UDC 004.056.55

Cryptanalysis of the «Vershyna» Digital Signature Algorithm

Yuliia Lytvynenko[1], Andrii Fesenko[1]

[1]*National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Institute of Physics and Technology*

Abstract

The CRYSTALS-Dilithium digital signature algorithm, which was selected as the prototype of the new «Vershyna» digital signature algorithm, is analyzed in this paper. The characteristics of the National Digital Signature Standard Project and the construction of the «Vershyna» algorithm are also presented. During the analysis of the project, the predicted number of iterations that the algorithm must perform to create the correct signature was calculated. In addition, basic theoretical information about the structure of Fiat-Shamir with aborts and its security in quantum and classical models oracle models is also provided. We obtain our own results on the resistance of the «Vershyna» algorithm to the attack without the use of a message in classical and quantum oracle models. The resistance of the «Vershyna» algorithm to a key recovery attack is based on the assumption of the hardness of the MLWE problem, and the resistance to existential signature forgery is based on the assumption of the hardness of the MSIS problem. In this work, the expected level of hardness of SIS and LWE problems is calculated, to which there are reductions from MSIS and MLWE problems.

Keywords: «Vershyna» algorithm, digital signature algorithm, lattice problems, security analysis

1. Introduction

It is well known that the security of most asymmetric cryptosystems today is based on the complexity of solving discrete logarithm and integer factorization problems. There are currently no known polynomial algorithms for solving these problems on classical computers. However, in 1994, the American mathematician Peter Shor demonstrated that these problems could be effectively solved in the quantum oracle model [1]. Consequently, with the advent of powerful quantum computers, most of today's asymmetric cryptosystems can be compromised.

In December 2016, NIST (US National Institute of Standards and Technology) officially launched the process of developing and standardizing new post-quantum public key cryptographic algorithms. Currently, experts from around the world are actively developing and verifying the security of candidates for post-quantum cryptosystems.

Recently, a project of a new National Digital Signature Standard was proposed, including the «Vershyna» digital signature algorithm. The CRYSTALS-Dilithium digital signature algorithm, one of the algorithms selected in the NIST competition, was used as a prototype for this project. The goal of this work is to evaluate the security of the proposed digital signature algorithm, as well as to examine the component algorithms and the modes of operation of the standard's digital signature scheme.

2. Preliminaries

In the current era, cryptographic primitives whose security is based on the hardness of complex problems on lattices are considered as potential replacements for existing cryptosystems, as we move towards post-quantum algorithms. Typically, the problems of finding the shortest vector (SVP), finding the short integer solution (SIS), and learning with errors (LWE) [2] are chosen as problems whose complexity will determine the security of the system. Compared to other analogues with similar security conditions, primitives from this category are considered the most promising and efficient.

Since «Vershyna» is a digital signature algorithm that uses algebraic lattices, it is crucial to define the concept of a lattice and its associated definitions.

m-dimensional lattice is a discrete additive subgroup of the group $\mathbb{R}^m$, which is represented as a set of integer linear combinations of n linearly independent vectors $\{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$. That is, $\mathfrak{L}(\mathbf{B}) = \left\{ \sum_{i=1}^{n} x_i \mathbf{b}_i \mid x_i \in \mathbb{Z},\ \forall i \in [1, n] \right\}$, where $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n]$ is called the *lattice basis* $\mathfrak{L}$, and n — *lattice rank*.

The determinant of the lattice $\mathfrak{L}$ is called the value: $det(\mathfrak{L}) = \sqrt{Det(\mathbf{B}^T\mathbf{B})}$, where Det is the determinant of the matrix.

Similar to most lattice-based schemes that use appropriate operations on polynomial rings, the creators of the Dilithium algorithm chose a ring that allows efficient implementation of multiplication by the Number-Theoretic Transformation (NTT). NTT is essentially a subset of the Discrete Fourier Transform (FFT), which is used when working with finite fields. To use NTT, it is necessary to choose a prime number q such that the group $\mathbb{Z}_q^*$ contains an element of order $2n$. In this scenario, the most time-consuming operations in terms of execution time will be the NTT transformation and its inverse.

When constructing compact signature schemes using lattices, a discrete Gaussian distribution is often used. Generating secure random sequences that are protected against side-channel attacks is a challenging task and can lead to insecure implementations. Therefore, the developers of the Dilithium algorithm avoided using of a discrete Gaussian distribution and preferring a uniform distribution instead. This distribution is also used in the «Vershyna» algorithm. Examples of how to estimate the distance between two distributions can be found in [3].

The authors of the «Vershyna» algorithm used the Fiat-Shamir structure with aborts [4]. One of its notable features is that it requires the generation of a nonce, a one-time value, for signing. It is one of the most widely used paradigms in the construction of post-quantum digital signature schemes. The Fiat-Shamir transform combines a canonical identification scheme ID and a hash function H to obtain a digital signature scheme $FS = FS[ID, H]$.

2.1. Security

The standard principle for ensuring the security of digital signatures is security in the UF-CMA (Unforgeability under Chosen-Message Attack): protection against attacks using a chosen message. In this model, the attacker has access to the public key and can use the signing oracle to sign any message of his choice. The goal of the attacker is to find the correct signature of a new message without using an oracle. A slightly stronger security requirement, which is also useful in some cases, is SUF-CMA (Strong Existential Unforgeability under Chosen-Message Attack). In this model, the attacker succeeds if he manages to create a different signature for the already chosen message.

It can be shown that in the random oracle model (ROM), the «Vershyna» algorithm, as well as Dilithium, exhibit security against SUF-CMA attacks due to the complexity of solving standard problems on lattices, namely the MLWE problem (which is a generalization of the LWE and Ring-LWE problems) and the MSIS problem (which is a generalization of the SIS and Ring-SIS problems). It is also important to consider the security of the scheme in the quantum random oracle model (QROM) where an adversary can send requests to the oracle to compute the hash function using quantum superposition of the input data. While the classical proof of security relies on the "forking lemma" [5], this reduction is generally not applicable in the QROM model.

The security of the «Vershyna» and Dilithium algorithms against key recovery attacks is based on the complexity of the MLWE problem, while their security against message forgery is based on the complexity of the SelfTargetMSIS problem and their security against existential signature forgery is based on the complexity of the MSIS problem [6]. The SelfTargetMSIS problem combines the MSIS problem with a cryptographic hash function H.

Formal description of MLWE, MSIS and SelfTargetMSIS problems For integers m, k and probability distribution $D\colon \mathbb{R}_q \to [0, 1]$ *advantage function of algorithm A in solving the $MLWE_{m,k,D}$ problem over the ring $\mathbb{R}_q$* has a

value equal to

$$Adv^{MLWE}_{m,k,D} := |Pr[A(\mathbf{A},\mathbf{t}) = 1 \mid \mathbf{A} \leftarrow \mathbb{R}_q^{m\times k}, \mathbf{t} \leftarrow \mathbb{R}_q^m] - Pr[A(\mathbf{A}, \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2) = 1 \mid \mathbf{A} \leftarrow \mathbb{R}_q^{m\times k}, \mathbf{s}_1 \leftarrow D^k, \mathbf{s}_2 \leftarrow D^m]|.$$

The $Adv^{MSIS}_{m,k,\gamma}(A)$ — *advantage function of algorithm A in solving the* $MSIS_{m,k,\gamma}$ *problem over the ring* $\mathbb{R}_q$

$$Adv^{MSIS}_{m,k,\gamma}(A) := Pr[(0 < \|\mathbf{y}\|_\infty \le \gamma) \wedge \wedge [\mathbf{I} \mid \mathbf{A}] \cdot \mathbf{y} = 0 \mid \mathbf{A} \leftarrow \mathbb{R}_q^{m\times k}, \mathbf{y} \leftarrow A(\mathbf{A})].$$

Suppose that $H\colon \{0,1\}^* \to B_\tau$ is a cryptographic hash function. Here, B_τ is the set from which the challenge polynomials c are chosen. All coefficients of the polynomial take values from the set $\{-1,\ 0,\ 1\}$, and each polynomial contains exactly τ non-zero coefficients. The $Adv^{SelfTargetMSIS}_{H,m,k,\gamma}(A)$ — *advantage function of the algorithm A in solving the* $SelfTargetMSIS_{H,m,k,\gamma}$ *problem over the ring* $\mathbb{R}_q$

$$Adv^{SelfTargetMSIS}_{H,m,k,\gamma}(A) := Pr[(\|\mathbf{y}\|_\infty \le \gamma) \wedge \wedge H([\mathbf{I} \mid \mathbf{A}] \cdot \mathbf{y} \| M) = c \mid \mathbf{A} \leftarrow \mathbb{R}_q^{m\times k}; \left(y := \begin{bmatrix} r \\ c \end{bmatrix},\ M\right) \leftarrow A^{|H>}(\mathbf{A})].$$

If the attacker A has access to the oracle H in the classical model, then there is a reduction that uses the forking lemma to prove that

$$Adv^{SelfTargetMSIS}_{H,m,k,\gamma}(B) \approx \sqrt{Adv^{MSIS}_{m,k,2\gamma}(A)/Q_H},$$

where Q_H is the number of classical queries to oracle H. This reduction is standard and implicit in (classical) proofs of digital signature protection, whose security is based on the complexity of the MSIS problem.

The security of the Dilithium algorithm in the QROM model can be expressed as follows:

$$Adv^{sUF-CMA}_{Dilithium}(A) \le Adv^{MLWE}_{k,l,D}(B) + Adv^{SelfTargetMSIS}_{H,k,l+1,\zeta}(C) + Adv^{MSIS}_{k,l,\zeta'} + 2^{-\alpha+1},$$

where D is a uniform distribution over S_η,

$$\zeta = max\{\gamma_1 - \beta, 2\gamma_2 + 1 + 2^{d-1}w_c\} \le 4\gamma_2,$$

$$\zeta' = max\{2(\gamma_1 - \beta), 4\gamma_2 + 2\} \le 4\gamma_2 + 2,$$

where w_c is the number of ± 1 in the polynomial $c \in B_\tau$.

There is also another method that can be used to evaluate the security of a digital signature scheme — **challenge polynomial entropy** [3].

In the «Vershyna» algorithm a strategy is presented in which the probability of deviation is reduced in order to achieve better efficiency. This is achieved by varying the number of non-zero coefficients in the challenge polynomial c, depending on the security level. The set B_τ, which is used to select the polynomial c, contains polynomials with exactly τ coefficients of either 1 or -1, and $n - \tau$ zero coefficients. Since all coefficients of the challenge polynomial belong to the set $\{-1,\ 0,\ 1\}$, the entropy of the polynomial is $\log\binom{256}{\tau} + \tau$ bits.

If the value of $c' \in B_\tau$ is fixed, an attacker can create a forged signature by choosing a value of z' that satisfies the condition

$$\|z'\|_\infty < \gamma_1 - \beta,$$

and then checking for equality

$$c' = H(M \| HighBith(\mathbf{A}z' - c'\mathbf{t}, 2\gamma_2)).$$

Here, $HighBits_q$ is a function that selects the highest part of a polynomial, H is a hash function, M is a message, $\beta,\ \gamma_1$ and γ_2 are parameters of the «Vershyna» algorithm.

As the entropy of z' surpasses that of c', the attack's time complexity in the classical model (ROM) will be $\mathcal{O}\left(\log\binom{256}{\tau} + \tau\right)$. Moreover, Grover's algorithm [7] can be used to obtain a quadratic speedup with time complexity $\mathcal{O}\left(\sqrt{\log\binom{256}{\tau} + \tau}\right)$ in the QROM, by considering the equation for c' as a function of z'.

3. Comments on the National Standard project with «Vershyna» algorithm

The National Standard project with the algorithm «Vershyna» consists of the following components:

1) A section containing scope, regulatory citations, acronyms and abbreviations, and general provisions.
2) Algorithms used to generate asymmetric key pairs and to create and validate a digital signature. All algorithms are presented in pseudo code.
3) System and additional parameters along with the purpose and formulas for calculations.
4) Precalculations for NTT transformations (direct and inverse) for values $n = 256$ and $n = 512$.

5) Test vectors for the hash function (DSTU 7564:2014), for generating pseudo-random sequences (DSTU 8845:2019), for key generation (the first and last components of the matrix $\mathbf{A}$, the vectors $\mathbf{s}_1$, $\mathbf{s}_2$, $\mathbf{t}_0$, $\mathbf{t}_1$ and actually the asymmetric keys themselves), for digital signature generation and validation. All test vectors are available for 4 modes of operation of the algorithm.
6) References.

Let's outline the main provisions and some differences of the «Vershyna» algorithm from Dilithium.

3.1. Options

As already mentioned, the authors of the «Vershyna» algorithm project modified the Dilithium algorithm and added 2 sets of parameters for new security levels. The first 2 sets of parameters are borrowed from the Dilithium algorithm. Thus, the «Vershyna» algorithm offers 4 modes of operation depending on the desired level of cryptographic security. A proper security analysis is required for these additional parameter sets.

We will clarify the parameters of the algorithm «Vershyna»:

1) n — degree of polynomials;
2) q — the module by which all coefficients of polynomials are reduced;
3) l — size of the vector s_1;
4) k — size of the vector s_2;
5) η defines the range of values for the coefficients of the polynomials that make up the vectors s_1 and s_2 that are part of the private key;
6) ω determines the maximum number of units in the matrix $\mathbf{h}$;
7) β — the parameter used to calculate the transformation rate;
8) γ_1 — the parameter used to calculate the masking vector;
9) γ_2 — a parameter used to compute the norm of transformations and to compute the low-order and high-order bits of the polynomials;
10) d — the number of bits for the set of the low-order part of the components of the vector $\mathbf{t}_0$ that is part of the public key;
11) $HASH\_OCTETS$ — the number of octets in the hash value;
12) $SEED\_OCTETS$ — the number of octets in pseudorandom sequences.

3.2. Hash functions and pseudorandom sequence generation

To obtain test vectors for the «Vershyna» algorithm, the algorithm of the national hashing standard DSTU 7564:2014 («Kupina») is used as a hash function. However, the developers themselves point out that in fact other valid algorithms can be used to compute the hash value. One such example is the SHAKE-256 hash function.

To generate pseudorandom sequences, the algorithm of the national stream encryption standard DSTU 8845:2019 («Strumok») was chosen instead of the SHAKE-128 hash function. Other approved algorithms, such as AES and SHAKE-256, can also be used to compute pseudorandom sequences. Depending on the chosen algorithm, the size of the block is determined in octets, since pseudorandom sequences in the «Vershyna» algorithm are constructed in blocks. Thus, the authors specify that the block size for the «Strumok» cipher is 128 bits, for AES — 64 bits, and for SHAKE-256 — 136 bits.

3.3. Algorithms of «Vershyna» Digital Signature

All algorithms are presented in the form of pseudo code, which is not always convenient for implementation. On the one hand, this is good for programmers. However, all implementations must conform to the pseudo code defined in the standard. This makes it impossible to choose an algorithm implementation that, for example, uses less memory or works faster. After all, different algorithm implementations are suitable for different situations and you should not limit yourself to just one. Developers have tried to choose the best algorithm, but mathematics improves and better and faster schemes for performing the same operations will appear.

In addition, some preliminary calculations are used in the «Vershyna» algorithm:

1) ρ — a random string for the matrix $\mathbf{A}$;
2) ρ_1 — random string for vectors $\mathbf{s}_1$, $\mathbf{s}_2$;
3) key — a random string that is part of the private key sk;
4) $z256$, $z256\_$, $z512$, $z512\_$ — arrays that are used for direct and inverse

NTT transformations and, accordingly, the value $n256\_ = 256^{-1} \pmod q$ and $n512\_ = 512^{-1} \pmod q$, which are also used for NTT transformations.

Preliminary calculations speed up the algorithm, but increase the amount of memory required, which can be critical on devices with limited memory.

3.4. Creating a digital signature

The «Vershyna» algorithm provides 2 variants of signature creation: one for the same message and different for the same message. This is achieved by introducing the variable $variant$ and using a random component of a certain length when choosing a variant with different signatures.

4. Analysis of characteristics and security levels of the «Vershyna» algorithm

Calculation of key and signature lengths and the expected number of iterations.

One of the most important features of a digital signature scheme is the size of the key and the calculated signature. If there are two signature schemes with the same level of security, it is better to choose the scheme with the smaller specified sizes. All sizes are measured in octets (bytes). To do this, use the function: $OCTETS(x)$ — a function that calculates the minimum number of octets sufficient to write a variable consisting of a given number of bits x.

Using the formulas (1), (2) and (3) it is possible to calculate the required values. The total size of the *public key* is:

$$PK\_OCTETS = SEED\_OCTETS + \\ + OCTETS(k \cdot n \cdot (log_2 q - d)). \quad (1)$$

The total size of the *private key* is:

$$SK\_OCTETS = 2 \cdot SEED\_OCTETS + \\ + HASH\_OCTETS + \quad (2) \\ + OCTETS(n \cdot ((l + k) \cdot log_2(2\eta + 1) + k \cdot d)).$$

The total size of the *digital signature* is:

$$DS\_OCTETS = SEED\_OCTETS + \quad (3) \\ + OCTETS(log_2(2\gamma_1) \cdot n \cdot l) + \omega + k \cdot n/256.$$

At first glance, it may seem that we have long keys and a signature. However, the method used in the Dilithium digital signature scheme and later borrowed by the authors of «Vershyna» is one of the most optimal of all.

The **expected number of iterations** needed to generate the correct signature is also important. This characteristic is calculated by the formula:

$$Exp.\ reps \approx e^{n \cdot \beta(l/\gamma_1 + k/\gamma_2)}.$$

All the results of calculations according to these formulas are given in the first block of the table 1.

4.1. Resistance of the «Vershyna» algorithm to possible attacks

We will consider the method of evaluating the resistance of the «Vershyna» algorithm to a forgery attack without using a message. That is, the entropy value of the challenge polynomial $c \in B_\tau$ is used.

Let's recall the entropy formula used to calculate the values in the second block of the table 1:

$$Entropy = \log \binom{256}{\tau} + \tau.$$

The value of entropy corresponds to the number of attack resistance bits in the ROM model. More specifically, the complexity is $\mathcal{O}(Entropy)$. In the QROM model, the complexity is reduced by the possibility of using Grover's algorithm.

As already mentioned, the resistance of the «Vershyna» algorithm to various attacks depends on the complexity of such problems as MLWE, SelfTargetMSIS, and MSIS. The $MLWE_{l,k,D}$ problem for some distribution D can be considered as a LWE problem of dimensions $n \cdot l$ and $n \cdot k$. There is also a reduction of the SelfTargetMSIS problem to the MSIS problem and thus to the SIS problem, with changing parameters.

The most famous algorithm for finding very short non-zero vectors in lattices is the Blok-Korkin-Zolotarev (BKZ) algorithm proposed by Schnorr and Eichner in 1991.

The BKZ algorithm with block size b makes calls to the algorithm that solves the problem of finding the shortest vector in the lattice of dimension b (SVP). The security of the «Vershyna» algorithm depends on the need to run the BKZ algorithm with a large block size b and on the fact that the complexity of solving the SVP problem is exponential over b. The most efficient classical algorithm for solving the SVP problem runs

Table 1
Output sizes, expected number of iterations, and security of the «Vershyna» algorithm

Mode	128/64	256/128	384/192	512/256
$pk\ size\ (bytes)$	1312	1952	4528	5824
$sk\ size\ (bytes)$	2355	3740	8673	10271
$sig\ size\ (bytes)$	2420	3293	6612	10552
$Exp.\ reps$	4.25	5.1	3.2	6.55
Classical forgery attack	194	226	386	512
Quantum forgery attack	97	113	193	256
BKZ block-size b to break SIS	417	602	1720	2318
Best Known Classical bit-cost	121	176	503	677
Best Known Quantum bit-cost	110	159	456	614
BKZ block-size b to break LWE	422	622	1496	2312
Best Known Classical bit-cost	123	181	437	676
Best Known Quantum bit-cost	111	164	396	613

in time $\approx 2^{0.292 \cdot b}$. The most efficient quantum algorithm for solving the SVP problem runs in time $\approx 2^{0.265 \cdot b}$. We can hope to improve the execution time to $\approx 2^{0.2075 \cdot b}$. Currently, no more efficient algorithm for solving the SVP problem has been found.

The third and fourth blocks of the table 1 show the results of calculations of the expected level of complexity of solving the SIS and LWE problems in the classical (Best Known Classical bit-cost) and quantum (Best Known Quantum bit-cost) computational models. An estimate of the complexity level of solving these problems by the most effective algorithm currently known (Best Plausible bit-cost) is also given. Since there are 2 types of attacks when solving the LWE problem (primal attack and dual attack), table 1 shows the smaller of the two possible values.

5. Conclusions

This work examines the design of the «Vershyna» algorithm, as well as the characteristics of the National Digital Signature Standard project. The length of the private and public keys and the signature size were evaluated. In addition, in this work, we have determined the expected number of iterations of the «Vershyna» algorithm required to create a valid signature. The security of the «Vershyna» algorithm is also analyzed. We estimated the number of blocks used in the BKZ algorithm to solve LWE and SIS problems, and the complexity of solving the relevant problems using classical and quantum computing models.

References

[1] Shor P. W. Polynomial-Time Algorithms for Prime Factorizationand Discrete Logarithms on a Quantum Computer. — 1994.

[2] Lattice Based Cryptography for Beginners. / Dong Pyo Chi, Jeong Woon Choi, Jeong San Kim, and Taewan Kim.

[3] Zhongxiang Zheng, Anyu Wang, Lingyue. Qin. Rejection Sampling Revisit: How to Choose Parameters in Lattice-Based Signature.

[4] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures.

[5] Pointcheval David, Stern Jacques. Security arguments for digital signatures and blind signatures. — 2000.

[6] Exploiting Determinism in Lattice-based Signatures - Practical Fault Attacks on pqm4 Implementations of NIST candidates. / Prasanna Ravi, Mahabir Prasad Jhanwar, James Howe, Anupam Chattopadhyay, and Shivam. Bhasin. — 2019.

[7] Lov Kumar Grover. A fast quantum mechanical algorithm for database search. — 1996.