Computer Science Journal of Moldova, vol.**16**, no.2(47), 2008

Digital Signature Scheme Based on a New Hard Problem[*]

Nikolay A. Moldovyan

Abstract

Factorizing composite number $n = qr$, where q and r are two large primes, and finding discrete logarithm modulo large prime number p are two difficult computational problems which are usually put into the base of different digital signature schemes (DSSes). This paper introduces a new hard computational problem that consists in finding the kth roots modulo large prime $p = Nk^2 + 1$, where N is an even number and k is a prime with the length $|k| \geq 160$. Difficulty of the last problem is estimated as $O(\sqrt{k})$. It is proposed a new DSS with the public key $y = x^k \bmod p$, where x is the private key. The signature corresponding to some message M represents a pair of the $|p|$-bit numbers S and R calculated as follows: $R = t^k \bmod p$ and $S = tx^{f(R,M)} \bmod p$, where $f(R,M)$ is a compression function. The verification equation is $S^k \bmod p = y^{f(R,M)}R \bmod p$. The DSS is used to implement an efficient protocol for generating collective digital signatures.

1 Introduction

Information authentication in computer networks and information systems is usually performed with digital signature schemes (DSSes) that are attributed to public key cryptosystems. The DSSes are based on some well investigated hard computational problems. The upper boundary of the DSS security level is defined by the difficulty of the

*This work was supported by Russian Foundation for Basic Research grants # 08-07-00096-a and # 08-07-90100-Mol_a.

used hard problem. To get sufficiently high security the signature generation and signature verification procedures use calculations modulo comparatively large numbers. The modulus length defines significantly performance of the DSSes.

The most efficient known DSSes are based on the following three difficult problems [17]:

1. Factorization of a composite number $n = qr$, where q and r are two large primes.

2. Finding discrete logarithm modulo large prime number p.

3. Finding discrete logarithm in a group of points of some elliptic curve.

The indicated problems are hard, if the used primes and elliptic curves satisfy special requirements [9, 17]. The first problem is used in the following cryptosystems: RSA [19], Rabin's DSS [18], and in DSSes proposed in [14, 15]. The second problem is used in ElGamal's DSS [4], Schnorr's DSS [24], American standard DSA [16], Russian standard GOST R 34.10-94 [5], and in some DSSes presented in [11, 14]. The third problem is used, for example, in American standard ECDSA [1] and Russian standard GOST R 34-2001 [6].

In general the security level of the DSS can be estimated as number of group operations required to forge a signature. In this paper the performance is compared for different DSS in the case of minimum security level that can be estimated at present as 2^{80} modulo exponentiation operations. Solving the difficult problem that is put into the base of some considered DSS allows one to calculate signatures corresponding to arbitrary messages. Therefore the security is less or equal to the difficulty of the hard problem that is put into base of the DSS.

In the best case the DSS is as secure as difficulty of the used computational problem. If for some DSS we can prove the last fact, then such DSS is called provably secure. In literature the formal proof of the security level is presented for Rabin's DSS [18, 22] and for a class of provably secure DSSes which generalize the Rabin's cryptosystem [12].

However these provably secure DSSes have not gained wide practical application because of their comparatively low performance.

The best known algorithms for solving the first two problems have subexponential complexity [10]. Therefore in the case of the RSA and Rubin's DSS the minimum length of the value n is 1024 bits. In these DSSes the value n is used as modulus while performing computations corresponding to signature generation and verification procedures. Respectively, to provide the minimum security level the ElGamal's DSS, Schnorr's DSS, DSA, and GOST R 34.10-94 should use computations modulo 1024-bit prime number.

The best known algorithms for solving the third hard problem has exponential complexity (for special class of elliptic curves) and its hardness is estimated as $O(\sqrt{q})$ operations of multiplication of points, where q is a prime order of the group of points of the considered elliptic curve ($O(\cdot)$ is the order notation [3]). Operations performed on points include computations modulo prime p such that $|p| \approx |q|$, where $|x|$ denotes the bit length of the value x. Due to exponential dependence of the hardness of the discrete logarithm problem in a group of points of elliptic curves the minimum security level can be provided using the modulus p having sufficiently small length ($|p| \geq 160$ bits). Therefore the performance of the DSSes based on elliptic curves is higher against mentioned above DSSes, addition of two points reqire perfoming several multiplications modulo p and some auxiliary operations though.

In present paper we consider in detail the approach to designing DSSes which has been proposed earlier in our patent application [13]. The approach is based on using a new hard computational problem. The rest of the paper is organised as follows. In Section 2 we show that in particular cases finding the kth roots modulo large prime number is a hard problem. Such cases correspond to prime modulus with the structure $p = Nk^s + 1$, where N is an even number, k is a large prime, and $s \geq 2$. Difficulty of this new hard problem is estimated as $O(\sqrt{k})$. In Section 3 we introduce new DSS and estimate that the minimum security is obtained with the length of $|k| \geq 160$ bits and the modulus length of $|p| \geq 1024$ bits. We also describe a modified DSS providing reduction of the signature length and show that the proposed algorithms are efficient to implement protocol for generating collective digital signatures. Section 4 concludes the paper.

Below we use the following terms.

The kth power residue modulo p is a value a for which the congruence $x^k \equiv a \bmod p$ has solutions.

The kth power non-residue modulo p is a value b for which the congruence $x^k \equiv b \bmod p$ has no solution.

These terms generalize the well known terms *quadratic residue* and *quadratic non-residue* [3, 9, 10].

Also we use the following notations:

kR_p is the set of the kth residues modulo p;

kNR_p is the set of the kth non-residues modulo p;

$|x|$ denotes the length of the binary representation of the value x;

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

$||$ is the concatenation operation;

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

$\#\{*\}$ denotes the number of elements in the set $\{*\}$;

$\{a : **\}$ denotes a set of elements a possessing the property $**$;

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

2 A new hard computational problem

2.1 Computing roots modulo prime

Difficulty of finding roots modulo a composite number is used in some of the known DSSes: RSA, Rabin's DSS, and others [11]. Indeed, the RSA [19, 10] is based on calculations modulo n that is a product of two randomly chosen prime numbers r and q: $n = rq$. In RSA the public key represents a pair of numbers (e, n). The signature corresponding to some message M is a value S, which satisfies the following verification equation: $S^e \bmod n = H$, where H is the hash function value corresponding to M. To generate a valid signature one should calculate the eth root modulo n. This problem is difficult untill the composite number n is factorized. The owner of the public key knows the related private key that is a number d, which is inverse of e modulo $\varphi(n)$, where

$\varphi(n) = (r-1)(q-1)$. Thus, we have $ed \bmod \varphi(n) = 1$. The signature generation is performed as follows: $S = H^d \bmod n$. Security of the RSA is based on difficulty of calculating d while φ(n) is an unknown value. The $\varphi(n)$ value can be easily calculated after factorization of the modulus n, therefore divisors of n have to be kept in secret. Thus, in the case of the RSA the problem of finding roots is dependent on the factorization problem.

The signature verification equation in Rabin's cryptosystems can be described as follows:

$$S^2 \bmod n = H||R,$$

where pair of numbers (S, R) is signature and H is the hash function value corresponding to the signed message M. The second element in the signature is the randomization value. To generate a signature one should select at random such R that value $H||R$ is quadratic residue modulo n and calculate quadratic root $\sqrt{H||R} \bmod n$. The last represents a difficult problem until the value n is factorized. If the divisors q and r are known, then the roots $\sqrt{H||R} \bmod q$ and $\sqrt{H||R} \bmod r$ can be easily calculated [22]. Then, using the Chinese Remainder Theorem, one can find the minimum value S such that $S \equiv \sqrt{H||R} \bmod q$ and $S \equiv \sqrt{H||R} \bmod r$, i. e. $S = \sqrt{H||R} \bmod n$. We have two different roots modulo q and two different roots modulo r, therefore we have four different roots modulo n. Each of the lasts satisfies the signature verification equation. Thus, finding square roots modulo n is a difficult problem that also depends on the factorization problem.

The main difference between the RSA and Rabin's DSS consists in the following. In RSA we have $\gcd(e, \varphi(n)) = 1$, $(\gcd(e, q-1) = 1$ and $\gcd(e, r-1) = 1)$, but in Rabin's DSS $\gcd(2, q-1) \neq 1$ and $\gcd(2, r-1) \neq 1$. Actually, the fact that $2|q-1$ and $2|r-1$ requires to use some special algorithm to calculate the square roots. For some random prime p and large prime divisor $k|p-1$ with probability very close to 1 the complexity of finding k roots $\sqrt[k]{a} \bmod p$, where a is one of the kth power residues modulo p, is sufficiently low. Indeed, if prime k is sufficiently large, then with high probability k does not devide $\frac{p-1}{k}$ and it is easy to find some value Δ such that k divides $\frac{p-1}{k} + \Delta$, i. e.

$\frac{p-1}{k} + \Delta = hk$, where h is an integer (note that k does not divide Δ). Then we have:

$$a^{\frac{p-1}{k}} \equiv 1 \bmod p \Rightarrow a^{\frac{p-1}{k}+\Delta} \equiv a^{\Delta} \bmod p \Rightarrow a^{hk} \equiv a^{\Delta' d} \bmod p,$$

where $d = \gcd(\Delta, p-1)$. Let $\Delta'' = \Delta'^{-1} \bmod p-1$. Then we have:

$$\left(\left(a^{1/d}\right)^{h\Delta''}\right)^{k} \equiv a \bmod p \Rightarrow a^{1/k} \equiv \left(a^{1/d}\right)^{h\Delta''}.$$

With high probability the value d is sufficiently small and the dth root can be easily found, for example, using the method described in Section 2.2.

If $k^2|p-1$, then the method described above does not work, i. e. in the case of the prime $p = Nk^s + 1$, where N is an even number and $s \geq 2$, computing the kth roots is difficult. Below we estimate the computational complexity of this hard probem. The primes p having the indicated structure and different length $|p|$ can be easily generated for many different values s and $|N|$ (some examples are presented in Appendix 1).

2.2 Computing the kth roots modulo $p = Nk^s + 1$

The following three facts are well known [7, 20, 21]:

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

2. For some value $a \in \text{kR}_p$ the congruence $a^{\frac{p-1}{k}} \equiv 1 \bmod p$ holds.

3. For some value $b_i \in \text{kNR}_p$ the congruence

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

where $e_i = \sqrt[k]{1} \bmod p \neq 1$ and $i = 1, 2, ..., k-1$, 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 $j = 1, 2, ..., \frac{p-1}{k}$, such that $b_{ij}^{\frac{p-1}{k}} \equiv e_i \bmod p$. Indeed, we have

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

i. e. the ratio $\frac{b_{ij}}{b_{ij'}}$ mod p is the kth power 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 $i = 1, 2, ..., k-1$. This fact is used while estimating the complexity of the algorithm described below.

Suppose a random value $a \in \mathrm{kR}_p$ is selected. It is evident that $a^{\frac{p-1}{k^2}} \bmod p \in \{e_1, e_2, ..., e_{k-1}, 1\}$. Number of the elements $a \in \mathrm{kR}_p$ for which we have $a^{\frac{p-1}{k^2}} \bmod p = 1$ is equal to $Z_1 = \#\{a : \omega_p(a) \leq \frac{p-1}{k^2} = N\}$. Number of the values $a \in \mathrm{kR}_p$ for which we have $a^{\frac{p-1}{k^2}} \bmod p \neq 1$ is equal to $Z_2 = \#\{a : \omega_p(a) \leq \frac{p-1}{k} = Nk\}$ (see Theorem 2.10.6 in [3]). It is known (see Theorem 2.8.4 in [3]) that

$$\sum_{d|N, d>0} \varphi(d) = N \Rightarrow Z_1 = N \;\; \text{and} \;\; Z_1 = Nk.$$

Probability $\Pr\left(a^{\frac{p-1}{k^2}} \equiv 1 \bmod p\right) = \frac{Z_1}{Z_2} = k^{-1}$ is negligible. With probability $1 - k^{-1}$ we have $a^{\frac{p-1}{k^2}} \bmod p = e_i \neq 1$.

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

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

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

If congruence (1) is fulfilled, then we can easily calculate a root $\sqrt[k]{a} \bmod p$. Indeed, congruence (1) can be represented as

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

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

$$\left(a^{N'} b^{N' \frac{p-1}{k^2}} \right)^k \equiv a \bmod p. \qquad (3)$$

Congruence (3) 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 $e_i X \bmod p$, $i = 1, 2, ..., k-1$ (roots $\sqrt[k]{1} \bmod p$ can be found computing the sequence $\{\epsilon, \epsilon^2 \bmod p, ..., \epsilon^{k-1} \bmod p, \epsilon^k \bmod p = 1\}$, where ϵ is the kth order element modulo p).

A value $b \in \mathrm{kNR}_p$ satisfying congruence (1) can be computed as follows. The value b can be represented as $b = b_i b_j \bmod p$, where $b_i, b_j \in \mathrm{kNR}_p$:

$$a^{\frac{p-1}{k^2}} b_i^{\frac{p-1}{k}} b_j^{\frac{p-1}{k}} \equiv 1 \bmod p \Rightarrow$$

$$a^{\frac{p-1}{k^2}} b_i^{\frac{p-1}{k}} \equiv b_j^{-\frac{p-1}{k}} \bmod p. \qquad (4)$$

The following algorithm finds the required values b_i and b_j with high probability.

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 $i = 1, 2, ..., [\sqrt{k}] + \Delta$, 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 $j = 1, 2, ..., [\sqrt{k}] + \Delta$, where $\Delta \ll [\sqrt{k}]$. Complexity of the second step is $O(\sqrt{k})$ 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 [17, 10]). Thus, with probability about 0.5 the algorithm finds values b_{i_0} and b_{j_0} satisfying congruence (4). Having such values we can easily compute the value $b = b_{i_0} b_{j_0} \bmod p$ satisfying congruence (1) and then compute $X = \sqrt[k]{a} \bmod p$. On the whole the 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. Hardness of this procedure is $W = O(\sqrt{k})$. If $|k| = 160$, then $W \approx 2^{80}$ exponentiation operations. Thus, the hardness depends exponentially on the value $\sqrt{k}$ and in the case $|k| = 160$ the considered problem is computationally infeasible (untill a new significantly more efficient algorithm is developed).

2.3 Dependence on the discrete logarithm problem

In Section 2.2 we have proposed a direct algorithm for finding the kth roots which works for arbitrary large values $|p|$. In the case of sufficiently small size of the value p the kth roots can be computed by means of finding discrete logarithm as follows.

1. Generate a primitive element g modulo p.

2. Calculate logarithm $\log_g a \bmod p$.

3. Divide $\log_g a \bmod p$ by k (at this step it gets the value $\log_g \sqrt[k]{a}$ mod p; note that logarithm corresponding to some value $a \in \mathrm{kR}_p$ is multiple to the value k).

4. Exponentiate g to the power $z = \log_g \sqrt[k]{a} \bmod p$ and get the value $\sqrt[k]{a} = g^{\log_g z} \bmod p$.

Let us justify the division operation that is to be performed at step 3. If $a \in \mathrm{kR}_p$, then there exists some value x for which $x^k \equiv a \bmod p$ holds. The last expression can be represented as follows

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

i. e. $k|\log_g a$. For values $|p| \approx 1024$ bits difficulty of finding logarithms is approximately equal to 2^{80} operations [8]. Therefore in the case $p = Nk^2+1$, $|k| = 160$ bits, and $|N| < 704$ bits the algorithm described in this section is more efficient. If $|N| > 704$ bits, then the algorithm described in Section 2.2 is more efficient.

3 New digital signature scheme

3.1 Initial design

New hard computational problem described in Section 2 is used in the DSS described below. It uses the prime modulus having the structure $p = Nk^2 + 1$, where k is a large prime ($|k| \geq 160$) and N is such even number that $|p| \geq 1024$ bits. A random value x is selected as a private key. The public key y is computed using the formula $y = x^k \bmod p$. The signature represents a pair of $|p|$-bit numbers S and R.

Suppose a message M is given. The signature generation procedure is performed as follows:

1. Select at random a value $t < p-1$ and calculate the first element of the signature: $R = t^k \bmod p$.

2. Using some specified hash function $F_H(M)$ calculate the hash value H corresponding to the message M and compute $f(R, M) = RH \bmod \delta$, where δ is a large prime that is a part of the signature generation algorithm. For example, it is acceptable to use a randomly selected prime δ such that $|\delta| = 160$. The function $F_H(M)$ is also a part of the DSS. For example, one can use the hash function SHA-1 [17] recommended by US National Institute of Standards and technology (NIST).

3. Calculate the second element of the signature: $S = x^{f(R,M)}t \bmod p$.

The signature verification procedure is defined by the following congruence:

$$S^k \equiv y^{f(R,M)} R \bmod p.$$

The signature length is $|S| + |R| \approx 2|p|$.

Accordingly to the hard problem put into base of this DSS the direct way to attack this cryptosystem is to compute one of the kth roots from the public key (it is not necessary to guess exactly the value of the secret key x, since each of the values $xe_i \bmod p$ can be used to generate a valid signature).

The next direct attack is connected with computing one of the kth degree roots from the signature element R. Suppose the attacker has computed a root $\sqrt[k]{R} \bmod p = t'$, where the element R satisfies the condition $\gcd(f(R,M), p-1) = 1$ (the respective signature can be selected from several known valid signatures). Then the attacker can compute integer $f' = f^{-1}(R,M) \bmod p-1$. The value t' is connected with t: $t' = te_i \bmod p$, therefore he can compute the value $x' = (S/t')^{f'} \bmod p = (S/t)^{f'} e_i^{-f'} \bmod p = xe_i' \bmod p$ that is one of the kth roots from the secret key. These attacks are infeasible at present, if $|k| \geq 160$ bits.

Let us consider an attack that for some given values $R \in \mathrm{kR}_p$ and H allows one to generate a valid signature (R,S). Such attack can be easily used to find the kth roots from y using the following algorithm.

1. Select at random a value t $(0 < t < p)$.

2. Calculate $R = t^k \bmod p$, $E = f(R,M) = RH \bmod \delta$, and $\gcd(p-1, E)$. If $\gcd(p-1,E) \neq 1$, then go to step 1.

3. Using the attack generate the signature (R,S). Under our assumption the signature (R,S) satisfies the congruence $S^k \equiv y^E R \bmod p$.

4. Calculate the values $E' = E^{-1} \bmod p-1$ and $\sqrt[k]{y} = (S/t)^{E'} \bmod p$.

Thus, any attack like the considered one is as difficult as finding the kth roots modulo p (any value $a \in \mathrm{kR}_p$ can be used in the considered algorithm as the public key value y).

There are also possible some hypothetic attacks that for some given values $y \in \mathrm{kR}_p$, S, and H allow one to compute such value R that the signature (R, S) satisfies the verification congruence. This means that attacker is able to solve the congruences like $Ry^{f(R,M)} \equiv C \bmod p$, where C is a constant. However the difficulty of solving such congruences modulo large prime p relatively R is put into the base of the GOST R 34.10-94, ElGamal's, and Schnorr's DSSes.

3.2 Modified DSS

It is possible to reduce the signature length using the value $E = f(R, M)$ as signature element instead of the R element. A variant of the modified DSS is presented by the following verification procedure:

1. Using the signature (E, S) compute $R = S^k y^{-E} \bmod p$.
2. Calculate $E' = f(R, M) = RH \bmod \delta$.
3. Compare E' with E. If $E' = E$, then the signature is valid.

In the modified DSS the signature length is equal to $|E| + |S| = |\delta| + |S| \approx |p|$. A numerical illustration of the modified DSS is presented in Appendix 2.

The straightforward attacks mentioned in Section 2.1 are also applicable to the modified DSS. Let us consider another attack that is efficient in the case of small size of the compression function $f(R, M)$, for example in the case of small value $|\delta|$. It can be performed as the following algorithm.

1. Select at random the values $E < \delta$ and $S < p$.
2. Compute value $R' = S^k y^{-E} \bmod p$.
3. Calculate value $E' = f(R', M)$.
4. Compare E' and E. If $E' \neq E$, then jump to step 1.

On the average the work effort of this algorithm is $\approx 2\delta$ exponentiation operations, since $\Pr(E' = E) = \delta^{-1}$. This attack is infeasible at present, if $|f(R, M)| \geq 160$ bits.

3.3 Collective signature protocol

Suppose the jth user owns the public key y_j depending on his private key $x_j < p$ as follows: $y_j = x_j^k \bmod p$, where $j = 1, 2, ..., \mu$. Suppose an electronic document M is given and m $(m \leq \mu)$ users owning the public keys y_{α_1}, $y_{\alpha_2}, ...,$ y_{α_m} should sign it simultaneusly. The following protocol produces a collective digital signature (CDS) and solves the indicated problem more efficiently than the known protocols for signing simultaneously a contract [23].

1. Each α_ith user selects at random a value $t_{\alpha_i} < p$ and computes the public value $R_{\alpha_i} = t_{\alpha_i}^k \bmod p$, where $i = 1, 2, ..., m$.

2. Some of the users (or one of them) calculate the common randomization value

$$R = R_{\alpha_1} R_{\alpha_2} ... R_{\alpha_m} \bmod p$$

and then calculate the first part of the CDS $E = f(R, M)$, where f is a specified compression function. For example, we will use the following function $E = RH \bmod \delta$, where δ is a large prime having length $|\delta| = 160$ and H is a hash value computed from the message M.

3. Using the values R and t_{α_i} each α_ith user computes its share in the CDS:

$$S_{\alpha_i} = x_{\alpha_i}^E t_{\alpha_i} \bmod p$$

that is supposed to be available to all users of the group.

4. Calculate the second element of the CDS:

$$S = S_{\alpha_1} S_{\alpha_2} ... S_{\alpha_m} \bmod p.$$

Thus, the CDS is computed with $2m$ modulo exponentiations. The CDS length is fixed and equals to $|S| + |\delta|$.

The CDS verification procedure is performed as follows.

1. Compute the collective public key y: $y = y_{\alpha_1} y_{\alpha_2} ... y_{\alpha_m} \bmod p$.

2. Using the CDS (E, S) compute value R':

$$R' = S^k y^{-E} \bmod p.$$

3. Compute $E' = R'H \bmod \delta$.

4. Compare values E' and E. If $E' = E$, then the signature is valid. Otherwise the signature is false.

The proposed algorithm can be used in the collective signature protocols that are free of the participation of any trusted party that is needed in the multisignature protocol proposed in [2] for implementing the fixed size signature corresponding to some group of users. In our algorithm we need no declosure of any user's private key, while generating the collective digital signature with fixed size.

4 Conclusion

A new hard computational problem have been proposed as cryptographic primitive. The problem consists in finding the kth roots modulo large prime $p = Nk^s + 1$, where k is a large prime such that $k \geq 160$ bits and N is an even number such that $|N| + s|k| \geq 1024$ bits. We have proposed an algorithm for solving this problem. The complexity W of the algorithm depends exponentially on $|k|$ and is described with the formula $W = O(\sqrt{k})$.

Using a novel hard computational problem a new DSS has been proposed. The proposed DSS has been used to design a collective digital signature protocol that produces the fixed size signature shared by arbitrary number of users $m \geq 1$.

We hope that this paper will initiate other attempts aimed to developing new efficient algorithms for solving the proposed computational problem. Such researches will contribute to the security estimation of the presented new DSS and to evaluation of the proposed computational problem as a new cryptographic primitive.

References

[1] ANSI X9.62 and FIPS 186-2. *Elliptic curve signature algorithm*, 1998

[2] A.Boldyreva. *Efficient Threshold Signature, Multisignature and Blind Signature Schemes Based on the Gap-Diffi-Hellman-Group Signature Scheme.* Springer-Verlag LNCS, 2003, vol. 2139, pp. 31–46.

[3] J. Buchmann. *Introduction to Cryptography.* Springer-Verlag. Berlin, 2003. - 335 p.

[4] ElGamal T. *A public key cryptosystem and a signature scheme based on discrete logarithms.* IEEE Transactions on Information Theory. 1985, Vol. IT-31, No. 4. pp.469–472.

[5] GOST R 34.10-94. *Russian Federation Standard. Information Technology. Cryptographic data Security. Produce and check procedures of Electronic Digital Signature based on Asymmetric Cryptographic Algorithm.* Government Committee of the Russia for Standards, 1994 (in Russian).

[6] 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).

[7] L.H.Hua. *Introduction to Number Theory.* Springer, Berlin, Heidelberg, New York, 1982.

[8] International Standard ISO/IEC 14888-3:2006(E). *Information technology – Security techniques – Digital Signatures with appendix – Part 3: Discrete logarithm based mechanisms.*

[9] N. Koblitz. *A Course in Number Theory and Cryptography.* Springer-Verlag. Berlin, 2003. - 236 p.

[10] A.J.Menezes, P.C. Van Oorschot, and S.A.Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, FL, 1997.

[11] A.A.Moldovyan, D.N.Moldovyan, L.V.Gortinskaya. *Cryptoschemes Based on New Signature Formation Mechanism.* Computer Science Journal of Moldova. 2006.vol. 14. No 3 (42). pp. 397–411.

[12] N.A.Moldovyan, A.A.Moldovyan. *Class of Provably Secure Information Authentication Systems.* 4th Int. Workshop MMM-ANCS'07 Proc. September 13-15, 2007, St. Petersburg, Russia. Springer Verlag CCIS. 2007. vol. 1, pp. 147–152.

[13] N.A.Moldovyan, A.A.Moldovyan. *A Method for Generating and Verifying Electronic Digital Signature Certifying an Electronic Document.* Russian patent application # 2007129254, July 30, 2007.

[14] N.A.Moldovyan. *New Public Key Cryptosystems Based on Difficulty of Factorization and Discrete Logarithm problems.* 3d Int. Workshop IF&GIS'07 Proc. May 28-29, 2007, St.Petersburg, Russia. Springer LNGC. 2007. Vol. XIV. pp. 160–172.

[15] N.A.Moldovyan. *Cryptoschemes Based on New Signature Formation Mechanism.* Computer Science Journal of Moldova. 2006. vol. 14. No 3 (42). pp. 390–396.

[16] National Institute of Standards and Technology, NIST FIPS PUB 186. Digital Signature Standard, U.S. Department of Commerce, 1994.

[17] Pieprzyk J., Hardjono Th., and Seberry J. *Fundamentals of Computer Security.* Springer-Verlag. Berlin, 2003. - 677 p.

[18] Rabin M.O. *Digitalized Signatures and Public Key Functions as Intractable as Factorization.* - Technical report MIT/LCS/TR-212, MIT Laboratory for Computer Science, 1979.

[19] R.L.Rivest, A.Shamir, and L.M.Adleman. *A Method for Obtaining Digital Signatures and Public Key Cryptosystems.* Communications of the ACM, 1978, vol. 21, no 2, pp. 120–126.

[20] H.E.Rose. *A Course in Number Theory.* 2nd ed. Oxford: Clarendon Press, 1994.

[21] K.H.Rosen. *Elementary Number Theory and its Applications.* 4th ed. Reading, MA: Addison-Wesley Publishing Company, 2000.

[22] N.Smart. *Cryptography: an Introduction. McGraw-Hill Publication.* London, 2003.

[23] B. Schneier. *Applied Cryptography.* Second Edition, John Wiley & Sons, Inc. New York, 1996. 758 p.

[24] Schnorr C.P. *Efficient signature generation by smart cards.* J. Cryptology. 1991. vol. 4. pp. 161–174.

Appendix 1.

Examples of different primes having the structure $p = Nk^s + 1$.

We have composed a computer program generating the primes $p = Nk^s + 1$, where k is a prime. Our experiment has shown that for some pairs of the values s and N there exist no primes k such that $p = Nk^s + 1$ is a prime. For example, there exist no primes p for even s and $N \equiv 2 \bmod 6$. Another interesting example of the negative cases is presented by the pair $s = 4$ and $N = 4$. For odd values s there are also some exceptions.

However primes p having different length exist in the majority of the cases. The experiment has shown that there exist sufficiently large portion of the primes p (such that $2 \leq |k| \leq 320$ bits and $5 \leq |p| \leq 2048$ bits) for even s and $N \equiv Z \bmod 6$, where $Z = 0$ or $Z = 4$. For odd values s we also have many positive cases corresponding to different values N. For example, we have the following primes p.

Case $N = 10$, $s = 2$, and $k = 725884693$ gives prime:
$p = 5269085875317042491$.

Case $N = 2$, $s = 3$, and $k = 1337274629$ gives prime:
$p = 4782905620790796691857520379$.

Case $N = 4$, $s = 2$, and $k = 67433803869294133$ gives prime:
$p = 18189271617129713532911550672886757$.

Case $N = 4$, $s = 2$, and $k = 53421592407668514126913$3 gives prime:
$p = 1141546614148426492824805789757003540711754286757$.

Case $N = 10000$, $s = 4$, and $k = 232667804209296877$ gives prime:
$p = 29305232068834704658806727836467329654824325853306769179650044728906410001$.

Case $N = 10$, $s = 2$, and
$k =$
210992170315015669831165407048312242755193421562671721912463
gives prime:
$p =$ 445176959342405796225971179632912050657688043890876918715
6270863445095751602498829760043700275865765178209495023472
63691.

Case $N = 6666666666666666666666664$, $s = 2$, and
$k = 3538099915166679475086\,73$ gives prime:
$p =$ 8345434006468309665533960024044699357922019259293043
5437657420239410857.

Case $N = 4444$, $s = 5$, and
$k =$
634250335263436903664058267667793136668703383339035124341219
gives prime:
$p =$ 4561183846297793094013667057268432711555442123679889164761
7160284437223018040099748245821799167074469900659476172328
7421022326800053382794865523789665384953093547280261852600
1744316926769023258923584868066826250018895519990130703971
1225083761606510103396105271426626642249210463630491712973
5122836127957.

Case $N = 2222222$, $s = 5$, and
$k =$
278786749998964831243188210765046251607207021302616959150353
gives prime:
$p =$ 3742381762093988507152189447632190903649163162287335583067
1404936445168459532417824257034251542278496474548453008440
2575374798148854967611051490767707918981794830928860273675
4488895966928454234296362272519605051950977141702184978690
6574563422739262415235459932192589013385761664971623455438
97670489118447.

Appendix 2.

A numerical example with artificially small parameters.

This example illustrates the signature generation and verification procedures in the DSS based on difficulty of finding roots modulo large prime $p = Nk^2 + 1$, where $k = 132104433635297779312031$ is prime and $N = 238$,
i. e. $p = 41534763698924652690128708976232823900474001007 19$.

Suppose a user private key is $x = 352637898132454335 3612$ and the hash function value calculated from the document to be signed is $H = 735687901190172318234 57$. Then his public key is $y = x^k \bmod p =$ $386485810021935294036977484778855201836705519 7706$.

The signature is represented by a pair of numbers (R, S) which is generated as follows.

1. Generate a random number $t = 871933234152435531151 36314$.

2. Calculate the value $R = t^k \bmod p$:
$R = 19651943943290548836692335934353542255535285430 48$.

3. Calculate the value E using the formula $E = RH \bmod \delta$, where $\delta = 35488784369499179$:
$E = RH \bmod \delta = 30895498554403274$.

4. Calculate the value S using the formula $S = x^E t \bmod p$:
$x^E \bmod p = 41428960321742932773705412861186030086874128 46512$;
$S = x^E t \bmod p =$
$= 34593999607864752462932100387875805936013775274 59$.

The signature verification is performed as follows.

1. Calculate $E' = (S^k y^{p-E-1} \bmod p)H \bmod \delta$:
$S^k \bmod p = 12236286329907996953803299609450507757550316 56237$;
$p - E - 1 \bmod p - 1 =$
$= 41534763698924652690128708976232514945488456974 44$;
$y^{p-E-1} \bmod p =$

= 39407982034742155741062810189353996402481761 39523;
$R' = S^k y^{p-E-1} \bmod p =$
= 196519439432905488366923359343535422555352 8543048;
$E' = R'H \bmod \delta = 30895498554403274$.

2. Compare the values E' and E.

The signature is valid, since $E' = E$.

Received July 7, 2007

N. A. Moldovyan,

Specialized Center of Program Systems "SPECTR",
Kantemirovskaya, 10, St.Petersburg 197342, Russia;
ph./fax.7-812-2453743,
E–mail: *nmold@cobra.ru*