British Journal of Mathematics & Computer Science
13(1): 1-20, 2016, Article no.BJMCS.22319
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

Fast and Secure Modular Matrix Based Digital Signature

S. K. Rososhek[1*]

[1]*Faculty of Mathematics and Mechanics, Tomsk State University, Tomsk, Russia.*

Article Information

DOI: 10.9734/BJMCS/2016/22319
Editor(s):
(1) Victor Carvalho, Polytechnic Institute of Cávado and Ave, Portuguese Catholic University and Lusiada University, Portugal.
Reviewers:
(1) Hazar Abu-Khuzam, American University of Beirut, Beirut, Lebanon.
(2) Grienggrai Rajchakit, Maejo University, Thailand.
(3) Varun Shukla, Dr. A.P.J. Abdul Kalam Technical University, India.
Complete Peer review History: http://sciencedomain.org/review-history/12382

Original Research Article

Received: 28 September 2015
Accepted: 03 November 2015
Published: 21 November 2015

Abstract

In this paper, digital signature that is resistant to attacks by quantum computer is designed in two versions – with the application and with message recovery. We study the security and performance of this digital signature by comparing it with the signatures of RSA and DSA. In particular, it appears that the new digital signature is not less secure, but it is much faster than these signatures are commonly used in practice.

Keywords: Digital signature; matrix ring; residue ring; automorphism.

1 Introduction

First commercial version of a quantum computer with 512 qubits [1] appeared in 2014, contrary to the skeptics opinion. Not have long to wait until the turn of 1024 qubits, the overcoming of which would threaten the security of the existing Internet infrastructure. In particular, the implementation on this computer of Shor's algorithms solving the integer factorization and discrete logarithm problems [2] will allow the possibility to break such public key cryptosystems as the RSA and the corresponding digital signature and moreover the ElGamal cryptosystem and the digital signature DSA. But then the security of the Internet infrastructure would be threatened as a secure communication protocol OpenSSL uses cryptographic primitives such as RSA, Diffie-Hellman key exchange protocol and digital signature DSA [3].

There is the need to replace potentially vulnerable cryptographic primitives with others components which are resistant to attacks by a quantum computer. You can offer one of the possible variants of replacement: instead RSA to use BMMC [4] or MMMC1 [5] and instead Diffie-Hellman key exchange protocol to use its

*Corresponding author: E-mail: rososhek@list.ru;

non-commutative analogue on the basis of BMMC [6]. It remains to replace the digital signature of DSA and that is the aim of this paper.

Briefly describe the contents of the paper. In Section 2 the new modular matrix based digital signature scheme (MMDS) with application is described. An example of computations in this digital signature scheme is contained in Section 3. Message recovery version of the MMDS is described in Section 4. The security of the MMDS is investigated in Section 5. Comparison of the different digital signatures performance is made in Section 6. Finally, a conclusion is given in Section 7.

2 Modular Matrix Based Digital Signature Scheme (MMDS) Description

2.1 Key Generation

Alice should do the following:

2.1.1 Generate a random prime number p.

2.1.2 Select one of two options:

a) Generate another random prime number $q \neq p$ and computes $n = pq$.

b) Generate a random integer $r \geq 2$ and computes $n = p^r$.

2.1.3 Select two random invertible matrices V, W in the Abelian subgroup *G* [5] of the group $GL_2(\mathbf{Z}_n)$.

2.1.4 Compute the matrix

$$U = V^{-1}W.$$

2.1.5 Alice's master public key is:

$$(n, U),$$

Alice's master private key is:

$$(V, W).$$

2.2 Digital Signature Generation

For the digital signature on the message *m* Alice should do the following:

2.2.1 Select a random matrix *Y* in the subgroup *G* of the group $GL_2(\mathbf{Z}_n)$ and a random matrix *L* in the group $GL_2(\mathbf{Z}_n)$. (Let *G* be the following set of 2×2 – matrices:

$$G = \left\{ \begin{pmatrix} a & b \\ b & a \end{pmatrix} \middle| a, b \in \mathbf{Z}_n \text{ } and \text{ } (a^2 - b^2) \in \mathbf{Z}_n^* \right\},$$

$\mathbf{Z}_n^*$ is an unit group of the residue ring $\mathbf{Z}_n$ modulo *n*.

It is easy to verify that *G* is an abelian subgroup of the group $GL_2(\mathbf{Z}_n)$.) [5]

2.2.2 Select random elements μ, λ in the residue ring $\mathbf{Z}_n$.
2.2.3 Alice's session private key is (μ, λ, Y, L).
2.2.4 Let $\chi_U, \chi_{WY}, \chi_{VY}$ be the automorphisms of the matrix ring $M_2(\mathbf{Z}_n)$ defined as the following:

$$\chi_U : D \to U^{-1}DU \,,$$
$$\chi_{VY} : D \to (VY)^{-1}D(VY),$$
$$\chi_{WY} : D \to (WY)^{-1}D(WY)$$

for every $D \in M_2(\mathbf{Z}_n)$.

2.2.5 Alice computes matrix *r* and bit string *s* as follows:

$$r = \lambda\chi_{VY}(L),$$
$$t = \chi_{WY}(L\ ), \mu t,$$
$$\delta = \lambda + \mu, s = h((m)_2 \parallel (\delta t)_2),$$

where $(m)_2$ is the bit string – binary representation of the message *m*, $(\delta t)_2$ is the bit string obtained after transferring matrix δt in the string of numbers as follows:

$$\delta t = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \to a_1 \parallel a_2 \parallel a_3 \parallel a_4$$

and replace the numbers a_i by their binary representations, $h(x)$ is the hash value of the bit string *x*.

2.2.6 Let μt be a session public key (verification key) for the verification of Alice's signature of the message *m* and (r, s) is the Alice's signature of the message *m*.

2.3 Digital Signature Verification

To verify the Alice's digital signature of message *m* Bob should do the following:

2.3.1 Obtain Alice's authentic master public key (*n*, *U*) and session public key μt .
2.3.2 Compute

$$v = \mu t + \chi_U(r) .$$

2.3.3 Compute

$$s' = h((m)_2 \parallel (v)_2)$$

2.3.4 Accept Alice's signature of the message *m* if and only if $s = s'$.

Theorem. MMDS verification is correct.

Proof. By verification description

$$v = \mu t + \chi_U(r) = \mu t + \lambda (V^{-1}W)^{-1}(VY)^{-1}L(VY)(V^{-1}W) = \mu t + \lambda t = (\mu + \lambda)t = \delta t .$$

Therefore

$$s' = h((m)_2 \parallel (v)_2) = h((m)_2 \parallel (\delta t)_2) = s$$

3 MMDS Example

In this section an example of computations in MMDS with artificially small modulus of the residue ring is given.

3.1 Key Generation

Alice should do the following:

3.1.1 Select $n = 35$.

3.1.2 Select

$$V, W \in G \subset GL_2(\mathbf{Z}_{35}),$$
$$V = \begin{pmatrix} 4 & 7 \\ 7 & 4 \end{pmatrix}, W = \begin{pmatrix} 7 & 9 \\ 9 & 7 \end{pmatrix}.$$

3.1.3 Compute

$$U = V^{-1}W = \begin{pmatrix} 0 & 11 \\ 11 & 0 \end{pmatrix}.$$

3.1.4 Alice's master public key is

$$\left(n = 35, U = \begin{pmatrix} 0 & 11 \\ 11 & 0 \end{pmatrix} \right),$$

Alice's master private key is

$$\left(V = \begin{pmatrix} 4 & 7 \\ 7 & 4 \end{pmatrix}, W = \begin{pmatrix} 7 & 9 \\ 9 & 7 \end{pmatrix} \right).$$

3.2 Signature Generation

Alice should do the following:

3.2.1 Select the matrices:

$$Y = \begin{pmatrix} 5 & 3 \\ 3 & 5 \end{pmatrix} \in G \subset GL_2(\mathbf{Z}_{35}), \quad L = \begin{pmatrix} 6 & 3 \\ 7 & 4 \end{pmatrix} \in GL_2(\mathbf{Z}_n).$$

3.2.2 Select the residues

$$\lambda, \mu \in \mathbf{Z}_{35}, \lambda = 8, \ \mu = 11.$$

3.2.3 Alice's session private key is

$$\left(\lambda = 8, \mu = 11, Y = \begin{pmatrix} 5 & 3 \\ 3 & 5 \end{pmatrix}, L = \begin{pmatrix} 6 & 3 \\ 7 & 4 \end{pmatrix} \right).$$

3.2.4 Compute

$$\chi_U(L) = U^{-1}LU = \begin{pmatrix} 0 & 16 \\ 16 & 0 \end{pmatrix} \begin{pmatrix} 6 & 3 \\ 7 & 4 \end{pmatrix} \begin{pmatrix} 0 & 11 \\ 11 & 0 \end{pmatrix} = \begin{pmatrix} 4 & 7 \\ 3 & 6 \end{pmatrix},$$

$$\chi_{VY}(L) = (VY)^{-1}L(VY) = \begin{pmatrix} 33 & 4 \\ 4 & 33 \end{pmatrix} \begin{pmatrix} 6 & 3 \\ 7 & 4 \end{pmatrix} \begin{pmatrix} 6 & 12 \\ 12 & 6 \end{pmatrix} = \begin{pmatrix} 6 & 7 \\ 3 & 4 \end{pmatrix},$$

$$\chi_{WY}(L) = (WY)^{-1}L(WY) = \begin{pmatrix} 29 & 3 \\ 3 & 29 \end{pmatrix} \begin{pmatrix} 6 & 3 \\ 7 & 4 \end{pmatrix} \begin{pmatrix} 27 & 31 \\ 31 & 27 \end{pmatrix} = \begin{pmatrix} 4 & 3 \\ 7 & 6 \end{pmatrix}.$$

3.2.5 Compute signature (*r*, *s*) as follows:

$$r = \lambda \chi_{VY}(L) = 8 \begin{pmatrix} 6 & 7 \\ 3 & 4 \end{pmatrix} = \begin{pmatrix} 13 & 21 \\ 24 & 32 \end{pmatrix},$$

$$t = \chi_{WY}(L) = \begin{pmatrix} 4 & 3 \\ 7 & 6 \end{pmatrix}, \mu t = 11\begin{pmatrix} 4 & 3 \\ 7 & 6 \end{pmatrix} = \begin{pmatrix} 9 & 33 \\ 7 & 31 \end{pmatrix},$$

μt is Alice's session public key (session verification key),

$$\delta = \lambda + \mu = 19,\ \delta t = 19\begin{pmatrix} 4 & 3 \\ 7 & 6 \end{pmatrix} = \begin{pmatrix} 6 & 22 \\ 28 & 9 \end{pmatrix},$$

$$s = h\big((m)_2 \,\|\, (\delta t)_2\big),$$

where $(m)_2$ is a bit string – binary representation of the message *m*, $(\delta t)_2$ is a bit string obtained after transferring matrix δt in the string of numbers and replace the numbers by their binary representations as follows:

$$06 \,\|\, 22 \,\|\, 28 \,\|\, 09 \to 00000110 \,\|\, 00100010 \,\|\, 00100100 \,\|\, 00000101.$$

3.3 Signature Verification

Bob should do the following:

3.3.1 Obtain the authentic Alice's master public key $\left(n = 35, U = \begin{pmatrix} 0 & 11 \\ 11 & 0 \end{pmatrix}\right)$ and session public key $\mu t = \begin{pmatrix} 9 & 33 \\ 7 & 31 \end{pmatrix}$.

3.3.2 Compute

$$v = \mu t + \chi_U(r) = \mu t + U^{-1} r U =$$
$$\begin{pmatrix} 9 & 33 \\ 7 & 31 \end{pmatrix} + \begin{pmatrix} 0 & 16 \\ 16 & 0 \end{pmatrix}\begin{pmatrix} 13 & 21 \\ 24 & 32 \end{pmatrix}\begin{pmatrix} 0 & 11 \\ 11 & 0 \end{pmatrix} =$$
$$\begin{pmatrix} 9 & 33 \\ 7 & 31 \end{pmatrix} + \begin{pmatrix} 32 & 24 \\ 21 & 13 \end{pmatrix} = \begin{pmatrix} 6 & 22 \\ 28 & 9 \end{pmatrix}.$$

3.3.3 Since $v = \delta t$, therefore

$$s = h((m)_2 \,\|\, (\delta t)_2) = h((m)_2 \,\|\, (v)_2) = s'.$$

Note. In practice, Bob during signature verification uses a hash function with a 256-bit hash value. If the modulus is too small number, then apply such hash function is difficult. Therefore, in the example with *n* = 35 due to the inability to compute a 256-bit hash value using indirect argument. Equality test of hash values produced by an indirect test for equality $v = \delta t$, although Bob computes only *v* value and does not know the δt value (δt value is taken from the steps of the Alice's signature generation).

4 Message Recovery Version of MMDS

Since this version of MMDS provides message recovery, a suitable redundancy function is required to guard against existential forgery attack [7]. Redundancy function selection will be made in this version of MMDS description. The message with bit length of more than 256 bits without additional transformations can not be signed by the MMDS with message recovery, so for these messages usually the MMDS with the application (Section 2) is used. Therefore the message recovery version of MMDS will be used mainly for short messages, i.e. for messages of bit length less than or equal to 256 bits (for such messages can also be used MMDS with the application). Of course, for "long" messages with the bit length of more than 256 bits, you can not sign the original message itself, but its hash value as a substitute for the original message using MMDS with message recovery and it is recovered during signature verification.

4.1 Key Generation

Alice should do the following:

4.1.1 Select 128-bit positive integer n_0.

4.1.2 Append on the left of the binary representation of n_0 a single bit 1 and get a 129-bit number *n*.

4.1.3 Select matrices

$$V, W \in G \subset GL_2(\mathbf{Z}_n)$$

and random unit $\gamma \in \mathbf{Z}_n^*$.

4.1.4 Compute

$$U = V^{-1}W.$$

4.1.5 Alice's public key is

(n, U),

Alice's private key is

$$(\gamma, V, W).$$

4.2 Signature Generation

To sign a message *m* with bit length of not more than 256 bits Alice should do the following:

4.2.1 Select the redundancy function of the following form:

$$R(m) = ((m)_2 \oplus h((m)_2)) \parallel h((m)_2).$$

4.2.2 Compute the hash value $h(R(m))$, it is a public session key (session verification key) for Alice's signature of the message *m*.

4.2.3 Represent 512-bit string $R(m)$ as follows:

$$R(m) = m_1 \| m_2 \| m_3 \| m_4$$

4.2.4 128-bit strings m_i are replaced by their decimal representations M_i and form a matrix

$$M = \begin{pmatrix} M_1 & M_2 \\ M_3 & M_4 \end{pmatrix}.$$

Note. Selecting the 129-bit modulus *n* of the residue ring ensures that all $M_i, i = 1,...,4$ are residues modulo *n*.

4.2.5 Select the random matrices $Y \in G$ and $L \in GL_2(\mathbf{Z}_n)$.

4.2.6 Compute L^{-1} and Y^{-1}.

4.2.7 Let $\chi_U, \chi_{VY}, \chi_{WY}$ be the automorphisms of the matrix ring $M_2(\mathbf{Z}_n)$ defined as the following:

$$\chi_U : D \to U^{-1}DU,$$
$$\chi_{VY} : D \to (VY)^{-1}D(VY),$$
$$\chi_{WY} : D \to (WY)^{-1}D(WY)$$

for every $D \in M_2(\mathbf{Z}_n)$.

4.2.8 Compute Alice's signature (*r*, *s*) of the message *m*:

$$r = \gamma\chi_{VY}(L) = \gamma(VY)^{-1}L(VY),$$

$$s = \gamma^{-1}\chi_{WY}(L^{-1}) = \gamma^{-1}M(WY)^{-1}L^{-1}(WY).$$

4.3 Signature Verification

Bob should do the following:

4.3.1 Obtain the authentic Alice's public master key (n,U) and public session key $h(R(m))$.

4.3.2 Compute

$$z = \chi_U(r) = U^{-1}rU.$$

4.3.3 Compute

$$sz = M'.$$

4.3.4 Let $(m')_2$ be a bit string obtained by transferring the matrix $M' = \begin{pmatrix} M_1' & M_2' \\ M_3' & M_4' \end{pmatrix}$ in the row of numbers

$$M_1' \| M_2' \| M_3' \| M_4'$$

and then followed by replacement of decimals M_i' by their 129-bit binary representations m_i':

$$m_1' \| m_2' \| m_3' \| m_4' = (m')_2.$$

4.3.5 Compute

$$h((m')_2) = g.$$

4.3.6 Do not accept the signature if

$$g \neq h(R(m)).$$

4.3.7 Since

$$h((m')_2) = h(R(m)),$$

then with overwhelming probability it follows that

$$(m')_2 = R(m).$$

4.3.8 Since

$$R(m) = ((m)_2 \oplus h((m)_2)) \| h((m)_2) = (m')_2 = left\ half \| right\ half$$

then it follows that

$$left\ half = (m)_2 \oplus h((m)_2),$$
$$right\ half = h((m)_2).$$

4.3.9 Finally restore the bit string $(m)_2$ by computing *XOR* of the left and right halves of the bit string $(m')_2$:

$$h((m)_2) \oplus ((m)_2 \oplus h((m)_2)) = (m)_2.$$

Theorem. Signature verification is correct.

Proof. Since the matrices *Y, U, V, W* belong to the Abelian group *G*, they mutually commute, and then we have:

$$M' = sz = (\gamma^{-1}M(WY)^{-1}L^{-1}(WY))((V^{-1}W)^{-1}r(V^{-1}W)) =$$
$$= (\gamma^{-1}M(YW)^{-1}L^{-1}WY)((V^{-1}W)^{-1}(\gamma(VY)^{-1}LVY)V^{-1}W) =$$
$$= (\gamma^{-1}\gamma)(M(YW)^{-1}L^{-1}YW)((YW)^{-1}LYW) = M.$$

Note. The numbers M_i' in 4.3.4, as follows from the proof of the *Theorem,* are the 128-bit numbers.

5 Attacks against MMDS

5.1 Key-only Attacks

In these attacks against MMDS an adversary knows only the signer's master public key $U = V^{-1}W$ and his aim is to find the signer's private key *(V, W)*. Crucial problem is the following:

5.1.1 Matrix modular factorization problem

Let *A* be any matrix from the group $GL_2(\mathbf{Z}_n)$ and consider the fixed factorization

$$A = XY,$$

X and *Y* are unknown matrices from the group $GL_2(\mathbf{Z}_n)$.

Find the unknown matrices *X, Y*.

a) Scalar case

Modular factorization problem

Let *a* be any unit modulo *n*, $a \in \mathbf{Z}_n^*$ and consider the fixed factorization

$$a = xy,$$

x and *y* are unknown units from the unit group $\mathbf{Z}_n^*$.

Find unknown units *x, y*.

The principal difference of this modular factorization from the integer factorization is as follows. Canonical integer factorization (the product of powers of prime numbers) is only up to invertible factors, that is which for the ring $\mathbf{Z}$ of integers is +1 or -1. To the residue ring situation is fundamentally different - there are at least $\varphi(n)$ (φ is Euler function) different modular factorizations, which can be obtained as follows. Let *a* be any unit modulo *n*, then select any other unit *b* modulo *n* and compute $c = b^{-1}a(\bmod n)$, we have the modular factorization $a = bc$. We call thus obtained factorization as trivial modular factorization and there are at least $\varphi(n)$ trivial modular factorizations. Let us now compare the complexity of solving the integer factorization and the modular factorization problems.

There are the subexponential algorithms to find the canonical integer factorization and is now using them can be found a canonical factorization for 512-bit and maybe the 1024-bit integer. Since there is no canonical modular factorization and modular factoring all indistinguishable, there is no algorithm other than an exhaustive search, to find a fixed modular factorization. This means that it is infeasible to find the solution of the modular factorization problem even for relatively small modulus, namely, for example, for a 256-bit modulus and maybe the 128-bit modulus.

b) General case

Trivial modular matrix factorization is defined similarly to the scalar case. Again, similar to the scalar case there is no algorithm for finding solutions of the modular matrix factorization, other than exhaustive search. Suppose *m* is the order of the group $GL_2(\mathbf{Z}_n)$, then there are at least *m* trivial modular matrix factorizations over $GL_2(\mathbf{Z}_n)$. Note that for two special cases of the modulus *n*, namely, $n = p^k$ and $n = pq$, *p, q* are primes, *k* is a positive integer, we have for *m* the following formulas.

Note 1. (i) In the case where *p* - prime, *i* - an integer, we have [8]:

$$m_{p^i} = f\left(p^i\right) = \left|GL_2\left(\mathbb{Z}_{p^i}\right)\right| = p^{4i-3}\left(p-1\right)^2\left(p+1\right) \qquad (*)$$

(ii) As a result of (i), in the case where *p* and *q* are primes, we have:

$$m_{pq} = f\left(pq\right) = \left|GL_2\left(\mathbb{Z}_{pq}\right)\right| = p\left(p-1\right)^2\left(p+1\right)q\left(q-1\right)^2\left(q+1\right). \qquad (**)$$

We now estimate how much you can reduce the bit length of the modulus of residue ring, analogous to the scalar case exhaustive search of modular matrix factorizations it remained infeasible.

Let *n* be a 32-bit integer, implemented in two ways.

1) $n = p^2$, *p* is 16-bit prime.

According to the formula (*) in this case *m* is 128-bit number.

2) $n = pq$, *p, q* are 16-bit primes.

According to the formula (**) in this case *m* is 128-bit number.

Thus, for a 32-bit modulus *n* for both cases of its construction, an exhaustive search is computationally infeasible, and therefore the modular matrix factorization problem has no solution in practice.

Let us now turn directly to the attacks on the MMDS. The adversary knows only the signer's public key $U = V^{-1}W$ and wants to find the signer's secret key (*V, W)*. For this, he is looking for the solution of a modular matrix factorization problem. At each step of an exhaustive search the adversary must decide whether it is right modular matrix factorization. The criterion for this decision is the answer to the question:

if for any matrix $L \in GL_2(\mathbf{Z}_n)$ and $Y \in G$ generate signature on the message *m* with this pair of matrices as a secret key, whether the Alice's public key *U* for some selection of the matrix *L* and matrix *Y* successfully works in the signature verification algorithm. Of course, you can restrict a probabilistic criterion and did not give an answer for any matrix *L*, but only for a relatively small number of random selection of the matrix *L*. But on the other hand, when using probabilistic criteria are more likely to give the wrong answer to a true factoring due to the lack of true matrix *L* among randomly selected matrices. Thus, if we take a 32-bit amount of random matrices as the matrix *L*, then for the 32-bit modulus *n* of residue ring (all the more for larger modulus) is computationally infeasible to find the signer's private key.

5.1.2 Existential forgery attack

An adversary might attempt to forge Alice's signature on the message *m* by selecting a random matrix $X \in GL_2(\mathbf{Z}_n)$ and computing $r' = \sigma X^{-1} N X$, trying to choose unit $\sigma \in \mathbf{Z}_n^*$ and matrix $N \in GL_2(\mathbf{Z}_n)$ so as to ensure the successful completion of the verification algorithm by performing equality $v' = \delta t$, where v' is computed by adversary as

$$v' = \mu t + U^{-1} r' U .$$

Assuming that Alice wants to select the modulus of residue ring so that the problem of a modular matrix factorization has been computationally infeasible, the multipliers of matrix μt (Alice's session public key) can not be known to the adversary. But not knowing μ and *t* he will have to limit ourselves to a random selection of σ and *N*, therefore for the selection of the modulus *n* as 32-bit integer the probability of success of the existential forgery attack is negligible.

6 Comparing the Performance of MMDS and RSA, DSA Signatures

For comparing the bit complexity of generation and verification algorithms considered in the paper digital signature schemes let us start with the known estimates of the bit complexity of basic operations in the residue ring ([7], Table 2.5).

Table 2.5. Bit complexity of basic operations in $\mathbf{Z}_n$

Operations	Bit complexity	
Modular addition	$(a+b) \bmod n$	$O(\lg n)$
Modular subtraction	$(a-b) \bmod n$	$O(\lg n)$
Modular multiplication	$(ab) \bmod n$	$O((\lg n)^2)$
Modular inversion	$a^{-1} \bmod n$	$O((\lg n)^2)$
Modular exponentiation	$a^k \bmod n, k < n$	$O((\lg n)^3)$

We now find the bit complexity of modular matrix operations used in compared digital signature schemes.

6.1 Matrix Modular Multiplication

This operation consists of 8 modular multiplications and 4 modular additions. Considering only the multiplications, we obtain the following estimate of the bit complexity of the matrix modular multiplication:

$8(\lg n)^2 C$ -bit operations for some constant *C*.

Using parallel computations at least of 4-core processor provides the following estimate:

$2(\lg n)^2 C$ -bit operations for some constant *C*.

For 32-bit *n* we obtain the following estimate:

$8(32)^2 = 8 \cdot 2^{10} \approx 8 \cdot 10^3 C$ -bit operations for some constant *C*.

In parallel computing case we have approximately:

$2(32)^2 = 2 \cdot 2^{10} \approx 2 \cdot 10^3 C$ -bit operations for some constant *C*.

For 64-bit *n* we obtain the following estimate:

$8(64)^2 = 2^{15} = 32 \times 2^{10} \approx 3,2 \times 10^4 C$ -bit operations for some constant *C*.

In parallel computing case we have approximately:

$8 \times 10^3 C$ -bit operations for some constant *C*.

6.2 Matrix Modular Multiplication in the Group *G*

This operation consists of 4 modular multiplications and 2 modular additions. Considering only the multiplications, we obtain the following estimate of the bit complexity of the matrix modular multiplication in the group *G*:

$4(\lg n)^2 C$ -bit operations for some constant *C*.

In parallel computing case we have:

$(\lg n)^2 C$ -bit operations for some constant *C*.

For 32-bit *n* we obtain the following estimate:

$4(32)^2 = 4 \cdot 2^{10} \approx 4 \cdot 10^3 C$ -bit operations for some constant *C*.

In parallel computing case we have:

$32^2 = 2^{10} \approx 10^3 C$ -bit operations for some constant *C*.

For 64-bit *n* we obtain the following estimate:

$4(64)^2 = 2^{14} = 16 \times 2^{10} \approx 1,6 \times 10^4 C$ -bit operations for some constant *C*.

In parallel computing case we have approximately:

$4\times10^3 C$ -bit operations for some constant *C*.

6.3 Modular Matrix Multiplication by a Scalar

This operation consists of 4 modular multiplications, then we obtain the following estimate of the bit complexity of the matrix modular multiplication by a scalar:

$4(\lg n)^2 C$ -bit operations for some constant *C*.

In parallel computing case we have:

$(\lg n)^2 C$ -bit operations for some constant *C*.

For 32-bit *n* we have:

$4(32)^2 = 4\cdot 2^{10} \approx 4\cdot 10^3 C$ -bit operations.

In parallel computing case we have:

$32^2 = 2^{10} \approx 10^3 C$ -bit operations.

For 64-bit *n* we obtain the following estimate:

$4(64)^2 = 4\times 2^{12} = 16\times 2^{10} \approx 1{,}6\times 10^4 C$ -bit operations for some constant *C*.

In parallel computing case we have approximately:

$4\times10^3 C$ -bit operations for some constant *C*.

6.4 Modular Matrix Multiplication by a Scalar in the Group *G*

Modular matrix multiplication by a scalar in the group *G* consists of 2 modular multiplications then we have:

$2(\lg n)^2 C$ -bit operations.

In parallel computing case we have:

$(\lg n)^2 C$ -bit operations.

For 32-bit *n* we have:

$2(32)^2 = 2\cdot 2^{10} \approx 2\cdot 10^3 C$ -bit operations.

In parallel computing case we have:

$$32^2 = 2^{10} \approx 10^3 C \text{ -bit operations.}$$

For 64-bit *n* we have:

$$2(64)^2 = 2 \cdot 2^{12} = 8 \cdot 2^{10} \approx 8 \cdot 10^3 C \text{ -bit operations.}$$

In parallel computing case we have:

$$64^2 = 2^{12} = 4 \cdot 2^{10} \approx 4 \times 10^3 C \text{ -bit operations.}$$

6.5 Matrix Modular Inversion

This operation consists of 2 modular multiplications and 1 modular subtraction (computation of the determinant), modular inversion and matrix modular multiplication by a scalar, then we obtain the following estimate of the bit complexity of the matrix modular inversion:

$$3(\lg n)^2 + 4(\lg n)^2 = 7(\lg n)^2 C \text{ -bit operations (modular subtraction is ignored).}$$

In parallel computing case we have:

$$2(\lg n)^2 C \text{ -bit operations (modular subtraction is ignored).}$$

For 32-bit *n* we obtain the following estimate:

$$7(32)^2 = 7 \cdot 2^{10} \approx 7 \times 10^3 C \text{ -bit operations for some constant } C.$$

In parallel computing case we have:

$$2 \cdot 32^2 \approx 2 \times 10^3 C \text{ -bit operations.}$$

For 64-bit *n* we obtain the following estimate:

$$7(64)^2 = 7 \times 2^{12} = 28 \times 2^{10} \approx 2{,}8 \times 10^4 C \text{ -bit operations for some constant } C.$$

In parallel computing case we have approximately:

$$2(64)^2 = 2 \cdot 2^{12} = 8 \cdot 2^{10} \approx 8 \times 10^3 C \text{ -bit operations for some constant } C.$$

6.6 Matrix Modular inversion in the Group *G*

Recall that in group *G* matrices are of the form $D = \begin{pmatrix} a & b \\ b & a \end{pmatrix}$, then we obtain:

$$3(\lg n)^2 + 2(\lg n)^2 = 5(\lg n)^2 C \text{ -bit operations.}$$

In parallel computing case we have:

$2(\lg n)^2 C$ - bit operations.

For 32-bit *n* we obtain the following estimate:

$5 \cdot 32^2 \approx 5 \times 10^3 C$ -bit operations.

In parallel computing case we have approximately:

$2 \cdot 32^2 \approx 2 \times 10^3 C$ -bit operations.

For 64-bit *n* we obtain the following estimate:

$5(64)^2 = 5 \cdot 2^{12} = 20 \cdot 2^{10} \approx 2 \times 10^4 C$ -bit operations for some constant *C*.

In parallel computing case we have approximately:

$2 \cdot 64^2 = 2 \cdot 2^{12} \approx 8 \times 10^3 C$ - bit operations.

Now we can compare the bit complexity of the signatures generation and verification algorithms discussed in the paper digital signatures and used now in practice signatures of RSA and DSA.

We first consider as a reference point the signature of RSA. Signature generation and verification in this case consists of only modular exponentiation, then their bit complexity are $(\lg n)^3 C$ -bit operations. For 1024-bit *n* and 1024-bit plaintext block we obtain the following estimate:

$1024^3 = (2^{10})^3 \approx (10^3)^3 = 10^9 C$ -bit operations for some constant *C*.

Note. Generation of RSA signature may be faster with a small public exponent, but on the other hand a small exponent contains a security threats. Moreover, since the RSA is a signature with the message recovery, to protect against existential forgery attack need to use some redundancy function [7], namely, using the ISO/IEC 9796. This makes the implementation of RSA signature in practice cumbersome and slow procedure.

Another reference point considers fast ciphers, namely, symmetric or stream ciphers. It is known that these ciphers are much faster than public-key cryptosystems, according to some estimates, about 10000 times faster. So there is a very roughly estimate of the bit complexity of fast ciphers as a range of$10^3 - 10^4$ -bit operations. Now let's see whether the MMDS falls in this range.

6.7 DSA

Following the description of the DSA in [9], we first find the bit complexity of individual parts, and then of the full cycle of the DSA application.

6.7.1 Key generation

Let *p* be 1024-bit prime number, $\mathbf{Z}_p^*$ be a multiplicative group of the residue field $\mathbf{Z}_p$ and *z* be a generator of the cyclic group $\mathbf{Z}_p^*$. Suppose further that *p-1* has the form $p-1=2^l q$, where *l* is an positive integer and *q* is a 160-bit prime number. Since *z* is of order *p-1* in the group $\mathbf{Z}_p^*$, the element $z^{\frac{p-1}{q}}$ is of order *q* and therefore is a generator of the cyclic subgroup $G=\left\langle z^{\frac{p-1}{q}} \right\rangle$ of the order *q* in the group $\mathbf{Z}_p^*$. To denote $z^{\frac{p-1}{q}}$ by *g* and to choose a random 160-bit integer *x*, we obtain the public key *y* by computing

$$y = g^x \bmod p$$

and *x* is the session key. Now we find the bit complexity of key generation phase.

One modular exponentiation modulo 1024-bit prime *p* needs $(\lg p)^3 C$ bits operations, in our case $1024^3 C \approx (10^3)^3 C = 10^9 C$ -bits operations for some constant C.

6.7.2 Signature generation

Alice does the following:

1. Computes hash value *h* (*m*).
2. Selects random 160-bit integer *k*.
3. Computes signature *(r, s)*, where

$$r = (g^k \bmod p) \bmod q,\ s = ((h(m) + xr)k^{-1}) \bmod q.$$

One modular exponentiation modulo 1024-bit prime *p* and one modular reduction modulo 160-bit prime *q* needs $((\lg p)^3 + (\lg q)^2)C \approx (\lg p)^3 C$ -bits operations, in our case

$1024^3 C \approx (10^3)^3 C = 10^9 C$ -bits operations for some constant *C*.

6.7.3 Signature verification

Bob does the following:

1. Computes hash value $h(m)$.
2. Computes

$$a = h(m)s^{-1} \bmod q,\ b = rs^{-1} \bmod q.$$

3. Computes

$$v = (g^a y^b \bmod p) \bmod q.$$

4. Signature is accepted if and only if *v=r*.

Two modular exponentiations modulo 1024-bit prime number *p* need $2(\lg p)^3 C$ bits operations, in our case

$2\times 1024^3 C \approx 2\times (10^3)^3 C = 2\times 10^9 C$ -bits operations for some constant *C*.

A complete cycle of the DSA application thus requires $4(\lg p)^3 C$ bit operations, in our case

$4\times 10^9 C$ -bit operations for some constant *C*.

6.8 MMDS

6.8.1 Key generation

One matrix modular multiplication and one matrix modular inversion in the group *G* need $4(\lg n)^2 + 5(\lg n)^2 = 9(\lg n)^2 C$ -bit operations.

In the parallel computing case we have:

$(\lg n)^2 + 2(\lg n)^2 = 3(\lg n)^2 C$ -bit operations.

1) For 32-bit *n* we have the following estimate of the bit complexity of MMDS key generation:

$9(32)^2 = 9\times 2^{10} \approx 9\times 10^3 \approx 10^4$ C-bit operations.

In the parallel computing case for 32-bit *n* we have:

$3(32)^2 = 3\cdot 2^{10} \approx 3\times 10^3$ C-bit operations.

2) For 64-bit *n* we have the following estimate:

$9(64)^2 = 9\cdot 2^{12} = 36\cdot 2^{10} \approx 36\cdot 10^3 = 3,6\times 10^4$ C-bit operations.

In the parallel computing case for 64-bit *n* we have:

$3(64)^2 = 3\cdot 2^{12} = 12\cdot 2^{10} \approx 12\cdot 10^3 = 1,2\times 10^4 C$ -bit operations.

6.8.2 Signature generation

When generating a signature used four modular matrix multiplications, one modular matrix inversion and one modular multiplication (addition is not included). In addition, used the operations in the group *G*: there are two modular matrix multiplications, two modular matrix inversions and one modular matrix multiplication by a scalar. Using estimates of the bit complexity of these operations made above, we get:

$(32(\lg n)^2 + 7(\lg n)^2 + (\lg n)^2) + (8(\lg n)^2 + 10(\lg n)^2 + 2(\lg n)^2) = 60(\lg n)^2 C$ -bit operations.

In the parallel computing case we get:

$$(8(\lg n)^2 + 2(\lg n)^2 + (\lg n)^2) + (2(\lg n)^2 + 4(\lg n)^2 + (\lg n)^2) = 18(\lg n)^2 C \text{ -bit operations.}$$

1) For 32-bit *n* we have the following estimate of the bit complexity of MMDS signature generation:

$$60 \cdot 32^2 = 60 \cdot 2^{10} \approx 60 \cdot 10^3 = 6 \times 10^4 C \text{ -bit operations.}$$

In the parallel computing case for 32-bit *n* we have:

$$18 \cdot 32^2 = 18 \cdot 2^{10} \approx 18 \cdot 10^3 = 1{,}8 \times 10^4 C \text{ -bit operations.}$$

2) For 64-bit *n* we have the following estimate:

$$60 \cdot 64^2 = 60 \cdot 2^{12} = 240 \cdot 2^{10} \approx 240 \cdot 10^3 = 2{,}4 \times 10^5 C \text{ -bit operations.}$$

In the parallel computing case for 64-bit *n* we have:

$$18 \cdot 64^2 = 72 \cdot 2^{10} \approx 72 \cdot 10^3 = 7{,}2 \times 10^4 C \text{ -bit operations.}$$

6.8.3 Signature verification

Under verifying the signature used two modular matrix multiplications, one modular matrix inversion in the group *G* and one modular matrix multiplication by a scalar. Using estimates of the bit complexity of these operations made above, we get:

$$16(\lg n)^2 + 5(\lg n)^2 + 4(\lg n)^2 = 25(\lg n)^2 C \text{ -bit operations.}$$

In the parallel computing case we get:

$$4(\lg n)^2 + 2(\lg n)^2 + (\lg n)^2 = 7(\lg n)^2 C \text{ -bit operations.}$$

For more specific values of the *n* bit length specify the appropriate values of the bit complexity of the MMDS application.

1) 32-bit length of *n*:

$$25(32)^2 = 25 \cdot 2^{10} \approx 25 \times 10^3 = 2{,}5 \times 10^4 C \text{ -bit operations;}$$

in the parallel computing case for 32-bit *n* we have:

$$7(32)^2 = 7 \cdot 2^{10} \approx 7 \cdot 10^3 C \text{ -bit operations.}$$

2) 64-bit length of *n*:

$$25(64)^2 = 25 \cdot 2^{12} = 100 \cdot 2^{10} \approx 100 \cdot 10^3 = 10^5 C \text{ -bit operations;}$$

in the parallel computing case for 64-bit *n* we have:

$$7(64)^2 = 7 \cdot 2^{12} = 28 \cdot 2^{10} \approx 28 \cdot 10^3 = 2{,}8 \times 10^4 C \text{ -bit operations.}$$

7 Conclusion

As a result, the consideration in section 5 of potential attacks on MMDS, use a 32-bit modulus *n* may be considered secure. In case of detection of vulnerabilities there is strengthening the protection by increasing the bit length of the modulus *n*. On the other hand, for a 32-bit modulus *n* as estimates show the bit complexity of key generation, generation and verification of signature, these steps of MMDS are much faster than the corresponding steps of the digital signatures used in practice nowadays. Moreover, if applied to parallel processing of at least 4-core processor, as seen from the results of section 6, the speed of the three steps MMDS approaching speeds of symmetric and stream ciphers. The complete cycle of the application of MMDS (key generation, the generation and verification of signatures) in this case requires only approximately $2{,}8\times10^4 C$ - bit operations versus $4\times10^9 C$ -bit operations in DSA or $2\times10^9 C$ -bit operations in RSA signature.

Competing Interests

Author has declared that no competing interests exist.

References

[1] Available: www.dwavesys.com/d-wave-two-system

[2] Shor PW. Algorithms for quantum computation: Discrete logarithm and factoring. Proceedings of the IEEE 35th Communications Annual Symposium on Foundation of Computer Science. Santa Fe, NM: Springer-Verlag. 1994;124-134.

[3] Available: https://www.openssl.org/docs/crypto/crypto.html

[4] Rososhek SK. New practical algebraic public-key cryptosystem and some related algebraic and computational aspects. Applied Mathematics. 2013;4(7):1043-1049.

[5] Rososhek SK. Modified matrix modular cryptosystems. British Journal of Mathematics and Computer Science. 2015;5(5):613-636.

[6] Rososhek SK, Gorbunov ES. Non-commutative analogue of Diffie-Hellman protocol in matrix ring over the residue ring. International Journal of Computers and Technology. 2013;11(10):3051-3059.

[7] Menezes A, van Ooshot P, Vanstone S. Handbook of applied cryptography. CPC Press; 1996.

[8] Kargopolov MI, Merzliyakov YI. Foundations of the group theory. Moskow, Nauka (Russian); 1977.

[9] Smart N. Cryptography: An introduction. McGraw-Hill; 2003.

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/12382