

IMPLICIT CERTIFICATE VERIFICATION

FIELD OF THE INVENTION

[0001] The present invention relates to public key cryptography.

SUMMARY OF THE INVENTION

[0002] Public key cryptography is used to permit secure communication between a pair of correspondents and to permit authentication of a message originating at one of the correspondents. In a public key cryptosystem each correspondent utilizes a private key and a public key related to the private key by a mathematical function. The mathematical function presents a "difficult" mathematical problem to ensure that a private key of a party cannot be obtained from the corresponding public key. Such problems include the difficulty in factoring the product of two large primes, as used in RSA cryptosystems, and the intractability of the discrete log problem over a finite field as used in the digital signature algorithm (DSA). Discrete log systems are widely used and a particular adaptation of such a system makes use of points of an elliptic curve defined over a finite field. Such systems, referred to as elliptic curve cryptosystems (ECC), offer high levels of security at smaller key sizes than other systems.

[0003] Where messages are to be exchanged securely, the public and private keys are used in particular protocols referred to generally as key agreement protocols to establish a shared key between the correspondents without disclosing the respective private keys. The shared key can then be used to encrypt and decrypt messages between the correspondents.

[0004] Elliptic curve public-key key agreement protocols typically include the following steps:

- *Key contributions.* Each party randomly generates a short-term (ephemeral) public key, from a private key that is a random integer and a seed point to provide a point representing the ephemeral public key and communicates the corresponding ephemeral public key to the other party (but not the private key). In addition, it may communicate its long-term static public key.
- *Key establishment.* Each party computes the shared key based on the static and ephemeral keys it received from the other party and based on the static and ephemeral

private keys it generated itself. Due to the properties of the elliptic curve, both parties arrive at the same shared key.

- *Key authentication.* Each party verifies the authenticity of the long-term static key of the other party, to obtain evidence that the only party that may be capable of computing the common key is, indeed, its perceived communicating party.
- *Key confirmation.* Each party evidences possession of the common key to the other party, usually by communicating a message authentication check value over the strings corresponding to the key contributions communicated by either party. This confirms to each party the true identity of the other party and proves that that party successfully computed the common key. This step may be done as part of the key agreement protocol or subsequently through the use of the shared keys.

[0005] The key authentication step is typically carried out independently of the other protocol steps described above and usually involves checking the validity of a certificate that vouches for the authenticity of the binding between a party and its private key by means of the corresponding public key. The separate key authentication step allows more flexibility (e.g., in the use of a certificate), but comes at significant cost, since the online computational cost of the key agreement protocol is dominated by the sum of the cost of the key authentication step and the cost of the key computation during the key establishment step.

[0006] The certificate is essentially a trusted party's signature of the public key of the correspondent together with the other information such as the name of the issuing party and the name of the certificate holder. In an ECC system, signatures are usually performed and verified using the ECDSA protocol although other signature protocols may be used. By way of example, ECDSA signature generation operates on the domain parameters of the ECC, a private key d , and a message m which, in the context of a certificate, will include the public key of the correspondent. The outputs are the signature (r,s) , where the signature components r and s are integers, and proceeds as follows.

1. Select a random integer $k \in_R [1, n - 1]$, n being one of the domain parameters.
2. Compute $kG = (x_1, y_1)$ and convert x_1 to an integer $\bar{x}_1$, where G is a point on an elliptic curve E and is one of the domain parameters.
3. Compute $r = \bar{x}_1 \bmod n$, wherein if $r = 0$, then go back to step 1.

4. Compute $e = H(m)$, where H denotes a cryptographic hash function whose outputs have a bit length no more than that of n (if this condition is not satisfied, then the outputs of H can be truncated).

5. Compute $s = k^{-1}(e + \alpha r) \bmod n$, where α is a long term private key of the signer.
If

$s = 0$, then go back to step 1.

6. Output the pair (r, s) as the ECDSA signature of the message m .

[0007] ECDSA signature verification operates on several domain parameters, a long term public key Q where $Q = \alpha G$, the message m , and the signature (r, s) derived above. ECDSA signature verification outputs a rejection or acceptance of the signature, and proceeds as follows.

1. Verify that r and s are integers in the interval $[1, n-1]$. If any verification fails then a rejection is returned.

2. Compute $e = H(m)$.

3. Compute $w = s^{-1} \bmod n$.

4. Compute $u_1 = ew \bmod n$ and $u_2 = rw \bmod n$.

5. Compute $R = u_1 G + u_2 Q = s^{-1} (eG + rQ)$ (from 3 and 4 above)

6. If $R = \infty$ then the signature is rejected.

7. Convert the x-coordinate x_1 of R to an integer $\bar{x}_1$; compute $v = \bar{x}_1 \bmod n$.

8. If $v = r$ then the signature is accepted, if not then the signature is rejected.

[0008] It will be appreciated that other signature generation and verification steps may be used such as ECGDSA and that certain steps to confirm the integrity of the signature are sometimes omitted.

[0009] Therefore, to authenticate the public key of the correspondent i.e. the message m which contains the public key of the correspondent and other information it is necessary to verify the signature on the certificate.

[0010] Subsequent to authentication, it is usual to confirm that the key of each party is indeed the same. During execution of a key agreement protocol between two parties A and

B, party A establishes the key K_A , while party B establishes the key K_B . Key confirmation consists of A evidencing knowledge of K_A to B by conveying some quantity $f(K_A)$, where f is some publicly known function, to B, who will compare this to his own computed value $f(K_B)$ and accept only if these values match (and vice versa). The function f is such that if $f(K_A)=f(K_B)$, then with overwhelming probability $K_A=\pm K_B$ and a common key has been established.

[0011] As noted above, the key authentication step adds significant computational load to the key agreement protocol. It is therefore an object of the present invention to obviate or mitigate the above disadvantages.

SUMMARY OF THE INVENTION

[0012] In general terms, the Applicant's have recognized that by computing conjointly a combination of the shared key and the certificate verification, an implicit verification of the certificate is obtained in the subsequent key confirmation step.

[0013] Preferably, the signature verification uses a homogeneous verification equation, i.e. one that equates to 0, such that failure to verify the certificate will result in an inequality between the computed values of the shared key.

[0014] In one aspect, there is provided a method of computing a cryptographic key to be shared between a pair of correspondents communicating with one another through a cryptographic system, one of the correspondents receiving a certificate of the other correspondents public key information to be combined with private key information of the one correspondent to generate the key, the method comprising the steps of computing the key by combining the public key information and the private key information and including in the computation a component corresponding to verification of the certificate, such that failure of the certificate to verify results in a key at the one correspondent that is different to the key computed at the other correspondent.

[0015] In another aspect, there is provided a computer readable media, a cryptographic unit and a system configured for performing the steps of the above method.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] An embodiment of the invention will now be described with reference to the accompanying drawings in which:

[0017] Figure 1 is a schematic representation of a data communication system.

[0018] Figure 2 is a representation of a first key agreement protocol.

[0019] Figure 3 is a representation of a further key agreement protocol.

DETAILED DESCRIPTION OF THE INVENTION

[0020] Referring therefore to Figure 1, a cryptographic communication system is generally denoted by numeral 10. The system 10 has a first correspondent 12 and a second correspondent 14 who may communicate with each other over a communication channel 16. The communication channel 16 may or may not be secure. Each correspondent has a cryptographic module 18 and 20 respectively, for performing cryptographic operations operating on the elliptic curve E defined over a chosen field. The cryptographic module 18 may be hardware, computer readable software instructions or a combination of both that is configured to perform cryptographic operations such as those described below.

[0021] The correspondent 12 has a long term private key "a" and a corresponding long term public key $A = aG$ where G is a base point on the curve specified in the domain parameters.

[0022] In order to authenticate the public key A to the correspondent 12, a certificate T_A is obtained from a trusted authority. The certificate includes the signature of the trusted authority on the public key A which can be verified using the trusted authority's public key Q .

[0023] Similarly, the correspondent 14 has a long term private key b , and a corresponding long term public key $B = bG$ and a certificate T_B of its public key. The certificate T_B may have been signed by another trusted party or using another public key than was used to obtain certificate T_A .

[0024] Typically, each correspondent 12, 14 can generate a random integer x, y respectively for use as a session or ephemeral private key that will generate corresponding ephemeral public keys xG, yG . However, for clarity, a first embodiment will be described in which a common key is established using the long term keys of each correspondent.

[0025] In a first key agreement protocol shown in Figure 2, an static elliptic curve Diffie Hellman (ECDH) key generation is employed with certificates T_A, T_B , signed using ECDSA.

[0026] Initially, the correspondents 12, 14 exchange public keys A, B and the respective certificates T_A , T_B . The common key is $K = abG$, which can be computed for each party from the public information received and the private information it has. Thus $K_A = aB$ and $K_B = bA$.

[0027] To verify the authenticity of the key, it is normal to verify the certificates T_A and T_B . However, in the present embodiment, the key generation and authentication are performed conjointly to provide efficiency. As noted above, the ECDSA provides in its verification that

$R = s^{-1} (eG + rQ)$. Accordingly $s^{-1} (eG + rQ) - R = \Sigma = O$ if the certificate verifies. The value of R can be recovered from the signature component r by taking r to be the value of the x-coordinate of the point R . The x-coordinate provides two possible y values and, to resolve the ambiguity, an indication of the correct value is included in the signature or there is a pre-agreed rule to select one of the values as the correct one that allows R to be computed. Additional techniques for recovering R are disclosed in US Application No. 11/333,296, the contents of which are incorporated by reference.

[0028] Each correspondent also has the signature (r,s) from the certificate T_A , T_B . To compute the key K , correspondent 12 computes the key K_A by combining the public key and private key information and including a component corresponding to the signature verification. Thus:

$$K_A = abG + \lambda_1(s_B^{-1}(e_B G + r_B Q_B) - R_B)$$

where

λ_1 is a random value

r_B , s_B are the signature components of the certificate T_B .

e_B is the hash of the message m in the certificate, which includes the public key B of correspondent 14 and;

R_B is the value recovered from r_B .

[0029] It will be recognized that the signature verification component $s_B^{-1} (e_B G + r_B Q_B) - R_B$ should be zero if the certificate authenticates the public key of B and therefore the computation should yield the key $K_A = aB$.

[0030] Similarly, the correspondent 14 computes $K_B = abG + \lambda_2(s_A^{-1}(e_A G + r_A Q_A) - R_A)$ where the values of r_A , s_A , and R_A relate to the certificate T_A . Again, if the certification authenticates, this should yield the key $K_B = bA$.

[0031] After each correspondent 12, 14 has computed the respective keys K_A , K_B , they implement the key confirmation step to exchange information that is a function of the common key. If the information received does not match that computed, then the key is rejected.

[0032] One reason for the common keys to differ is that the value obtained from the certificate verification portion is not zero, indicating that the certificate has not authenticated. By inserting a random value λ in each computation, which will usually be different for each the correspondents 12, 14, the efficacy of the authentication process is enhanced.

[0033] Thus, by conjointly computing the value of the common key and, a relationship corresponding to O , if the certificate authenticates, a reduction of the computation is obtained.

[0034] A further embodiment is shown in Figure 3 in which an MQV key agreement protocol is combined with authentication of certificates signed with ECDSA.

[0035] In the embodiment of Figure 3, correspondent 12 and 14 have respectively long term private keys a , b , long term public keys A , B , short term private keys x , y and short term public keys X , Y . In ECMQV the common key is of the form:

$$K = cX + dB$$

where

$$c = x + ag(x) \pmod{n} \text{ and}$$

$$d = cg(Y) \pmod{n}$$

where g is publicly known representation function that maps elliptic curve points to a suitable integer.

[0036] Assuming each correspondent has the certificate T_A , T_B of the other, each correspondent sends the short term public key X , Y , respectively to the other correspondent.

[0037] The correspondent 12 computes $s_{12} = (x + ag(x)) \pmod{n}$. Correspondent 14 computes $s_{14} = (y + bg(y)) \pmod{n}$. The common key K can then be computed as

$$K = s_{12} (Y + g(y)B) = s_{14} (X + g(x)A)$$

[0038] To authenticate the key, the certificate of the correspondents long term public key must be authenticated. Accordingly, as described above, the common key K is computed by correspondent 12 as

$$K_A = s_{12} (Y + g(y)B) + \lambda \Sigma = (x + ag(x)) (Y + g(y)B) + (\lambda_1 e_B s_B^{-1}) G + (\lambda_1 r_B s_B^{-1}) Q + \lambda_1 R_B$$

and by correspondent 14 as

$$K_B = s_{14} (X + g(x)A) + \lambda \Sigma = (y + bg(y)) (X + g(x)A) + (\lambda_2 e_A s_A^{-1}) G + (\lambda_2 r_A s_A^{-1}) Q + \lambda_2 R_A$$

[0039] For each correspondent, λ is a random integer and Σ is the signature verification relationship $s^{-1}(eG + rQ) - R$ for the respective certificates.

[0040] Assuming $\Sigma = O$, the keys will match and the certificate will be authenticated implicitly.

[0041] The efficiencies obtained may be attributed to savings on point doubling operations and the possibility to use multiple point multiplication strategies.

[0042] From a security point, the random values of λ should be chosen from a set of size $O(\sqrt{n})$ since this corresponds to the workload of solving the ECDLP problem. The set from which λ is chosen may be varied, with consequent variation in security. If λ is drawn from a set of size 2^t , where $t < (\log_2 n)/2$, the key authentication strength may go down to t bits. With such a choice of λ , computing $aB + \lambda R$ involves $wt_H(a)$ point additions involving B or $B - R$, where wt_H is the Hamming weight of the binary representation of the value a . This is the same number of point additions involved in computing aB .

[0043] The quantity $(\lambda e s^{-1})G + (\lambda r s^{-1})Q$ can be computed using non-adjacent form (NAF) as described more fully in Guide to Elliptic Curve Cryptography, Vanstone et al., Springer, at pages 98 to 101, the contents of which are incorporated by reference.

[0044] The workload is evaluating K_A in this way is approximately m point additions and m doublings, where m is the bit size of the curve.

[0045] By comparison, evaluating $K_A = aB$ and verifying the ECDSA signature separately has a workload of $5m/6$ additions and $2m$ doubles.

[0046] A 30% improvement is therefore attainable. If the “fast verify” techniques shown in US Patent Application No. 11/333,296, noted above, are used than savings in the order of 10% are attainable.

[0047] One can make further improvements by considering λ with integers in their NAF representation, rather than in their binary representation. Since $wt_H(a) \approx m/3$, one has $|L| \approx 3^{m/3} > \sqrt{n}$ (note that all integers in L are in NAF form, so are unique). Carrying out a similar analysis as above, the approach given here yields approximately $5m/6$ adds and m doubles. In this case, the approach given here yields $1.52m$ adds, rather than $1.70m$ adds with the binary approach described above, thus yielding an efficiency improvement of 22%, resp. 47% compared to the separate computations (Fast Verify, resp. Ordinary ECDSA Verify). Note that in this case one might write: $aB - \lambda R = (a - \lambda^+ + \lambda^-) B - \lambda^+ (R - B) - \lambda^- (R + B)$; where $\lambda = \lambda^+ + \lambda^-$ and λ^+ and λ^- and denote those components of λ with either the same or different sign respectively as corresponding components of a .

[0048] One can further generalize this towards w -NAF representations described in Guide to Elliptic Curve Cryptography, Vanstone et al., Springer, at pages 99 ($w \geq 2$). Then $|L| = q^t$, where $q := 2^{w-1} + 1$ and $t := wt_H(a)$ (all integers are in w -NAF form, so are unique). If one assumes $wt(a) \approx m/(w+1)$, one obtains $L = (2^{w-1} + 1)^{m/(w+1)} \geq 2^{m/2} \approx \sqrt{n}$. A more detailed analysis yields that the additions in $aB + \lambda R$ can be computed at the same cost as those in aB (using the same method), except for the latter having far larger pre-computation cost approx. $2^{(w+1)(w-2)/2}$ rather than 2^{w-2} . Storage cost is as follows: 1 vs. 3 ($w=2$), 2 vs. 8 ($w=3$), 4 vs. 24 ($w=4$), 8 vs. 64 ($w=5$). Hence, if storage cost is the metric of choice, comparisons are somewhat misleading: if one stores 8 points and uses w -NAF techniques only, then one can compute aB using a 5-NAF or use a 3-NAF computation to compute $aB + \lambda R$, with cost $0.166m$ additions vs. $0.25m$ additions with the present technique. Thus, the present technique can be thought of as saving m doubles in the evaluation of the ECDSA signature at the expense of $0.083m$ additional additions in the evaluation of $aB + \lambda R$.

[0049] This approach basically removes the doubling operations from the ECDSA verification procedure, i.e., the incremental cost of ECDSA verification becomes $m/2$ adds only, rather than (after scaling) $0.85m$ adds (Fast Verify), resp. $1.02m$ adds (ordinary ECDSA verify), i.e., presenting a 70-104% efficiency improvement of the ECDSA verification

procedure. Further improvements may be obtained by more fully exploiting the repertoire of multiple point multiplication methods and by using windowing and w -NAF methods.

[0050] With the embodiment of Figure 3 involving MQV and ECDSA:

$$K_A' = c Y + d B + \lambda \Sigma = c Y + d B - \lambda R + ((\lambda e s^{-1}) G + (\lambda r s^{-1}) Q).$$

[0051] Let $L := \{x \mid \text{supp}(x) \subset \text{supp}(c) \cup \text{supp}(d)\}$, where integers x are represented in binary representation. If c and d are in their JSF-representation, then one has $\text{supp}(c) \cup \text{supp}(d)$ has weight $\approx m/2$ (where $m := \log_2 n$), so $|L| \approx \sqrt{n}$. With this choice of λ computing $K_A + \lambda R$ involves precisely $wt_H(c, d)$ point additions. λ is chosen such that the nonzero components hereof have the same sign as the corresponding coefficient in $c Y + d B$ with respect to pre-computed points $Y, B, Y-B, Y+B$. (This ensures that all λ are, indeed, unique.) So, all computations can be done involving the 8 points $Y, B, Y+B, Y-B, R+Y, R+B, R+Y+B, R+Y-B$ (rather than requiring 12 points). The quantity $(\lambda e s^{-1}) G + (\lambda r s^{-1}) Q$ can be computed using a joint NAF form, once λ is determined as indicated above. The total workload involved in evaluating K_A' this way is approximately m point additions and m point doublings. Obviously, improvements are possible, by exploiting the repertoire of multiple point multiplication methods and by using windowing and w -NAF methods. The main improvement, however, is that the combined computation offers the perspective of considerably saving on doubling operations.

[0052] The approach above can be used so as to yield a simple side channel attack resistant implementation. Details are as follows. Let $L := \{x \mid \text{supp}(x) \subset \text{supp}(c)\}$, where we consider all integers in binary representation. Since $wt(c) \approx m/2$, one has $|L| \approx \sqrt{n}$. Now, let $\lambda = \lambda^+ + \lambda^-$, where λ^+ is a random element of L and where λ^- is set to 1 for all positions outside $\text{supp}(c)$ and to 0 otherwise. The computation of $c Y + d B + \lambda^+ R$ can be carried out using the 5 pre-computed points $Y, B, Y+B, B+R$, and $Y+B+R$. The quantity $\lambda^- R + ((\lambda e s^{-1}) G + (\lambda r s^{-1}) Q)$ can be computed, once λ is determined as indicated above using the 7 nonzero partial sums involving R, Q , and G . By pre-computing $22 = 5 \times 3 + 3 + 4$ points, one can compute K_A' using multiple point multiplication, using exactly $m+22$ point additions and m point doubles. The choice of λ^- ensures that each step in the binary method for point multiplication involves an addition. By taking $\lambda^- := 0$, it would have been possible to pre-compute only 19 points, but then it would not have been possible to enforce one addition per iteration round and so a side

channel attack may have been possible. This could have been ensured by picking another λ^+ until such a condition is obtained, however, this might leak side channel information by itself. The approach above has approximately the same cost as the method for prevention of simple side channel attacks based on adding dummy point addition operations during the computation of K_A , but achieves ECDSA signature verification as well. Thus, ECDSA signature verification comes virtually for free with this approach.

[0053] It is assumed that Σ lies in the prime order subgroup generated by G . It should have sufficiently large order. If Σ corresponds to an ECDSA verification equation this condition is easily checked by checking that $hR \neq O$, where h is the co-factor of the curve.

[0054] The approach described above leads to performance improvements, due to savings on doubling operations and the prospects of exploiting multiple point multiplication strategies. Moreover, this approach is not restricted to combining key authentication and key establishment steps during the execution of a key agreement protocol; it can be applied to any setting where one has to compute some keys and verify some elliptic curve equations (in fact, the key need not even be secret). This being said, the approach is most useful in settings where one can verify whether the computed key is correct (since this would yield a verdict on whether the homogeneous elliptic curve equation $\Sigma = O$ holds). Thus, the approach works effectively in all settings where one indeed evidences knowledge of the computed key.

[0055] The method for combining verification with key computation works in more general settings than described above. Verification of multiple ECDSA signatures (certificate chains) may be obtained by using combinations of computation and verification equations with a different λ for each verification. Similarly, verification of any elliptic curve and batch verification of multiple elliptic curve equations is possible.

[0056] Key computation with ECDH-schemes in ANSI X9.63, NIST SP800-56a (including ECIES, Unified Model, STS, ECMQV, ElGamal encryption) is also possible. Computation of non-secret ECC point provided correctness can be checked for example in a PV signature with an ECDSA certificate. Similarly, computation of multiple ECC points provided correctness can be checked.

[0057] It is also possible to apply such techniques to operation in other algebraic structures (including hyper-elliptic curves, identity-based crypto systems, etc.).

[0058] The use of λ to provide simple side channel resistance is virtually for free.

[0059] Although the above has been described in terms of certain exemplary embodiments, it will be appreciated that variations thereof are possible within the scope of the claims which follow.

Claims

1. A computer implemented method of conjointly computing a cryptographic key and including a component in said key for authenticating said key, said key to be shared between a pair of correspondents (12,14) communicating with one another through a cryptographic system (10), one of said correspondents (12,14) receiving a certificate of the other correspondents (12,14) public key information to be combined with private key information of the one correspondent (12,14) to generate said key, said method comprising the steps of computing said key by combining said public key information and said private key information and including in said key said component corresponding to verification of said certificate, such that failure to verify said certificate therefore results in a key at said one correspondent (12,14) that is different to the key computed at said other correspondent.
2. A method according to claim 1 wherein said verification utilises a signature contained in said certificate.
3. A method according to claim 1 or claim 2 wherein said component equates to zero when said certificate verifies.
4. A method according to any one of claims 1 to 3 wherein a random element is included in said component to mask verification of said certificate.
5. A method according to claim 4 wherein said random element is selected to provide repetitive iterations in verification of said signature and thereby inhibit side channel attacks.
6. A method according to any one of the preceding claims wherein said cryptographic system is an elliptic curve cryptographic system.
7. A method according to claim 6 wherein said computation is performed using multiple point multiplications.

8. A method according to claim 6 or claim 7 wherein said certificate includes a signature performed by an elliptic curve signature protocol.
9. A method according to claim 8 wherein said signature protocol is ECDSA.
10. A method according to claim 6 wherein said public key information is representative of an elliptic curve point and combination of public key information and said private key information includes a point multiplication.
11. A method according to claim 10 wherein verification of said signature requires multiple point multiplications.
12. A method according to claim 11 wherein said computation is performed using multiple point multiplications.
13. A method according to any one of the preceding claims including the step of confirming the keys generated by said one correspondent conforms to that of the other correspondent.
14. A system for conjointly computing a cryptographic key and including a component in said key for authenticating said key, said key to be shared between a pair of correspondents (12,14) communicating with one another through a cryptographic system (10), one of said correspondents (12,14) receiving a certificate of the other correspondent's (12,14) public key information to be combined with private key information of the one correspondent (12,14) to generate said key, said system comprising means to compute said key by combining said public key information and said private key information and including in said key said component corresponding to verification of said certificate, such that failure to verify said certificate therefore results in a key at said one correspondent that is different to the key computed at said other correspondent.
15. A computer readable medium having computer executable instructions stored thereon for execution on a processor of a computing system so as to cause said computing system to implement the steps of the method of any one of claims 1 to 13.

16. A cryptographic unit configured to perform the method according to any one of claims 1 to 13.