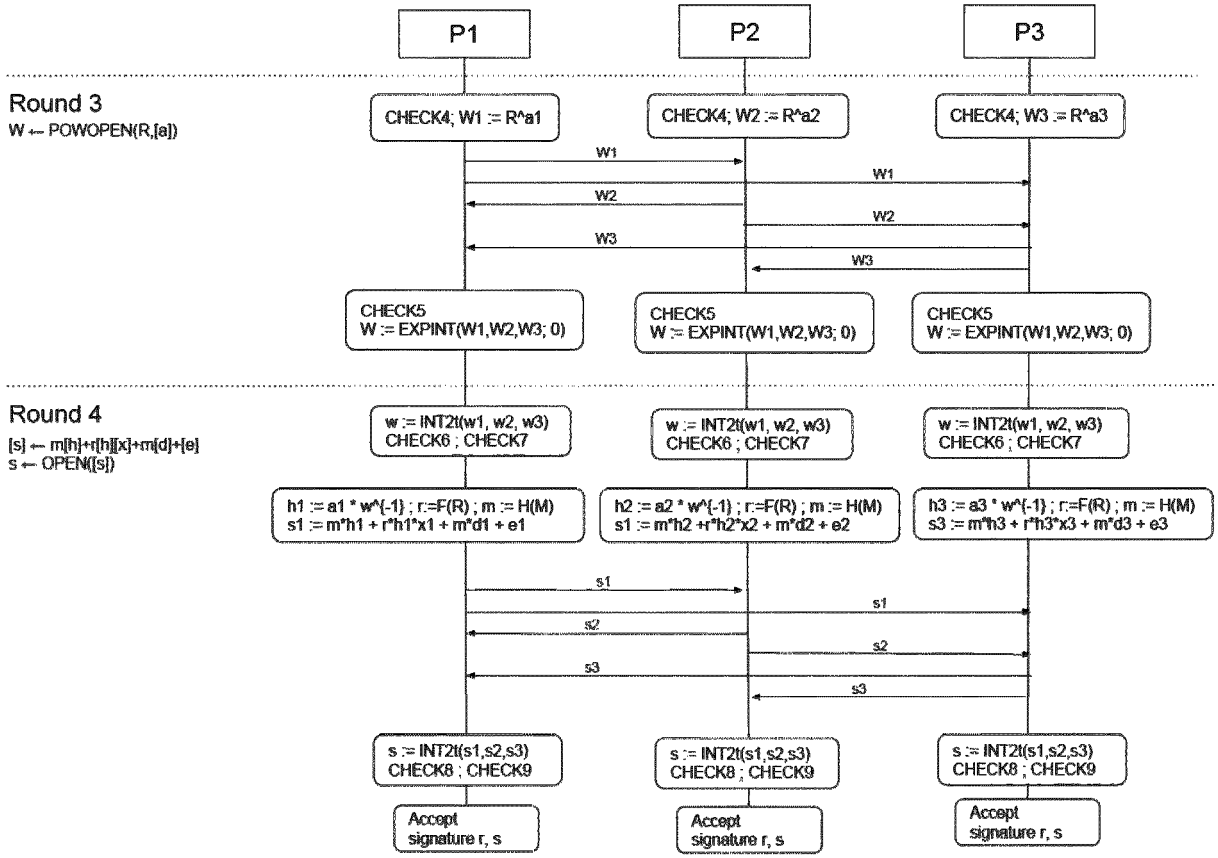
P1
P2
P3
Round 3
W ← POWOPEN(R,[a])
CHECK4; W1 := R^a1
CHECK4; W2 := R^a2
CHECK4; W3 := R^a3
W1
W2
W3
CHECK5
W := EXPINT(W1,W2,W3; 0)
Round 4
[s] ← m[h]+r[h][x]+m[d]+[e]
s ← OPEN([s])
w := INT2t(w1, w2, w3)
CHECK6 ; CHECK7
h1 := a1 * w^{-1} ; r:=F(R) ; m := H(M)
s1 := m*h1 + r*h1*x1 + m*d1 + e1
h2 := a2 * w^{-1} ; r:=F(R) ; m := H(M)
s1 := m*h2 +r*h2*x2 + m*d2 + e2
h3 := a3 * w^{-1} ; r:=F(R) ; m := H(M)
s3 := m*h3 + r*h3*x3 + m*d3 + e3
s1
s2
s3
s := INT2t(s1,s2,s3)
CHECK8 ; CHECK9
Accept
signature r, s

Fig. 3b

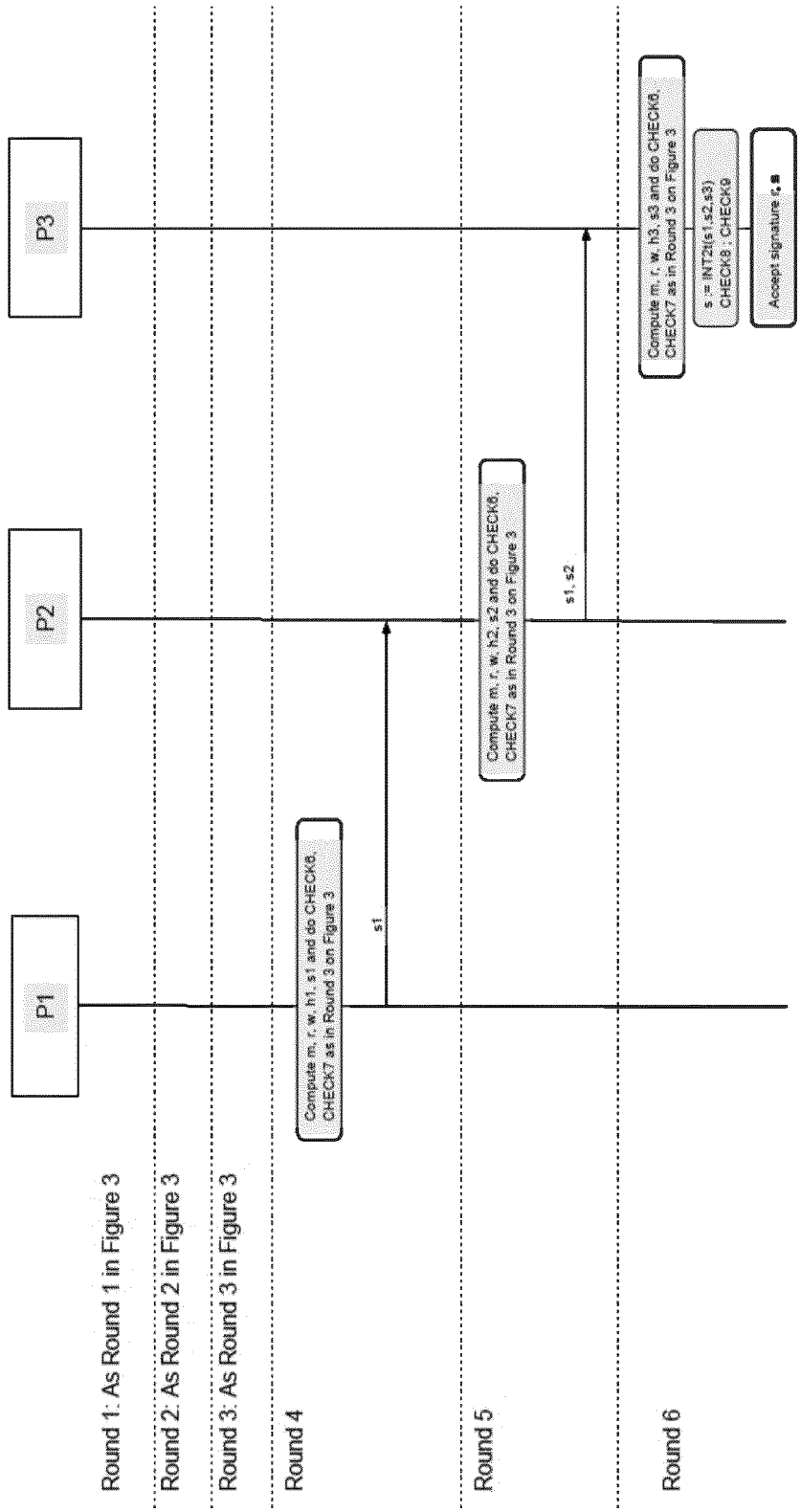
P1
P2
P3
Round 1: As Round 1 in Figure 3
Round 2: As Round 2 in Figure 3
Round 3: As Round 3 in Figure 3
Round 4
Round 5
Round 6
Compute m, r, w, h1, s1 and do CHECK6, CHECK7 as in Round 3 on Figure 3
s1
Compute m, r, w, h2, s2 and do CHECK6, CHECK7 as in Round 3 on Figure 3
s1, s2
Compute m, r, w, h3, s3 and do CHECK6, CHECK7 as in Round 3 on Figure 3
s := INT2I(s1,s2,s3)
CHECK8 ; CHECK9
Accept signature r, s

Fig. 4

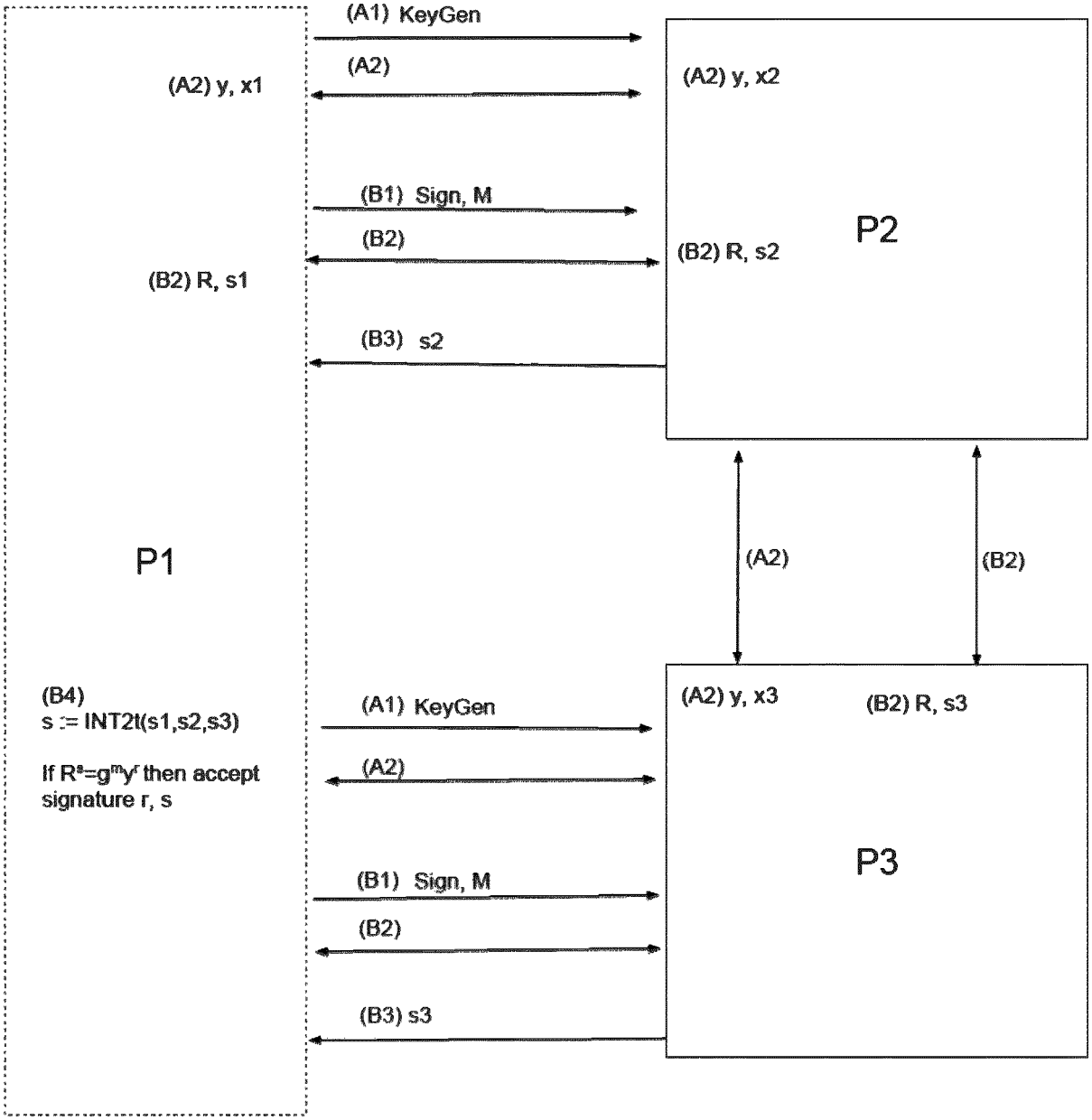
(A1) KeyGen
(A2)
(A2) y, x1
(A2) y, x2
(B1) Sign, M
(B2)
(B2) R, s1
(B2) R, s2
P2
(B3) s2
P1
(A2)
(B2)
(B4)
s := INT2t(s1,s2,s3)
If R^s=g^m y^r then accept signature r, s
(A1) KeyGen
(A2)
(A2) y, x3
(B2) R, s3
P3
(B1) Sign, M
(B2)
(B3) s3

Fig. 5

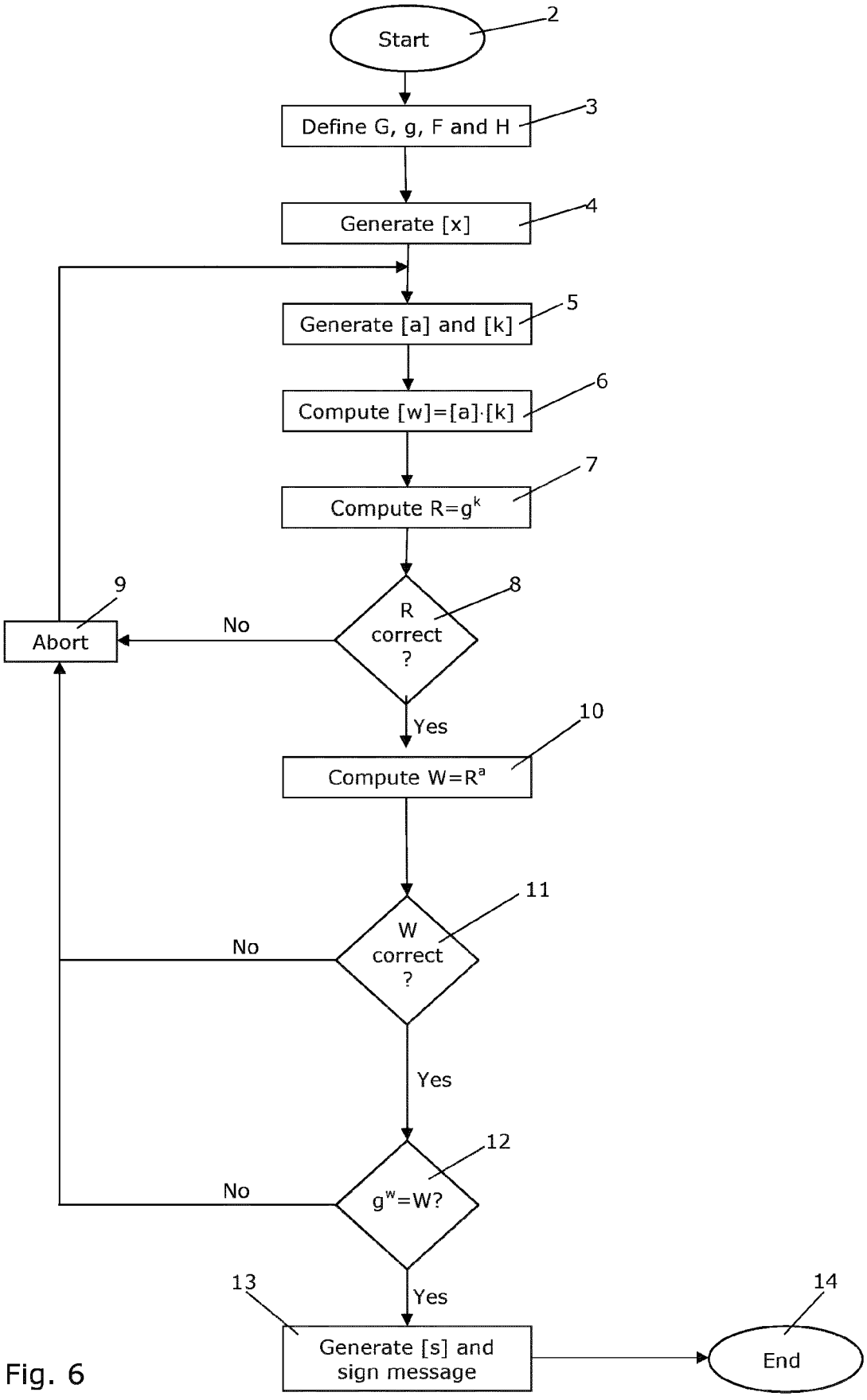
Start
2
Define G, g, F and H
3
Generate [x]
4
Generate [a] and [k]
5
Compute [w]=[a]·[k]
6
Compute R=g^k
7
R correct ?
8
No
Abort
9
Yes
Compute W=R^a
10
W correct ?
11
No
Yes
g^w=W?
12
No
Yes
Generate [s] and sign message
13
End
14

Fig. 6

## REFERENCES CITED IN THE DESCRIPTION

*This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.*

### Patent documents cited in the description


- WO 2015160839 A1 **[0007] [0008]**


### Non-patent literature cited in the description


- **ROSARIO GENNARO et al.** Robust Threshold DSS Signatures. *Information and Computation,* 2001, vol. 164, 54-84 **[0009]**