Short Signatures from Difficulty of Factorization Problem

Nikolay A. Moldovyan
Specialized Center of Program Systems "SPECTR"
Kantemirovskaya, 10, St-Petersburg 197342, Russia. (Email: nmold@cobra.ru)
(Received July 10, 2007; revised Dec. 5, 2007; and accepted Mar. 3, 2008)

Abstract

New ways are proposed to design short signature schemes based on difficulty of factorizing a composite number n that is a product of two large secret primes. The paper presents digital signature schemes in which the signature represents a pair of numbers (k, g) and its length is reduced to 320 bits providing security of the RSA cryptosystem with 1024-bit modulus.

Keywords: Cryptography, digital signature, factorization problem

1 Introduction

Public key cryptosystems based on hard mathematical problems are well approved for information authentication with digital signatures. An important practical problem is developing digital signature schemes (DSSes) with short signature length [7]. In general the minimum signature length provided by a DSS depends on the required security level that can be estimated as number of group operations that should be performed to forge a signature. In this paper we consider the signature length of different DSSes, which relates to the security level corresponding to 2^{80} group operations. At present this figure can be accepted as the minimum security level. The RSA and Rabin's DSSes based on factorization problem use the 1024-bit signature length [5]. The Rabin's DSS is very attractive as provably secure one, but becuase of comparatively large size of its signature it is not widely used in practice. More suitable for practical applications are DSSes based on difficulty of discrete logarithm problem in multiplicative groups [2] or in groups of elliptic curve points, which allow reducing the signature size [11]. The DSA standard and Schnorr's DSSes based on difficulty of finding discrete logarithm modulo large prime number provide comparatively short signatures having the 320-bit length, while providing the security level of the 1024-bit RSA [10]. The ECDSA standard also requires the use of the 320-bit signature size [1].

In present paper we consider some ways to reduce the signature length in DSSes based on difficulty of factorizing the composite number n generated as product of two unknown large primes q and r. To reduce the signature length in the DSSes based on complexity of the factorization problem we use the randomization signature formation mechanism. The paper proposes designs with signature length 320 bits. In Section 2 we describe a new signature formation mechanism used to reduce the signature length and present several new DSSes based on the factorization problem, which provide reduction of the signature length. We show that security of the proposed DSSes can be formally proved using the approach [8] used earlier to provide security proof for Schnorr's DSS. In Section 3 we propose a "double exponentiation" function to design modified variants of the 320-bit DSSes and estimate their security. Section 4 presents conclusion of the paper.

Notation:

G_m denotes multiplicative group $(\mathbb{Z}/m\mathbb{Z})^*$;

$\omega_m(x)$ denotes the order of the element $x \in G_m$;

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

$(x\|y)$ denotes the concatenation of the values x and y;

$\gcd(x, y)$ denotes the greatest common divisor of the values x and y;

$\operatorname{lcm}[x, y]$ denotes the least common multiple of the values x and y;

$H = F_H(M)$ denotes the hash value calculated from message M, using some specified hash function F_H Algorithm, for example SHA-1 [5].

2 Algorithms Based on a New Signature Formation Mechanism

2.1 Method for Reduction of the Signature Length

The well known cryptosystem RSA [9] is based on calculations modulo n that is a product of two randomly chosen prime numbers r and q: $n = rq$. The public key is represented by a pair of numbers (e, n), and the secret key is a number d, which is calculated using the following formula: $ed \bmod \varphi(n) = 1$, where $\varphi(n) = (r-1)(q-1)$ is Euler phi function of n. Security of this system is based on difficulty of calculating d while φ(n) is an unknown value. The φ(n) value can be easily calculated after factorization of the modulus n, therefore divisors of n have to be kept in secret (or to be annihilated after the keys e and d are generated). The signature corresponding to a plaintext 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 the message to be signed. The signature generation Equation is the following one: $S = H^d \bmod n$. In RSA the signature length is approximately equal to $|n|$. Security requirements define: $|n| \geq 1024$ bits (this value corresponds to the mentioned above minimum security level).

To reduce the signature length in the case of DSSes based on the factorization problem we use the novel signature formation mechanism [6] that can be applied while developing DSSes with two-element signatures denoted as (k, g). The mechanism is characterized in using a two-element public key with the structure (n, α), where $\alpha \in G_n$ and $\omega_n(\alpha) = \delta$, i. e. α is a generator of the δ order subgroup of G_n. The secret key is δ.

Some internal relation between the α and n values provides potentially some additional possibilities to factorize modulus n. Let us assume that δ is a divisor of $r-1$ and δ does not divide $q-1$. In practice to generate the α value the following expression is used: $\alpha = h^{\varphi(n)/\delta} \bmod n \neq 1$, where h is a random number for which we have $\gcd(h, n) = 1$. In genegal $\forall\alpha : \omega_n(\alpha) = \delta$ there exists such $h \in G_n$ that $\alpha = h^{\varphi(n)/\delta} \bmod n$. (Using the Euler's Theorem one can easily demonstrate that $\alpha^\delta = h^{\varphi(n)} \equiv 1 \bmod n$, where $\varphi(n) = (r-1)(q-1)$, i. e. if $h^{\varphi(n)/\delta} \bmod n \neq 1$, then the $h^{\varphi(n)/\delta} \bmod n$ value is a generator of the δ-order group). We have:

$$\begin{array}{l} \alpha = h^{\varphi(n)/\delta} \bmod n \equiv (h^{(q-1)})^{(r-1)/\delta} \bmod n \Rightarrow \\ \alpha \equiv (h^{(q-1)})^{(r-1)/\delta} \equiv 1^{(r-1)/\delta} \equiv 1 \bmod q \Rightarrow \\ \alpha - 1 \equiv 0 \bmod q \Rightarrow q|\alpha - 1 \Rightarrow \gcd(\alpha - 1, n) = q. \end{array}$$

Thus, in the considered case it is possible to factorize modulus using Euclidean Algorithm. Therefore some restrictions should be imposed on generation of the public key. A way preventing the described factorization method is to use such numbers r and q that both of them contain the same required large divisor δ, the δ^2 value dividing neither $r-1$ nor $q-1$. If this additional requirement is imposed, then the α parameter can be generated as follows: $\alpha = h^{L(n)/\delta} \bmod n \neq 1$, where $L(n) = \mathrm{lcm}\,[r-1, q-1]$ is generalized Euler's function. Thus, $\alpha = h^{uv} \bmod n \neq 1$, where $u = (r-1)/\delta$ and $v = (q-1)/\delta$. If we use, while generating the α value, a value that is simultaneously primitive element modulo r and primitive element modulo q as the number h (i.e. h is a "double" primitive element), then we will have simultaneously $\alpha \not\equiv 1 \bmod q$ and $\alpha \not\equiv 1 \bmod r$. While using a "double" primitive element we deterministically generate a "strong" α value. But it is not strictly necessary to use "double" primitive elements. We can generate a "strong" value α selecting random values h. In this case we should check if $\alpha \not\equiv 1 \bmod q$ and $\alpha \not\equiv 1 \bmod r$ holds.

The second way to generate "strong" public key is to use composite value δ, i. e. $\delta = \delta'\delta''$, where $\delta'|r-1$ and $\delta''|q-1$ and δ' and δ'' do not divide $q-1$ and $r-1$, correspondingly. For generating the parameter α we have the following formula: $\alpha = h^{L(n)/\delta} \bmod n \neq 1$, i. e $\alpha = h^{uv} \bmod n \neq 1$, where $u = (r-1)/\delta'$ and $v = (r-1)/\delta''$. Analogously to the first case, while using the value h that is "double" primitive element, we get $\alpha \not\equiv 1 \bmod q$ and $\alpha \not\equiv 1 \bmod r$.

Thus, we have two different ways to define difficulty of the n modulus factorization in the considered DSS. Unfortunately in the first way we have a problem to avoid possibility to calculate the secret parameter δ without factorizing the n modulus. Indeed, we have:

$$\begin{aligned} n - 1 &= (u\delta + 1)(v\delta + 1) - 1 \\ &= uv\delta^2 + u\delta + v\delta = (uv\delta + u + v)\delta. \end{aligned}$$

Usually the value $n-1$ can be easily factorized. Therefore the secret δ can be recovered, if no new restriction requirements are imposed on selection of the n modulus. In the second way factorization of the value $n-1$ does not allows one to determine the δ secret. Thus, we should use the second way while generating the public key. To choose the size of the δ value we should take into account that the α value can be used to factorize the modulus n with the help of the calculation of the value $\gcd(\alpha^i \bmod n - 1, n)$ for $i = 1, 2, \ldots \min\{\delta', \delta''\}$. Therefore we should use the 80-bit values δ' and δ''. Thus, we get the length of the value δ should be $|\delta| \approx |\delta'| + |\delta''| \geq 160$ bit.

2.2 Digital Signature Schemes

A secure variant of the DSS with the 320-bit signature length is described by the following verification Equation:

$$k - g = F_H(M \| \alpha^{k+g(v-1)} \bmod n), \qquad (1)$$

where v is a specified 80-bit prime number and M is a message. The signature generation is performed as follows:

1) Generate a random number U and calculate $H = F_H(M\|\alpha^U \bmod n)$;

2) Solve simultaneously Equation $k - g = H$ and congruence $k + g(v-1) \equiv U \bmod \delta$.

The solution gives the k and g signature elements:

$$g = \frac{U - H}{v} \bmod \delta \quad \text{and} \quad k = H + g. \qquad (2)$$

The signature size is $|k| + |g| \approx |H| + |\delta| \approx 320$ bits in the case of the 160-bit hash function. Note that without using the prime v the signature scheme is not secure, since in such case the secret δ is not used to calculate signature, if we have $U > H$.

One can simplify the verification Equation and present the following modified DSS:

$$k = F_H(M\|\alpha^{kg} \bmod n). \tag{3}$$

In this case the signature is calculated using the formulas:

$$k = F_H(M\|\alpha^{U} \bmod n) \text{ and}$$
$$g = \frac{U}{k} \bmod \delta. \tag{4}$$

Note that, while generating the signature, the events corresponding to the case $\gcd(\delta, k) > 1$ have negligible probability (if one of such cases takes place, then the signer should repeat the signature generation procedure using another value U). If $k|U$, then the value g is calculated without using the secret value δ. However such events have also negligible probability due to sufficiently large value $|k| = |H| = 160$ bits (we assume that hash function SHA-1 is used).

2.3 Security Discussion

The assumptions underlying the schemes described above are the following two:

1) Problem to factorize n using the public key (n, α) is difficult.

2) Finding discrete logarithm $x = \log_\alpha y$ modulo n is difficult.

Solving one of these two hard problems allows one to determine the secrete key. For the first problem it is evident. For the second problem we should consider the following attack. Select two 160-bit numbers k' and g' and calculate the value $y = \alpha^{kg} \bmod n$. Finding discrete logarithm $x = \log_\alpha y \pmod{n}$ and factorizing the value $kg - x$ one can get the secret value δ. For the known value $\delta = \delta'\delta''$ the difficulty of finding discrete logarithm $x = \log_\alpha y \pmod{n}$ can be estimated as $\sqrt{z}$ exponentiation operations, where $z = \max\{\delta', \delta''\}$. However for unknown value $\omega_n(\alpha)$ the difficulty of the second problem is approximately equal to $\sqrt{\delta'\delta''}$ operations. Let us note that difficulty of the second problem depends on the difficulty of the first one. Indeed, solving the first problem reduces the second problem to the problem of finding logarithms modulo q and modulo r [5]. Since $|q| \approx |r| \approx 0.5 \cdot |n|$ the difficulty of the second problem decreases drastically after factorizing modulus n.

Like in the DSS proposed by Schnorr [11], in the proposed DSS Algorithms the hash function value is computed after the value $\alpha^U \bmod n$ is calculated and concatenated to the message that is to be signed. Due to this feature the formal security evidence of the proposed DSS Algorithms can be provided using the approach proposed in paper [8] and applied to provide formal security proof for Schnorr's DSS Algorithm. We will assume that the hash function used in the considered DSSes posses no special properties that the adversary can take advantage of. In compliance with that approach we uses the following reductionist security claim corresponding to each of the proposed DSSes. If an adversary can forge signatures, then he can factorize modulus n in essentially the same amount of time that it takes to forge a signature. On the analogy of the formal security consideration of Scnorr's DSS the following argument can be supplied.

Suppose that the adversary can forge a signature for message M. While his computing the value $H = F_H(M\|\alpha^U \bmod n)$, suppose that he is suddenly given another hash function F'_H. Since the used hash function has no special properties (see our assumption above) used in the forgery Algorithm the last will work equally well with both F_H and F'_H (the formal proof of this fact is provided in [8] using the random oracle model and this fact does not concern the details of the proposed DSSes). Thus, the forger computes $H' = F'_H(M\|\alpha^U \bmod n)$ as well as $H = F_H(M\|\alpha^U \bmod n)$ and produces two valid signatures (k', g') and (k, g) for M, respectively. The both signatures are calculated with the same value U but with different H' and H.

Therefore, in the case of the first of the proposed DSSes, we have $k' \neq k$ and $g' \neq g$ but $k' + g'(v-1) \equiv U \bmod \delta$ and $k + g(v-1) \equiv U \bmod \delta$, i. e. we have $k' + g'(v-1) \equiv k + g(v-1) \bmod \delta$, therefore δ divides the value $k' - k + (g' - g)(v-1)$. Since the size of δ is sufficiently small it is not difficult to find δ after the factorization of the value $k' - k + (g' - g)(v-1)$.

In the case of the second of the proposed DSSes, we have $k' \neq k$ and $g' \neq g$ but $k'g' \equiv U \bmod \delta$ and $kg \equiv U \bmod \delta$, i. e. we have $k'g' \equiv kg \bmod \delta \Rightarrow \delta | k'g' - kg$ and δ can be computed factorizing the value $k'g' - kg$. Using the computed value δ it is easy to factorize n. Indeed, due to comparatively small size of δ it is easy to factorize δ: $\delta = \delta'\delta''$ and to calculate $\gcd(\alpha^{\delta'} \bmod n - 1, n) = r$ (see Subsection 2.1).

Thus, a successful attack on each of the proposed DSSes, which provides possibility to forge signature on a message, can be applied to factorize n, i.e. the proposed DSSes are as secure as problem of factorizing n is difficult.

In the next Section we propose a signature formation mechanism that provides possibility to use prime value δ. An interesting peculiarity of the scheme described below is using calculations modulo prime q that is secret value.

3 Signature Schemes with Prime δ

3.1 The Used Function

Let us consider the main trick that provides to use prime values of secrete number δ. Suppose we would like to provide to a verifier possibility to check if a is congruent to b modulo q, the q value being a secret one. It is possible to be done using very simple mechanism avoiding direct calculations modulo q. Really, let $p = 2n + 1$ is a prime, where n is the product of two large primes q and r, i. e. $n = qr$, and β is an element in G_p for which we have $\omega_p(\beta) = q$. Then we have

$$\beta^a \equiv \beta^b \bmod p \Leftrightarrow a \equiv b \bmod q.$$

Thus, one can check validity of the last congruence performing calculations modulo p, i. e. without use of the secrete value q. In the DSS design we use the following function:

$$y = \beta^{\alpha^x \bmod n} \bmod p,$$

where $p = 2n + 1$, $n = qr$, $\beta \in G_p$, $\omega_p(\beta) = q$, and the value $\alpha \in G_q$ is such that $\omega_n(\alpha) = \delta(r-1)$ and $\omega_q(\alpha) = \delta$, where δ is a prime such that $\delta|q-1$ and $\delta \not| r-1$.

3.2 Signature Scheme

Using this mechanism we modify the verification Equations (1) and (2), respectively, in the following way:

$$\begin{aligned} k - g &= F_H(M \| \beta^{\alpha^{k+g(v-1)} \bmod n} \bmod p), && (5) \\ k &= F_H(M \| \beta^{\alpha^{kg} \bmod n} \bmod p), && (6) \end{aligned}$$

where the public key is (p, β, α) and the secret key is (q, δ). We suppose the following minimum length of the key values: $|\delta| = 160$ bits, $|q| = 512$ bits. In DSSes described by verification Equations (5) and (6) the g value in the signature is an element of the prime order group G_δ: $g \in G_\delta$.

In the modified DSS described by Equation (5) the signature elements g and k are defined by Formula (2), with exception that now we have $Z = F_H(M \| \beta^{\alpha^U \bmod q} \bmod p)$. In the modified DSS described by Equation (6) the signature element k is calculated as follows:

$$k = F_H(M \| \beta^{\alpha^U \bmod q} \bmod p).$$

The second signature element is defined by Formula (4).

Statement 1. The signature verification Equations (5) and (6) work correctly.

Proof. For Equations (5) and (6) the proof is analogous. Let us consider the case of the verification Equation (6). Let (k, g) be a valid signature corresponding to some signed message M. Taking into account that $\omega_q(\alpha) = \delta$, $\omega_p(\beta) = q$, and $g = U/k \bmod \delta$, from the verification Equation we get:

$$\begin{aligned} k' &= F_H(M \| \beta^{\alpha^{kg} \bmod n} \bmod p) \\ &= F_H(M \| \beta^{\alpha^{kg} \bmod q} \bmod p) \\ &= F_H(M \| \beta^{\alpha^{k\frac{U}{k}} \bmod q} \bmod p) \\ &= k. \end{aligned}$$

Since $k' = k$ the signature verification result is positive, i.e. the verification Equation (6) works correctly.

Let us consider the case of the verification Equation (5). For some valid signature (k, g) for the message M we have (see Formulas (2)):

$$\begin{aligned} k - g &= H, \text{ where} \\ H &= F_H(M \| \beta^{\alpha^U \bmod q} \bmod p) \\ &= F_H(M \| \beta^{\alpha^U \bmod n} \bmod p) \quad \text{and} \\ H' &= F_H(M \| \beta^{\alpha^{k+g(v-1)} \bmod n} \bmod p) \\ &= F_H(M \| \beta^{\alpha^{H+g+g(v-1)} \bmod n} \bmod p) \\ &= F_H(M \| \beta^{\alpha^U \bmod n} \bmod p) \\ &= H. \end{aligned}$$

Since $H' = H$ the verification Equation (5) works correctly. □

3.3 Security Discussion

In the DSSes proposed in Section 3.2 the β value should satisfy the following security requirement: *the $\omega_n(\alpha)$ value should be sufficiently large, i.e. $|\omega_n(\alpha)| > 160$ bits.* This requirement is defined by the following possible attack against the Scheme (6) (analogous attack is possible against Scheme (5)):

1) Calculate the value $y = \alpha^{kg} \bmod n$;

2) Find the value $x = \log_\alpha y \pmod{n}$;

3) Calculate δ as one of divisors of the value $kg - x$.

If $\omega_n(\alpha) > 160$ bits (this condition is satisfied in the considered case, since we have $\omega_n(\alpha) = \delta(r-1)$), then Step 2) is computationally infeasible. Note that until modulus n is not factorized the divisors of $\omega_n(\alpha)$ are not known. We also suppose to follow the recommendations by [3], i.e. to use strong primes q and r to generate n. For strong prime r the value $r - 1$ contains large prime divisor, therefore finding the logarithm $x = \log_\alpha y \pmod{n}$ is computationally intractable even in the case of known value $\omega_n(\alpha)$.

In another attack one can try to find value $x' = \log_\beta K \pmod{p}$, where $K = \beta^{\alpha^{kg} \bmod n} \bmod p$, and calculate q as a divisor of the value $(\alpha^{kg} \bmod n) - x'$. Then the secret value δ can be determined dividing the value $q-1$. Due to sufficiently large values $|q|$ and $|p|$ this attack is also computationally infeasible.

Security against some other attacks is justified by Statement 2.

Statement 2. An attack providing calculation of the secret value δ is as difficult as factorizing the modulus n is difficult.

Proof. Suppose an attack allows to calculate δ. Since $\alpha^\delta \equiv 1 \bmod q$, then $(\alpha^\delta \bmod n) \equiv 1 \bmod q \Leftrightarrow q|\alpha^\delta \bmod n - 1 \Leftrightarrow \gcd(\alpha^\delta \bmod n - 1, n) = q$. Thus, difficulty of factorizing the n modulus is comparable with difficulty of the considered attacks.

In the DSSes proposed in Section 3.2 the hash function is calculated after the value $R = \beta^{\alpha^{kg} \bmod q} \bmod p$ is computed. Therefore the forger is forced to choose the value U before the determination of the hash value (we use the standard assumption the used hash function is secure, i. e. the function F_H possesses no special properties that the forger can take advantage of), like in the case of Snorr's DSS (for details see pp. 25-26 in [4]). So we can apply the same argument as that used to provide formal security evidence for DSSes presented in Section 2 (see Subsection 2.3). Such argumentation shows that the successful forger can find the value δ and then he can factorize the modulus n (see proof of Statement 2).

Thus, the DSSes presented by Formulas (5) and (6) are also as secure as problem of factorizing the number n is difficult □

3.4 Security Estimation

In this Subsection we estimate security of the DSS described by Formula (5) and (6) as complexity of the best known Algorithms providing calculation of the δ value given the function $y = \beta^{\alpha^x \bmod n} \bmod p$. Due to "double exponentiation" construction of this function the methods based on calculating discrete logarithms are not efficient relatively required time and storage. The following Algorithm implements a more efficient method.

Algorithm 1

```
1: Select random $U > 2^{|\delta|+10}$ and calculate $y = \beta^{\alpha^U \bmod n} \bmod p$.
2: For $i = 0$ to $N$, where integer $N \approx \sqrt{\delta}$, calculate $z'(i) = \beta^{\alpha^{iN} \bmod n} \bmod p$.
3: Order the Table of pairs $(i, z'(i))$ according to the $z'(i)$ value and set $j = 0$.
4: Calculate $z''(j) = y^{\frac{1}{\alpha^j} \bmod n} \bmod p$.
5: Check if in the Table there exists $z'(i_0)$ such that $z'(i_0) = z''(j)$. If $z''(j) \neq z'(i)$ for $i = 0$ to $N$, then increment the $j$ counter $j \leftarrow j + 1$ and go to Step 4.
6: Calculate values $U' = i_0N + j$ and factorize the $U - U'$ value.
7: Select divisor $\delta$ such that $\beta^{\alpha^\delta \bmod n} \bmod p = \beta$.
```

The Algorithm computes the value $U' = i_0N + j_0$ such that $y = \beta^{\alpha^{U'} \bmod n} \bmod p$. Indeed, suppose for i_0 and j_0 we have

$$y^{\frac{1}{\alpha^{j_0}} \bmod n} \bmod p = \beta^{\alpha^{i_0 N} \bmod n} \bmod p.$$

Then

$$\begin{aligned} y &\equiv \beta^{\alpha^{i_0N+j_0} \bmod n} \bmod p \\ \Rightarrow \beta^{\alpha^U \bmod n} &\equiv \beta^{\alpha^{U'} \bmod n} \bmod p \\ \Rightarrow \beta^{\alpha^U \bmod q} &\equiv \beta^{\alpha^{U'} \bmod q} \bmod p \\ \Rightarrow \alpha^U &\equiv \alpha^{U'} \bmod q. \end{aligned}$$

Therefore $U \equiv U' \bmod \delta \Rightarrow \delta | U - U'$.

Difficulty of Step 2 is about $2N = 2\sqrt{\delta}$ exponentiation operations. Difficulty of Step 3 is about $N \log_2 N$ comparison operations performed on $|p|$-bit values. Difficulty of Steps 4 and 5 is about $\sqrt{\delta}$ exponentiations plus $2^{-1}N \log_2 N$ comparison operations. Difficulty of Steps 1, 6, 7 and difficulty of all comparison operations are negligible in comparison with difficulty of all exponentiation operations. In total the difficulty of Algorithm 1 is $W \approx 3\sqrt{\delta}$ exponentiation operations. For $|\delta| \approx 160$ bits we have $W \approx 2^{81}$ exponentiation operations. The Algorithm requires very lage storage $\approx 2^{90}$ bits for $\approx 2^{80}$ $|p|$-bit numbers. Algorithm 1 is efficient on time, but not efficient on storage.

Minimum storage requirement and efficient on time computations are achieved with the Floyd's Algorithm [10] applied to the $\{x_0, x_1, .., x_i, ..., x_j, ...\}$ sequence of the 1024-bit numbers obtained with recursive formula

$$x_{i+1} = \beta^{\alpha^{x_i} \bmod n} \bmod p,$$

where x_i is selected arbitrarily. Such random sequence represents a a non-periodic part (tail) followed by the periodic rest (multiple repetition of some cycle). The average length of the tail is $\lambda = \sqrt{\pi\delta/8}$. The average length of the cycle is $\mu = \sqrt{3\pi\delta/8}$ [10]. For our sequence the Floyd's Algorithm [10] allows to find i and j such that $x_i \neq x_j$ and $x_{i+1} = x_{j+2}$, where

$$(x_{i+1}, x_{j+2}) = (\beta^{\alpha^{x_i} \bmod n} \bmod p, \beta^{\alpha^{x_j} \bmod n} \bmod p),$$

performing about $4\mu(1 + \lambda/\mu) \approx 4\sqrt{\delta} \approx 2^{82}$ exponention operations and using extremely small storage. Therefore the δ divides the $x_i - x_j$ value and can be easily find. Thus, the secret prime δ should have the length $|\delta| \geq 156 \approx 160$ bits. So we have the minimum signature length ≈ 320 bits for the DSS described by Formulas (5) and (6).

4 Conclusion

Using a novel mechanism in the DSS based on difficulty of factorization problem we have reduced the signature size in such schemes to 320 bits in the case of the minimum security level (2^{80} operations). To obtain possibility to use prime value of the secrete key element δ we have proposed the DSSes defined by the "three-level" verification Equations, combining calculations in four different groups G_δ, G_q, G_n, and G_p. In such schemes, while generating signature, calculations are performed in three groups G_δ, G_q, and G_p. While verifying the signature, the calculations are performed in two groups G_n, and G_p. In the streighfoward implementation of the DSSes based on "three-level" verification Equations with the modulus $p = 2rq + 1$ the public key generation procedure is sufficiently more complex than in the DSA standard and in RSA, however it can be essentially simplified using the p modulus with the structure $p = erq + 1$, where e is the t-bit even number, $t = 10$ to 16 bits. In this case for the selected primes q and r one can choose with high probability e such that p is prime. This variant provides sufficient computational efficiency of the key generation procedure.

The DSSes described by verification Equations (1) and (3) use composite element δ, however they have simpler design and are faster while both the signature generation and the signature verification. Schemes (1) and (3) appear to be more interesting for practical applications. The computational efficiency of the signature generation in these DSSes is about the same as in the DSA standard. Complexity of the signature verification procedure is about two times lower than in DSA. For the proposed DSSes we have provided formal security evidence.

Due to using a novel signature formation mechanism we have succeeded to develop DSSes having comprehensible design and providing at present the shortest signature size for the DSSes based on factorization problem.

Acknowledgements

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

References

[1] ANSI X9.62, and FIPS 186-2, *Elliptic Curve Signature Algorithm*, 1998.

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

[3] J. Gordon. "Strong primes are easy to find," *Advances in cryptology - Eurocrypt '84*, LNCS 209, pp. 216-223, Springer-Verlag, 1985.

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

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

[6] A. A. Moldovyan, D. N. Moldovyan, and L. V. Gortinskaya, "Cryptoschemes based on new signature formation mechanism," *Computer Science Journal of Moldova*, vol. 14, no. 3, pp. 397-411, 2006.

[7] L. Pintsov, and S. Vanstone "Postal revenue collection in the digital age," *Proceedings of Financial Cryptography 2000*, LNCS 1962, pp. 105-20, Y. Frankel, Editor, Springer-Verlag, Berlin, 2000.

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

[9] R. L. Rivest, A.Shamir, and L. M. Adleman. "A method for obtaining digital signatures and public key cryptosystems," *Communications of the ACM*, vol. 21, no 2, pp. 120-126, 1978.

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

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

Nikolay A. Moldovyan is an honored inventor of Russian Federation (2002), a chief researcher with the Specialized Center of Program Systems "SPECTR", and a Professor with the Saint Petersburg Electrical Engineering University. His research interests include computer security and cryptography. He has authored or co-authored more than 50 inventions and 200 scientific articles, books, and reports. He received his Ph.D. from the Academy of Sciences of Moldova (1981). Contact him at: nmold@cobra.ru.