Blind Collective Signature Protocol Based on Discrete Logarithm Problem

Nikolay A. Moldovyan and Alexander A. Moldovyan
(Corresponding author: Nikolay A. Moldovyan)

St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences
14 Liniya, 39, St. Petersburg 199178, Russia (Email: nmold@mail.ru)
(Received Aug. 3, 2009; revised and accepted Nov. 15, 2009)

Abstract

Using Schnorr's digital signature (DS) scheme as the underlying scheme there is designed the collective DS protocol. In the proposed collective DS protocol the signature is formed simultaneously by all signers, therefore using this protocol leads to natural solution of the problem of signing simultaneously a contract. Using the proposed collective DS protocol the blind collective DS protocol has been designed, which is a new type of the multi-signature schemes. For simultaneous signing a package of different contracts by different sets of signers it is proposed another new multi-signature scheme called composite signature.

Keywords: Blind collective signature, blind signature, composite signature, collective digital signature, digital signature, discrete logarithm problem

1 Introduction

The digital signatures (DS) are widely used in information systems. To solve different practical problems connected with electronic documents authentication a variety of the DS protocols has been proposed [11, 14, 15]. A particular type of the protocols, called multi-signature protocols, provides to form a single signature shared by several signers [1, 3]. Recently a particular variant of the multi-signature schemes, called collective DS, has been proposed [9]. That variant of the multi-signature protocols is based on the difficulty of finding large prime roots modulo a 1024-bit prime p possessing the structure $p = Nk^2 - 1$, where N is an even number and k is a 160-bit prime. The protocol produces a fixed size collective DS for arbitrary number of signers, however the DS length is sufficiently large, actually, 1184 bits.

Using the general design of the collective DS scheme [9] in this paper there is designed the collective DS protocol based on difficulty of finding discrete logarithm, which produces a 320-bit collective signature. The proposed collective DS scheme has been used to construct a new type multi-signature scheme called blind collective DS protocol. It can be applied, for example, in the electronic money systems in which the electronic banknotes are issued by several banks. It is also described another new protocol called composite DS protocol characterized in that the document-dependent public keys are used.

2 Collective Signature Protocols

2.1 Protocol Based on Difficulty of Finding Roots Modulo A Prime

In the paper [9] there is proposed a DS scheme based on difficulty of finding the kth roots modulo large prime p such that $k^2|p-1$, where k and p are primes. In that scheme the public key Y is computed as follows $Y = X^k \bmod p$, where X is the secret key. A signature to some message M consists of two numbers (E, S). The signature verification is performed in three steps:

1) It is computed value $R^* = Y^E S^k \bmod p$;
2) Using some specified hash function F_H it is computed the hash value E^* from the message M to which the value R is concatenated: $E^* = F_H(M\|R)$, where $\|$ denotes the concatenation operation;
3) The value E^* is compared with E. If $E^* = E$, then the signature is valid. Otherwise the signature is rejected.

The upper boundary of the security of the DS scheme is defined by the difficulty of finding the kth roots $\bmod p$ in the mentioned special case. To estimate the security in [9] there are proposed two methods for finding the kth roots $\bmod p$. Independently of the length of the modulo p the difficulty of the fist method is estimated as $O(\sqrt{k})$, where $O(\cdot)$ is the order notation (see Appendix 1). The first method is efficient for arbitrary value p, if $|k| < 160$ bits, where $|k|$ denotes the binary length (size) of the number k. In the second method the most difficult procedure is finding discrete logarithm modulo p, therefore it has subexponential difficulty. The second method is efficient for arbitrary prime k ($k < \sqrt{p}$), if $|p| < 1024$ bits (see Appendix 2).

Taking into account that methods the $O(2^{80})$ difficulty of finding the kth roots mod p is provided, if the primes k and p have the length $|k| \geq 160$ bits and $|p| \geq 1024$ bits. Sufficiently large size of the modulus p defines sufficiently large size of the DS produced by the algorithms based on the mentioned difficult problem. In the best case the signature size is equal to 1184 bits.

Specific design of that DS scheme has been used to construct a multi-signature scheme that works as follows. Suppose the jth user owns the private key $X_j < p$ and the public key $Y_j = X_j^k \bmod p$, where $j = 1, 2, \cdots, n$. Suppose some subset of m users is to sign a message M with some single DS called collective DS. The following protocol solves the problem.

1) Each α_ith user generates a random value $t_{\alpha_i} < p$ (it is a disposable secret key) and calculates the value $R_{\alpha_i} = t_{\alpha_i}^k \bmod p$, where $i = 1, 2, \cdots, m$ and $\forall i: \ \alpha_i \in \{1, 2, \cdots, n\}$.

2) The common randomization value R is computed: $R = R_{\alpha_1} R_{\alpha_2} \cdots R_{\alpha_m} \bmod p$.

3) The first part E of the collective DS (E, S) is computed using some specified hash function F_H: $E = F_H(M \| R)$.

4) Using the common value E and individual disposable secret key t_{α_i} each of the users computes its share in the collective DS: $S_{\alpha_i} = X_{\alpha_i}^{-E} t_{\alpha_i} \bmod p$, $i = 1, 2, \cdots, m$.

5) Compute the second part S of the collective DS: $S = S_{\alpha_1} S_{\alpha_2} \cdots S_{\alpha_m} \bmod p$.

The collective DS verification is performed as follows.

1) Compute the collective public key Y as product $Y = Y_{\alpha_1} Y_{\alpha_2} \cdots Y_{\alpha_m} \bmod p$.

2) Using the signature (E, S) compute value R^*: $R^* = Y^E S^k \bmod p$.

3) Compute $E^* = F_H(M \| R^*)$.

4) Compare values E^* and E. If $E^* = E$, then the signature is valid. Otherwise the signature is rejected.

The described protocol works correctly, i. e. it produces the collective signature (E, S) that satisfies the signature verification equation, in which there is used the public key that is equal to $Y = \prod_{i=1}^m Y_{\alpha_i} \bmod p$. Indeed, computation of the value R^* gives

$$R^* \equiv Y^E S^k \equiv \left(\prod_{i=1}^m Y_{\alpha_i}\right)^E \left(\prod_{i=1}^m S_{\alpha_i}\right)^k \equiv$$

$$\equiv \left(\prod_{i=1}^m X_{\alpha_i}\right)^{kE} \left(\prod_{i=1}^m X_{\alpha_i}^{-E} t_{\alpha_i}\right)^k \equiv$$

$$\equiv \prod_{i=1}^m t_{\alpha_i}^k \equiv \prod_{i=1}^m R_{\alpha_i} \equiv R \bmod p \Rightarrow$$

$$\Rightarrow E^* = F_H(M \| R^*) = F_H(M \| R) = E.$$

The main advantage of the described protocol consists in its internal integrity. Namely, in the protocol none of the signers generates his individual signature. He generates only its share in the collective DS that corresponds exactly to the set of m user presented by numbers $\alpha_1, \alpha_2, \cdots, \alpha_m$. Besides, it is computationally difficult to manipulate with shares S_j $j \in \{\alpha_1, \alpha_2, \cdots, \alpha_m\}, \cdots$ and compose another collective DS, relating to some different set of users. Due to the internal integrity of the collective DS protocol it solves naturally the problem of signing simultaneously a contract [11]. Note that the multi-signature protocols proposed in [1] are not able to solve this problem without the help of some trusted party participating in the protocol.

However the considered protocol produces collective signature having comparatively large size that equals to 1184 bits in the case of 80-bit security. To reduce the signature length there is proposed the collective DS protocol described in the next subsection.

2.2 Protocol Based on Difficulty of Finding Discrete Logarithm

The problem of reducing the collective signature size is solved using the computationally difficult problem of finding discrete logarithm in the finite field $\mathbb{F}_p$, where p is a sufficiently large prime. The following collective DS protocol combining Schnorr's DS scheme [12] with general construction of the protocol by [9] produces the 320-bit collective signature. Suppose there is used a prime modulus p such that $p-1$ contains a large 160-bit prime factor q, the element g that is generator of the q order subgroup in $\mathbb{F}_p^*$, and public key $Y = g^x \bmod p$, where x is the secret key. Selected sizes of the parameters p and q provide the 80-bit security. Suppose also that m users should sign the given message M. The collective DS protocol works as follows.

1) Each of the users generates his individual random value t_i and computes $R_i = g^{t_i} \bmod p$.

2) It is computed the common randomization parameter as the product $R = R_1 R_2 \cdots R_m \bmod p$.

3) Using the common randomization parameter R and some specified 160-bit hash function F_H it is computed the first element E of the collective DS: $E = F_H(M \| R)$, where M is the message to be signed and $\|$ is the concatenation operation.

4) Each of the users computes his share S_i in the second element of the collective DS $S_i = t_i + x_i E \bmod q$, $i = 1, 2, \cdots, m$.

5) The second element S of the collective DS (R, S) is computed as follows $S = S_1 + S_2 + \cdots + S_m \bmod q$.

Size of the value S is equal to 160 bits, since it is computed modulo a 160-bit value q. The total size of the signature (E, S) is 320 bits that is significantly less then in the case of the collective DS protocol based on difficulty of finding large prime roots modulo a 1024-bit prime p. The signature verification is performed exactly as in Schnorr's DS algorithm [12], except the first step is added:

1) Compute the collective public key as product of individual public keys of each of the users: $Y = Y_1 Y_2 \cdots Y_m \bmod p$.

2) Using the collective signature (E, S) shared by the given set of m users compute the value $R^* = Y^{-E} g^S \bmod p$.

3) Compute the value $E^* = F_H(M \| R^*)$.

4) Compare the values E^* and E. If $E^* = E$, the collective DS is valid, otherwise the signature is rejected.

The proposed collective DS protocol works correctly. Indeed,

$$\begin{aligned} R^* &\equiv Y^{-E} g^S \equiv Y^{-E} g^{\sum_{i=1}^{m}(t_i + x_i E)} \\ &\equiv Y^{-E} g^{\sum_{i=1}^{m} t_i} g^{E \sum_{i=1}^{m} x_i} \equiv Y^{-E} g^{\sum_{i=1}^{m} t_i} Y^E \\ &\equiv \prod_{i=1}^{m} g^{t_i} \equiv \prod_{i=1}^{m} R_i \bmod p \\ \Rightarrow R^* &= R \\ \Rightarrow E^* &= F_H(M, R^*) = F_H(M, R) = E. \end{aligned}$$

Since the equality $E^* = E$ holds, the collective signature produced with the protocol satisfies the verification procedure, i.e. the described collective signature protocol is correct. Security items of the protocol are considered in the following subsection.

2.3 Attacks on The Collective DS Protocol

Let us consider security of the proposed collective DS protocol based on the discrete logarithm problem. The participants of the collective DS protocol have significant more possibilities to attack the protocol than outsiders. Therefore below there are discussed the following two types of attacks. The first type corresponds to forgery of the collective DS. The second type corresponds to calculation of the secret key of one of the signers that shares a collective DS.

The first attack. Suppose it is given a message M and $m-1$ signers attempts to create a collective DS corresponding to m signers owning the collective public key $Y = Y^* Y_m \bmod p$, where $Y^* = \prod_{i=1}^{m-1} Y_i \bmod p$, i.e. $m-1$ users unite their efforts to generate a pair of numbers (E^*, S^*) such that $R^* = Y^{-E^*} g^{S^*} \bmod p$ and $E^* = F_H(M \| R^*)$. Suppose that they are able to do this, i.e. the collective forger (i.e. the considered $m-1$ signers) is able to calculate a valid signature (E^*, S^*) corresponding to collective public key $Y = Y_1 Y_2 \cdots Y_m \bmod p$ The collective DS satisfies the following relation:

$$\begin{aligned} R^* &\equiv Y^{-E^*} g^{S^*} \equiv (Y^* Y_m)^{-E^*} g^{S^*} \\ &\equiv Y^{*-E^*} Y_m^{-E^*} g^{S^*} \\ &\equiv g^{*-E^* \sum_{i=1}^{m-1} x_i} Y_m^{-E^*} g^{S^*} \\ &\equiv Y_m^{-E^*} g^{S^* - E^* \sum_{i=1}^{m-1} x_i} \bmod p \\ \Rightarrow R^* &= Y_m^{-E^*} g^{S^{**}}, \end{aligned}$$

where $S^{**} = S^* - E^* \sum_{i=1}^{m-1} x_i$. The collective forgery have computed the signature (E^*, S^{**}) which is a valid signature (to message M) of the mth signer, since E^* is equal to $F_H(M \| R^*)$ and the pair of numbers (E^*, S^{**}) satisfies the verification procedure of the underlying DS scheme. Thus, any successful attack breaking the collective DS protocol also breaks the underlying DS algorithm. Since it is known that the Schnorr's DS scheme is a provably secure one [6, 10] the proposed protocol is also secure (if it is not secure, then using the proposed attack two or more persons are able to forge a signature of the underlying DS algorithm, i.e. to break Schnorr's DS scheme).

The second attack. Suppose that $m-1$ signers that share some collective DS (R, S) with the mth signer are attackers trying to calculate the secret key of the mth signer. The attackers know the values R_m and S_m generated by the mth signer (see the protocol description). This values satisfy the equation $R_m = Y_m^{-E} g^{S_m} \bmod p$, where the values R_m and E are out of the attackers control, since the value $R_m = g^{t_m} \bmod p$, where t_m is a random number generated by the mth signer, and E is the output of the hash function algorithm. It is supposed that a secure hash function is used in the protocol, therefore the attackers are not able to select the value R producing some specially chosen value E. This means that, like in the case of underlying Schnorr's DS scheme, computing the secret key requires solving the discrete logarithm problem, i.e. i) to find $t_m = \log R_m$ and then compute $x_m = E^{-1}(S_m - t_m) \bmod q$ or ii) to compute $x_m = \log Y_m$.

In analogous way applying the considered two attacks to the collective DS protocol described in Subsection 2.1 one can shown that it is as secure as the undelying DS algorithm based on difficulty of finding the kth roots mod p is secure.

3 Blind Collective Signature Protocol

The collective DS scheme proposed in Section 2 can be used to design on its base the blind collective signature

protocol that uses the blinding factors Y^{τ} and g^{ϵ} applied earlier to construct a blind signature scheme based on Schnorr's DS scheme [10, 16]. The following scheme is a variant of the implementation of the blind collective DS protocols. Suppose some user U is intended to get a collective DS (corresponding to message M) of some set of m signers using a blind signature generation procedure.

The following protocol solves the indicated problem.

1) Each signer generates a random value $t_i < q$ and computes $R_i = g^{t_i} \bmod p$, and presents the value R_i to each of the signers.

2) It is computed the common randomization parameter as the product $R = R_1 R_2 \cdots R_m \bmod p$.

3) The value R is send to user U.

4) User U generates random values $\tau < q$ and $\epsilon < q$.

5) User U computes the value $R' = RY^{\tau} g^{\epsilon} \bmod p$.

6) User U calculates the value $E' = F_H(M \| R')$ that is the first parameter of the collective DS.

7) User U calculates the value $E = E' + \tau \bmod q$.

8) User U presents the value E to the signers.

9) Each signer using his individual value t_i and his secret key x_i computes his "blind" share in the collective DS: $S_i = t_i + x_i E \bmod q$.

10) It is computed the second part S of the blind collective DS: $S = S_1 + S_2 + \cdots + S_m \bmod q$.

11) User U computes the second parameter of the collective DS: $S' = S + \epsilon \bmod q$.

The signature verification procedure is exactly the same as described in the case of collective DS. The signature (E', S') is a valid collective DS corresponding to the message M. Indeed, using the collective public key $Y = Y_1 Y_2 \cdots Y_m \bmod p$ we get

$$\begin{aligned} R^* &\equiv Y^{-E'} g^{S'} \equiv Y^{-(E-\tau)} g^{S+\epsilon} \equiv Y^{-E} Y^{\tau} g^{S} g^{\epsilon} \\ &\equiv g^{-E\sum_{i=1}^{m} x_i} Y^{\tau} g^{\sum_{i=1}^{m}(t_i + x_i E)} g^{\epsilon} \\ &\equiv RY^{\tau} g^{\epsilon} \bmod p \\ \Rightarrow R^* &= R' \\ \Rightarrow E^* &= F_H(M \| R^*) = E'. \end{aligned}$$

Thus, the protocol works correctly and the described procedure yields the collective DS (E', S') that is known for user U and unknown for each of the signers. The protocol provides anonymity of the user in the case when the message M and collective signature (E', S') will be presented to all or to one of the signers. Here it is supposed that many different users present electronic messages to some given set of signers for blind signing. Suppose the signers save in a data base all triples (E, S, R) Produced by all of the performed blind collective DS procedures. Let (E_1, R_1, S_1) and (E_2, R_2, S_2) are two of such triples. Accordingly to the blind collective DS protocol construction the elements of the first triple satisfy the expression:

$$R_1 = Y^{-E_1} g^{S_1} \bmod p. \quad (1)$$

The signature (E', S') satisfy the expression:

$$R' = Y^{-E'} g^{S'} \bmod p. \quad (2)$$

Dividing Equation (2) by (1) we get

$$\frac{R'}{R_1} = Y^{E_1 - E'} g^{S' - S_1} \bmod p,$$

therefore $R' = R_1 Y^{\tau} g^{\epsilon} \bmod p$, where $\tau = E_1 - E' \bmod q$ and $\epsilon = S' - S_1 \bmod q$. Analogously, the signature (E', S') could be produced from the triple (E_2, R_2, S_2), if the values $\tau = E_2 - E' \bmod q$ and $\epsilon = S' - S_2$ are selected at step 4 of the protocol. Since during the protocol the values ϵ and τ are selected at random the signature could be produced from each of two considered triples as well as from each of the triple in the data base, i.e. the anonymity is provided by the proposed protocol.

4 Multi-signature Protocol for Simultaneous Signing a Package of Contracts

Due to fact that individual shares of the collective DS formed with the protocols described above are valid only in the frame of the given set of m signers the mentioned protocols can be used to solve efficiently the problem of simultaneous signing a contract. However they do not provide efficient solution of the problem of simultaneous signing a package of contracts. The last problem considers the cases when the first subset of some signers should sign the first document, the second subset should sign the second document, the third subset should sign the third document, and so on. Besides, all documents should be signed simultaneously. Since in such problem we have different documents and different hash functions corresponding to the respective documents, the described above collective DS protocols are not applicable to solve the problem. However using the idea of the collective DS protocols it is possible to propose the analogous multi-signature protocol that provides the solution. Such protocol, called composite DS protocol, uses the collective public key dependent on the set of documents to be signed.

Suppose the parameters p, q, and g as well as the secret key x and the public key $Y = g^x \bmod p$ are specified as in the protocol presented in Subsection 2.2. Suppose the m users should sign m messages M_i, $i = 1, 2, \cdots, m$, where to some subsets of the values i correspond the same messages. For example, if signers $\alpha_1, \alpha_2, \cdots, \alpha_{m'}$ are to sign the document M, then $M_{\alpha_1} = M_{\alpha_2} = \cdots = M_{\alpha_{m'}} = M$.

For implementing the composite DS protocol there is used the basic signature scheme characterized in using the

document-dependent public keys Y_h that are computed from the source public keys Y as follows $Y_h = Y^h \bmod p$, where h is the hash value computed from the document to be signed, i. e. $h = F_H(M)$. Using the signature (E, S) the signature verification in the underlying scheme is performed as follows:

1) It is computed the hash value h from the document M to which the current signature corresponds: $h = F_H(M)$, where $F_H(M)$ is some specified hash function.

2) Using the source public key of the signer it is computed the document-dependent public key $Y_h = Y^h \bmod p$ and the value $R = Y_h^E g^S \bmod p$;

3) It is computed the compressed value E^* from the value R: $E^* = f(R)$, where f is some compression function;

4) The value E^* is compared with E. If $E^* = E$, then the signature is valid.

The signature generation in the underlying scheme is as follows:

1) Generate a random number $t \le q-1$ and compute the values $R = g^t \bmod p$ and $E = f(R)$, where E is the first element of the signature;

2) Compute the second element of the signature: $S = t - xhE \bmod q$.

The composite DS protocol looks as follows.

1) Each ith signer selects at random some value $t_i < q$ and computes the randomization factor $R_i = g^{t_i} \bmod p$, where $i = 1, 2, \cdots, m$.

2) It is computed the common randomization parameter R: $R = R_1 R_2 R_3 \cdots R_m \bmod p$.

3) The first element E of the composite DS is computed using the formula $E = f(R)$, where f is some compression function, for example, $f(R) = R \bmod q$.

4) Each of the users computes his share in the composite DS as follows: $S_i = t_i - Eh_i x_i \bmod q$, where x_i is the secret key of the ith user.

5) The second element S of the composite DS is computed as the following sum: $S = S_1 + S_2 + \cdots + S_m \bmod q$.

The verification procedure of the composite DS is as follows.

1) Compute the composite public key Y as the product of all data-dependent keys of the signers: $Y = \prod_{i=1}^m Y_i^{h_i} \bmod p$, where h_i is the hash function value computed from the ith document and $Y_i = g^{x_i} \bmod p$ is the source public key of the ith signer.

2) Compute the values $R^* = Y^E g^S \bmod p$ and $E^* = f(R^*)$.

3) Compare E and E^*. If $E^* = E$, then the composite DS is valid.

The correctness of the composite DS is proved as follows:

$$\begin{aligned} R^* &\equiv Y^E g^S \equiv Y^E g^{\sum_{i=1}^m S_i} \equiv Y^E g^{\sum_{i=1}^m (t_i - Eh_i x_i)} \\ &\equiv Y^E \left(\prod_{i=1}^m g^{t_i}\right) \left(\prod_{i=1}^m g^{h_i x_i}\right)^{-e} \\ &\equiv Y^E R y^{-E} \equiv R \bmod p \\ \Rightarrow E^* &= f(R^*) = f(R) = E. \end{aligned}$$

Any successful attack of the first type considered in Section 2.3, which breaks the proposed composite DS protocol, also breaks the underlying DS algorithm. Suppose $m-1$ signers can forge the composite DS E^*, S^* such that there are satisfied the following expressions $R^* = y^{E^*} g^{S^*} \bmod p$ and $E^* = f(R^*)$. In this case we have

$$\begin{aligned} R^* &\equiv Y^{E^*} g^{S^*} \equiv (Y^* Y_m)^{E^*} g^{S^*} \\ &\equiv Y^{*E^*} Y_m^{E^*} g^{S^*} \equiv g^{*E^* \sum_{i=1}^{m-1} h_i x_i} Y_m^{E^*} g^{S^*} \\ &\equiv Y_m^{E^*} g^{S^* + E^* \sum_{i=1}^{m-1} h_i x_i} \bmod p \\ \Rightarrow R^* &= Y_m^{-E^*} g^{S^{**}}, \end{aligned}$$

where $S^{**} = \left(S^* + E^* \sum_{i=1}^{m-1} h_i x_i\right) \bmod p$. This means that collective forgery have computed the signature (E^*, S^{**}) which is a valid signature (to message M) of the mth signer, since the pair of numbers (E^*, S^{**}) satisfies the verification procedure of the underlying DS scheme. Thus, any successful attack breaking the collective DS protocol also breaks the underlying DS algorithm.

Computing the secret key of the mth signer by the $m-1$ signers sharing a composite signature with the mth signer requires solving the discrete logarithm problem. This can be illustrated like in the case of the collective DS protocol based on Schnorr's signature scheme (see Section 2.3).

One can propose some scenario of practical application of the blind composite DS protocols, which justifies interest to such protocols, however we have not succeeded to construct such protocol using the composite DS scheme described in this section.

5 Conclusion

A new multi-signature scheme called collective DS protocol have been constructed using the difficulty of the discrete logarithm problem. Providing the 80-bit security the protocol produces 320 bit signature notifying that m indicated signers ($m = 1, 2, 3, \cdots$) have signed an electronic message. Then the designed protocol has been modified into the blind collective DS protocol. The attractive feature of the proposed protocols is the simultaneous procedure of the signature generation. Therefore

they are efficient as protocols of simultaneous signing contracts. The composite signature protocol can be applied for solving the problem of simultaneous signing a package of different contracts by different sets of signers.

It seems that the blind collective DS scheme is attractive for application in the electronic money systems in which the electronic banknotes are issued by several banks.

Using the general construction of three proposed protocols, one can design the collective, blind collective, and composite protocols applying computations on elliptic curves (EC) [5, 8, 13]. The EC-based implementation of the protocols will provide higher performance for given security value. In future research it is also interesting to develop analogous collective and blind collective DS protocols using the DS algorithms recommended by official standards [2, 4] as the underlying signature generation procedure.

The composite DS protocol uses specific signature verification procedure, therefore it is out of the implementation based on the known standards until new design ideas will be applied. Designing the composite DS schemes based on the known standards is an open problem at present.

Designing a blind composite DS protocol remains another open problem.

Acknowledgments

The work supported by Russian Foundation for Basic Research grant # 08-07-00096-a.

References

[1] A. Boldyreva, "Efficient threshold signature, multisignature and blind signature shemes based on the gap-Diffie-Hellman-group signature scheme", *LNCS*, vol. 2139, pp. 31-46, Springer-Verlag, 2003.

[2] GOST R 34.10-2001, Russian Federation Standard, *Information Technology. Cryptographic Data Security. Produce and Check Procedures of Electronic Digital Signature*, Government Committee of the Russia for Standards, 2001 (in Russian).

[3] M. S. Hwang and C. C. Lee, "Research issues and challenges for multiple digital signatures", *International Journal of Network Security*, vol. 1. no 1, pp. 1-7, 2005.

[4] International Standard ISO/IEC 14888-3:2006(E), *Information Technology – Security Techniques – Digital Signatures with Appendix – Part 3: Discrete Logarithm Based Mechanisms*, 2006.

[5] B. King, "Mapping an arbitrary message to an elliptic curve when defined over $GF(2^n)$," *International Journal of Network Security*, vol. 8, no. 2, pp. 169-176, 2009.

[6] N. Koblitz and A. J. Menezes, "Another look at provable security", *Journal of Cryptology*, vol. 20, pp. 3-38, 2007.

[7] J. Li and S. Wang, "New efficient proxy blind signature scheme using verifiable self-certified public key," *International Journal of Network Security*, vol. 4, no. 2, pp. 193-200, 2007.

[8] T. C. Lin, "Algorithms on elliptic curves over fields of characteristic two with non-adjacent forms," *International Journal of Network Security*, vol. 9, no. 2, pp. 117-120, 2009.

[9] N. A. Moldovyan, "Digital signature scheme based on a new hard problem", *Computer Science Journal of Moldova*, vol. 16, no 2, pp. 163-182, 2008.

[10] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures", *Journal of Cryptology*, vol. 13, pp. 361-396, 2000.

[11] B. Schneier, *Applied Cryptography*, Second Edition, John Wiley & Sons, Inc. New York, 1996.

[12] C. P. Schnorr, "Efficient signature generation by smart cards", *Journal of Cryptology*, vol. 4, pp. 161-174, 1991.

[13] Z. J. Shi and H. Yun, "Software implementations of elliptic curve cryptography", *International Journal of Network Security*, vol. 7, no 1, pp. 141-150, 2008.

[14] G.K. Verma, "A proxy blind signature scheme over braid groups", *International Journal of Network Security*, vol. 9, no 3, pp. 214-217, 2009.

[15] G. K. Verma, "Probable security proof of a blind signature scheme over braid groups," *International Journal of Network Security*, vol. 12, no. 1, pp. 21-23, 2011.

[16] Z. M. Zhao, "ID-based weak blind signature from bilinear pairings," *International Journal of Network Security*, vol. 7, no. 2, pp. 265-268, 2008.

Appendix 1

Below we use the following terms and notations:

The kth residue (non-residue) mod$p$ is a value a such that congruence $x^k \equiv a \bmod p$ has solutions (no solution).

kR_p is the set of the kth residues mod$p$;

kNR_p is the set of the kth non-residues mod$p$;

$[\sqrt{k}]$ means the integer part of $\sqrt{k}$;

$\omega_p(a)$ denotes the order of the element a modulo p;

$\varphi(n)$ is Euler phi function of n.

The following three facts are well known from elementary number theory:

1) There exist $\frac{p-1}{k}$ different values $a_j \in \mathrm{kNR}_p$, where $j = 1, 2, \cdots, \frac{p-1}{k}$, each of which is the kth residue.

2) For some $a \in \mathrm{kR}_p$ it holds $a^{\frac{p-1}{k}} \equiv 1 \bmod p$.

3) For some value $b_i \in \mathrm{kNR}_p$ the congruence

$$b_i^{\frac{p-1}{k}} \equiv e_i \bmod p,$$

where $e_i = \sqrt[k]{1} \bmod p \neq 1$ and $i = 1, 2, \cdots, k-1$, holds.

Using these facts, it is easy to show that each of the roots e_i defines exactly $\frac{p-1}{k}$ different values b_{ij}, where $j = 1, 2, \cdots, \frac{p-1}{k}$, such that $b_{ij}^{\frac{p-1}{k}} \equiv e_i \bmod p$. Indeed [9],

$$\begin{aligned} b_{ij}^{\frac{p-1}{k}} &\equiv b_{ij'}^{\frac{p-1}{k}} \bmod p \\ \Rightarrow \left(\frac{b_{ij}}{b_{ij'}}\right)^{\frac{p-1}{k}} &\equiv 1 \bmod p \\ \Rightarrow \frac{b_{ij}}{b_{ij'}} \bmod p &= a_{j''}, \end{aligned}$$

i. e. the ratio $\frac{b_{ij}}{b_{ij'}} \bmod p$ is the kth residue. There exist exactly $\frac{p-1}{k}$ different values $a_{j''}$, hence there exist exactly $\frac{p-1}{k}$ different values $b_{ij'}$. Therefore selecting at random a value t we have probabilities

$$\Pr\left(t^{\frac{p-1}{k}} \bmod p = 1\right) = \Pr\left(t^{\frac{p-1}{k}} \bmod p = e_i\right)$$

for all $i = 1, 2, \cdots, k-1$. This fact is used while estimating the complexity of the algorithm described below.

Taking into account that for each i there exists i' such that $e_{i'} = e_i^{-1} \bmod p$ we can write $a^{\frac{p-1}{k^2}} e_{i'} \equiv 1 \bmod p$, therefore

$$a^{\frac{p-1}{k^2}} b^{\frac{p-1}{k}} \equiv a^N b^{\frac{p-1}{k}} \equiv 1 \bmod p, \tag{3}$$

where $b \in \mathrm{kNR}_p$.

If Congruence (3) is fulfilled, then we can easily calculate a root $\sqrt[k]{a} \bmod p$. Indeed, Congruence (3) can be represented as

$$a^k b^{\frac{p-1}{k^2} k} \equiv a^{k-N} \bmod p, \tag{4}$$

where with sufficiently high probability we have $\gcd(k - N, p-1) = 1$. Suppose that the last relation holds (in other case the problem is only a bit more complex). Then it is possible to compute value $N' = (k-N)^{-1} \bmod p - 1$. Therefore we get $a^{N'k} b^{N' \frac{p-1}{k^2} k} \equiv a \bmod p$, hence

$$\left(a^{N'} b^{N' \frac{p-1}{k^2}}\right)^k \equiv a \bmod p. \tag{5}$$

Congruence (5) shows that value $X = a^{N'} b^{N' \frac{p-1}{k^2}} \bmod p$ represents one of roots $\sqrt[k]{a} \bmod p$. Other $k-1$ roots $\sqrt[k]{a} \bmod p$ can be computed using the formula $e_i X \bmod p$, $i = 1, 2, \cdots, k-1$ (roots $\sqrt[k]{1} \bmod p$ can be find computing the sequence $\{\epsilon, \epsilon^2 \bmod p, \cdots, \epsilon^{k-1} \bmod p, \epsilon^k \bmod p = 1\}$, where ϵ is the kth order element modulo p).

A value $b \in \mathrm{kNR}_p$ satisfying Congruence (3) can be computed as follows. The value b can be represented as $b = b_i b_j \bmod p$, where $b_i, b_j \in \mathrm{kNR}_p$:

$$\begin{aligned} a^{\frac{p-1}{k^2}} b_i^{\frac{p-1}{k}} b_j^{\frac{p-1}{k}} &\equiv 1 \bmod p \\ \Rightarrow a^{\frac{p-1}{k^2}} b_i^{\frac{p-1}{k}} &\equiv b_j^{-\frac{p-1}{k}} \bmod p. \end{aligned} \tag{6}$$

The required values b_i and b_j can be found with high probability as follows [9]:

1) Select at random a value b_i and calculate the value $A_i = a^{\frac{p-1}{k^2}} b_i^{\frac{p-1}{k}} \bmod p$. Construct a table with entries (A_i, b_i) for $i = 1, 2, \cdots, [\sqrt{k}] + \Delta$, where $\Delta \ll [\sqrt{k}]$. Complexity of this step is $O(\sqrt{k})$ exponentiation operations.

2) Select at random a value b_j and calculate the value $B_j = b_j^{-\frac{p-1}{k}} \bmod p$. Construct a table with entries (B_j, b_j) for $j = 1, 2, \cdots, [\sqrt{k}] + \Delta$, where $\Delta \ll [\sqrt{k}]$. Complexity of the second step $O(\sqrt{k})$ exponentiation operations.

3) Sort the first table by component A_i. Complexity of this step is $O(\sqrt{k} \cdot |k|)$ comparison operations.

4) For $j = 1$ to $[\sqrt{k}] + \Delta$ check if the value B_j is equal to the value of the first component of some entry in the first table. Complexity of this step is $O(\sqrt{k} \cdot |k|)$ comparison operations.

This algorithm requires storage for about $4\sqrt{k}$ (i. e. $O(\sqrt{k})$) $|p|$-bit numbers. For randomly selected b_i and b_j we have $\Pr(A_i = B_j) = k^{-1}$, therefore in two tables each of which contains $\sqrt{k} + \Delta$ random values with probability more than 0.5 there are equal values $A_{i_0} = B_{j_0}$ (see birthday paradox [11]). Thus, with probability about 0.5 the algorithm finds values b_{i_0} and b_{j_0} satisfying Congruence (6). Having such values we can easily compute the value $b = b_{i_0} b_{j_0} \bmod p$ satisfying Congruence (3) and then compute $X = \sqrt[k]{a} \bmod p$. On the whole complexity of the algorithm can be estimated as $\approx 2\sqrt{k}$ modulo exponentiation operations. Trying the algorithm several times we will get value X with probability close to 1. Difficulty of this procedure is $W = O(\sqrt{k})$. If $|k| = 160$, then $W \approx 2^{80}$ exponentiation operations.

Appendix 2

In the case of sufficiently small size of the value $p = Nk^2 + 1$ the kth roots from the public key Y can be computed by means of finding discrete logarithm as follows.

1) Generate a primitive element g modulo p.

2) Calculate logarithm $\log_g Y \bmod p$.

3) Divide $\log_g Y \bmod p$ by k (at this step it is get the value $\log_g \sqrt[k]{Y} \bmod p$; note that logarithm from Y is multiple to the value k).

4) Raise the number g to the power $z = \log_g \sqrt[k]{Y} \bmod p$ and get the value $\sqrt[k]{Y} = g^{\log_g z} \bmod p$.

Let us justify the division operation that is performed at step 3. The public key Y is computed as $Y = X^k \bmod p$. The last expression can be represented as follows

$$\left(g^{\log_g X}\right)^k \equiv g^{k \cdot \log_g x} \equiv Y \equiv g^{\log_g Y} \bmod p,$$

i. e. k divides $\log_g Y$. For values $|p| \approx 1024$ bits difficulty of finding logarithms is approximately equal to 2^{80} operations [4].

Nikolay A. Moldovyan is an honored inventor of Russian Federation (2002), a laboratory head at St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences, and a Professor with the St. Petersburg State Electrotechnical University. His research interests include computer security and cryptography. He has authored or co-authored more than 60 inventions and 220 scientific articles, books, and reports. He received his Ph.D. from the Academy of Sciences of Moldova (1981). Contact him at: nmold@mail.ru.

Alexander A. Moldovyan head of the Information Security Problems Department at St. Petersburg Institute for Informatics and Automation of Russian Academy of Sciences and a Professor with the St. Petersburg State Electrotechnical University. His research interests include information security and cryptographic protocols. He has authored or co-authored more than 40 inventions and 180 scientific articles, books, and reports. He received his Ph.D. from the St. Petersburg State Electrotechnical University (1996). Contact him at: maa1305@yandex.ru.