The object of this study is digital signatures. The Falcon digital signature scheme is one of the finalists in the NIST post-quantum cryptography competition. Its distinctive feature is the use of floating-point arithmetic, which leads to the possibility of a key recovery attack with two non-matching signatures formed under special conditions. The work considers the task to improve the Falcon in order to prevent such attacks, as well as the use of fixed-point calculations instead of floating-point calculations in the Falcon scheme. The main results of the work are proposals for methods on improving Falcon's security against attacks based on the use of floating-point calculations. These methods for improving security differ from others in the use of fixed-point calculations with specific experimentally determined orders of magnitude in one case and proposals for modifying procedures during the execution of which the conditions for performing an attack on implementation level arise in the second case. As a result of the analysis, the probability of a successful attack on the recovery of the secret key for the reference implementation of the Falcon was clarified. Specific places in the code that make the attack possible have been localized and code modifications have been suggested that make the attack impossible. In addition, the necessary scale for fixed-point calculations was determined, at which it is possible to completely get rid of floating-point calculations. The results could be used to qualitatively improve the security of existing digital signatures. This will make it possible to design more reliable and secure information systems using digital signatures. In addition, the results could be implemented in existing systems to ensure their resistance to modern threats

Keywords: quantum-resistant transformations, Falcon, floating point, NIST PQC, NTRU

UDC 004.056.5
DOI: 10.15587/1729-4061.2024.310521

IMPROVING PROTECTION OF FALCON ELECTRONIC SIGNATURE SOFTWARE IMPLEMENTATIONS AGAINST ATTACKS BASED ON FLOATING POINT NOISE

Olena Kachko
PhD, Head of Department
Department of Software Engineering
Kharkiv National University of Radio Electronics
Nauky ave., 14, Kharkiv, Ukraine, 61166
Programming Department*
Yurii Gorbenko
PhD, First Deputy Chief Constructor*
Serhii Kandii
PhD Student**
Research Associate-Consultant*
Yevhenii Kaptol
Corresponding author
PhD Student**
Information Protection Systems Analyst*
*Institute of Information Technologies PrJSC
Profesora Otamanovskoho str., 15, Kharkiv, Ukraine, 61166
**Department of Security of Information Systems and Technologies
V. N. Karazin Kharkiv National University
Svobody sq., 6, Kharkiv, Ukraine, 61022

Received date 30.05.2024
Accepted date 08.08.2024
Published date 30.08.2024

How to Cite: *Kachko, O., Gorbenko, Y., Kandii, S., Kaptol, Y. (2024). Improving protection of falcon electronic signature software implementations against attacks based on floating point noise. Eastern-European Journal of Enterprise Technologies, 4 (9 (130)), 6–17. https://doi.org/10.15587/1729-4061.2024.310521*

1. Introduction

Lattice-based cryptography has become one of the most promising areas of research in modern cryptography. The main advantage of lattice-based cryptography is its resistance to attacks by quantum computers, which makes it particularly relevant in the context of the development of quantum technologies. Software implementations of such transformations often use new, little-researched techniques that can lead to vulnerabilities.

A typical example is the Falcon electronic signature scheme [1], which is a finalist in the third stage of the NIST PQC competition [2] and is based on problems in lattice theory. There are draft standards for all finalists. However, the Falcon is an exception. The problem of creating a standard for the Falcon algorithm is the complexity of implementing key generation and electronic signature (ES) algorithms.

One of the problems is related to the need to use floating point. The use of floating point complicates the verification of test vectors, makes it impossible to apply many protection techniques against side-channel attacks, and negatively affects performance on devices that do not support floating-point operations. In addition to implementation complexity issues, the use of floating-point calculations can lead to vulnerabilities.

In summary, research on methods for improving the security of software implementations of quantum-resistant electronic signatures is extremely important since these cryptographic methods will be standardized in the near future and will ensure the security of information and telecommunication systems. In particular, Falcon's electronic signature, as a finalist in the NIST PQC competition, needs a detailed study. This is especially relevant due to its peculiarity – the use of floating-point calculations, which remains underexplored.

2. Literature review and problem statement

The Falcon electronic signature was proposed in [1]. Its feature is the use of floating-point calculations. However, at the time of publication of the work, there was not enough

analysis of the required accuracy of the operations because at that time floating-point calculations were not common in cryptography. At that time, there was simply no need for such an analysis.

In work [3], Rényi divergence was used to justify the security of using 53 bits of accuracy. However, the analysis only considers attacks on the signature scheme itself, leaving out attacks on implementation features. The authors of the work were not interested in such an analysis since it has a purely practical orientation, while they developed theoretical models without reference to implementation. The authors of the Falcon electronic signature recognize that the use of floating-point calculations is problematic and creates numerous implementation difficulties. Some devices do not support floating point operations. Moreover, floating-point calculations lead to the fact that two correct implementations can give different signatures. This complicates the design of test vectors and makes possible attacks that cannot be applied to other electronic signatures.

There are no known implementation options for the Falcon algorithm without using a floating point. In [4], it is proposed to use the standard for floating point IEEE-754, which, as a rule, is implemented by modern compilers. The emulation option is an option for computing devices that do not support floating point but leaves all the disadvantages of floating point associated with limiting the precision of the task and calculation. This doesn't solve all floating point problems, but it does mitigate them.

In [5], an attack on the Falcon electronic signature through external channels was proposed, which uses the features of multiplying polynomials in the Fourier representation using a floating point. In the work, it is shown that it is possible to protect against this attack with the help of the correct implementation of calculations. But in work [6] a new attack was proposed, which made it possible to recover the secret key using power analysis, which again raised the question about the security of software implementations. Developing a software implementation that uses fixed-point computations could make this kind of attack more difficult, if not impossible.

Paper [7] shows a key recovery attack in the presence of two non-matching signatures, which uses floating-point calculations. But the probability of an attack was estimated roughly enough. The authors did not aim to find exact estimates but limited themselves to approximate estimates. Questions related to protection against the proposed attack also remained unresolved. It was stated in the work that this is the subject of further research.

Paper [8] proposes a method for generating keys without using floating point, which are replaced by fixed-point operations. There is an agreement with NIST about the possibility of using this method instead of the generation methods proposed in the reference implementation. However, the possibility of using this method at the stage of generating an electronic signature has not yet been investigated, since the topic is relatively new. Although the use of a fixed point does not completely solve the problem of floating point, it can make it more difficult to conduct attacks on third-party channels.

All this gives reason to assert that it is advisable to perform research that would make it more difficult to carry out attacks on the implementation of signatures that use floating-point calculations. Creating an implementation of Falcon that didn't use floating-point arithmetic at all would make many attacks more difficult. Therefore, possible areas of research are the use of a fixed point instead of a floating point and modification of the order of calculations in software implementations. Also, some attacks, such as [7], require clarification of the conditions of the attack. Clarifying the conditions under which attacks are carried out can provide a better understanding of an attacker's capabilities.

3. The aim and objectives of the study

The purpose of our study is to improve the security of software implementations of the Falcon electronic signature against attacks using the noise of floating-point calculations. The research will make it possible to highlight the necessary adjustments and methods for improving the security of both the ES Falcon scheme and other electronic signature schemes that use floating-point arithmetic.

To achieve the goal, the following tasks were set:

– to evaluate the influence of system-wide parameters on the probability of an attack;

– to determine what changes must be made in the software implementation so that the attack is impossible;

– to determine the possibility of using a fixed point instead of a floating one at the stage of generating an electronic signature to increase security.

4. The study materials and methods

The object of our research is the security of the ES Falcon scheme against attacks based on the use of features in the application of floating-point arithmetic.

The main hypothesis of the study assumes the existence of methods of protection against attacks using the influence of floating-point computing noise on the Falcon ES.

For the study, the ES Falcon scheme [2] adopted for standardization based on the results of the NIST PQC competition and its reference implementation was used.

The Falcon electronic signature scheme is based on the GPV framework, which was first proposed in [9] for the construction of quantum-resistant electronic signatures on lattices. The essence of the GPV framework is as follows:

– the public key is given by a matrix $A\in\mathbb{Z}_q^{n\times m}$ (where $m>n$). This matrix defines the basis of the q-ary lattice Λ;

– the secret key is specified by the matrix $B\in\mathbb{Z}_q^{m\times m}$. This matrix specifies the basis of the dual lattice $\Lambda_q^{\perp}$, which, according to the definition, is orthogonal to Λ modulo q. That is, for any vectors $x\in\Lambda$ and $y\in\Lambda_q^{\perp}$ $\langle x,y\rangle=0 \bmod q$ is fulfilled, where $\langle\langle\cdot,\cdot\rangle$ is a scalar product operation;

– for a given message m, the signature is a small (in the sense of the Euclidean norm) vector $s\in\mathbb{Z}_q^m$, for which $sA^T=H(m)$ is fulfilled, where $H:\{0,1\}^*\to\mathbb{Z}_q^n$ is a collision-resistant hash function. To check the signature, it is enough to check that the equation $sA^T=H(m)$ is fulfilled;

– to calculate the signature, an arbitrary random vector $c_o\in\mathbb{Z}_q^m$, is first calculated, for which $c_0A^T=H(m)$ is performed. Since no requirements are imposed on the vector c_0 regarding the values of its Euclidean norm, it can be found by standard means of linear algebra in polynomial time. Next, the secret basis B is used to calculate a vector $z\in\Lambda_q^{\perp}$, which is close to the vector c_0. The difference of vectors $s=c_0-z$ is a correct signature since $sA^T=c_0A^T-zA^T=c-0=H(m)$. If c_0 and v are close enough, then s will be small.

The Falcon electronic signature uses the Λ NTRU lattice [1, 10] as the lattice. The following notations have been introduced. Let $\phi=x^n+1$ (n is the power of two) and $q\in\mathbb{N}^*$. The polynomials $f,g,F,G\in\mathbb{Z}[x]/(\phi)$ define the NTRU lattice if the NTRU equation is fulfilled:

$$fG-gF=q\left(\mathrm{mod}\,\phi\right). \tag{1}$$

Explicitly, the basis of the NTRU lattice is given as follows:

$$\begin{pmatrix} f & g \\ F & G \end{pmatrix}. \tag{2}$$

The polynomial $h=g\cdot f^{-1}\mathrm{mod}q$ also defines the basis of the same lattice and explicitly the NTRU basis of the lattice is given as follows:

$$\begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}. \tag{3}$$

If the polynomials f, g have small coefficients, then the task of restoring f, g from the polynomial h is considered difficult (NTRU problem). By applying the lattice NTRU to the GPV framework, Falcon makes the following changes to GPV:

– the public key is the polynomial h, which is used to calculate the NTRU basis of the lattice Λ;

– the secret key is the polynomials $f,g,F,G\in\mathbb{Z}[x]/(\phi)$, which are used to calculate the basis of the dual lattice $\Lambda_q^{\perp}$;

– the signature for the message m consists of a pair of polynomials (s_1, s_2) for which $s_1+s_2h=H(r\|m)$, where r is salt, is fulfilled. When calculating the signature, a secret key is used to calculate a vector $z\in\Lambda_q^{\perp}$, which is close to the vector $t=(H(m\|r),0)$. The difference of vectors t and z is the correct signature.

To calculate the vector $z=(z_0,z_1)$, the sampling algorithm [11] is used, which returns a vector from a normal distribution. A feature of the Falcon sampling algorithm [12] is the use of the algebraic structure of the cyclotomic field and the Fourier transform to speed up operations. Falcon also uses tree-like data structures – LDL trees. Details can be found in the specification from [1].

In work [7], an attack on the implementation of ES Falcon was proposed. The attack is based on the use of two different implementations, which, due to rounding noise, give different signatures for the same messages. The key equation of the attack is:

$$h=\frac{g\delta_0+G\delta_1}{f\delta_0+F\delta_1}, \tag{4}$$

where $\delta_0=z_0^1-z_0^0$ and $\delta_1=z_1^1-z_1^0$ (the superscript indicates the serial number of the implementation).

The idea of the attack is to choose the message m in such a way that $\delta_1=0$, and δ_0 is some small enough to use the algorithm for finding the greatest common divisor of a polynomial (ideally – $\delta_0\in\mathbb{Z}$). Then, it will be possible to calculate the secret key $\tilde{f}$, $\tilde{g}$ as:

$$\tilde{f}=\left(s_1^1-s_1^0\right)/\gcd\left(s_1^1-s_1^0,s_0^0-s_0^1\right),$$

$$\tilde{g}=\left(s_0^0-s_0^1\right)/\gcd\left(s_1^1-s_1^0,s_0^0-s_0^1\right), \tag{5}$$

where gcd is the greatest common divisor of the polynomials.

Since the attack is based on the use of floating-point noise, one defense is to account for rounding errors, and another is to use fixed-point operations instead of floating-point.

This study was performed on the implementation example [1], folder Extra. In the following, this implementation is designated as the reference implementation. In the considered version of implementation, two options for calculating ES are given:

Option 1. Calculation based on the secret key – polynomials f, g, F, G. In this case, the LDL tree and components of the ES are calculated in parallel with the ES calculation (sign_dyn function).

Option 2. All computations for the LDL tree depend only on the secret key and are independent of the signature message. They are performed in advance (expand_privkey function). This significantly speeds up the formation of ES if it is necessary to sign several documents using one secret key, but significantly increases the size of the required memory (LDL tree must be stored. If the secret key carrier allows for protected memory, these calculations can be performed once after the generation of keys. In this case, the effect of applying the second method will hold even in the case of forming only one signature (function sign_tree).

The attack from [7] makes it possible to recover or calculate new polynomials f, g, F, G, sign documents and successfully verify the signature with a legal public key. The attack provides these opportunities in the case of obtaining different signatures under the conditions of using the same signature messages and random seed2 and nonce data. Therefore, it is advisable to analyze in more detail the conditions for calculating different signatures and the means of preventing this.

The following statistical methods are used within this study:

– the Chi-Square test [13]: this is a statistical test used to test hypotheses about the independence or correspondence of the observed frequency distribution to the theoretical frequencies. It is used for categorical data and is based on a comparison of expected and actual frequencies;

– the Kolmogorov-Smirnov test [14]: this is a non-parametric test that is used to compare two samples or to check whether the sample matches the theoretical distribution. The test is based on the largest distance between the cumulative functions of the sample distribution and the theoretical distribution;

– quantile-quantile graph (Q–Q graph) [15]: this is a graphical tool for comparing two distributions by plotting their quantiles against each other. If the points on the graph form an approximately straight line, this means that the distributions are similar;

– Kruskal-Wallis test: this is a non-parametric method for comparing the medians of several independent groups. It is a generalization of the Mann-Whitney test and is used when the normality assumption for the data is not met;

– Student's t-test: this is a parametric test used to compare the means of two samples or to test whether the mean of one sample corresponds to a given value. It is used when the data have a normal distribution and known or unknown variances;

– polynomial regression: this is a method of regression analysis in which the relationship between the independent variable and the dependent variable is modeled as a polynomial. Polynomial regression allows taking into account nonlinear dependences between variables.

5. Results of investigating methods for improving the security of software implementations of the Falcon electronic signature

5. 1. Assessment of the impact of system-wide parameters on the probability of an attack

The following statistics were collected to obtain the required estimates: for 10 random keys for $\log_2 n$=2, 3, 4, 5, 6, 7, 8, 9, 10, signatures were computed on random 33-byte messages until 100 events were obtained:

– differences in signatures with identical messages and seed;

– a successful key recovery attack.

Thus, two statistics were acquired. It was hypothesized that for each key, for each $\log_2 n$ value, the statistic obeys the exponential distribution. This hypothesis was tested at the 0.01 level by the Chi-square and Kolmogorov-Smirnov tests. For all keys, the null hypothesis of an exponential distribution was accepted. Fig. 1 shows an example of a typical distribution and a Q-Q plot.

Although events are exponentially distributed for all keys, the parameters of the distribution may differ for each key. To test the hypothesis of equivalence of keys, the Kruskal-Wallis test (H-test) was applied to all keys for all values of $\log_2 n$. The corresponding data are entered in Table 1. If the p-value is greater than the significance level, then it is considered that there is no statistically significant difference between the distributions.

For the completeness of the picture, the Kruskal-Wallis test was also applied in pairs to each of the distributions. The results of the comparison are shown in Fig. 2, 3, where the square with coordinates (i, j) visualizes the p-value obtained by comparing the distributions of the i-th and j-th keys. Green color corresponds to one, red color to zero.

Table 1

Estimation of the equality of distribution medians by the Kruskal-Wallis test

N	There is a difference in signatures		Successful attack	
	Statistics	p-value	Statistics	p-value
2	62.911	3.67e-10	131.961	4.68e-24
3	76.360	8.507e-13	45.352	7.93e-07
4	31.218	0.00027	53.805	2.05e-08
5	22.257	0.00809	26.346	0.00179
6	30.343	0.00038	28.641	0.00074
7	48.355	2.19e-07	33.766	9.81e-05
8	18.5688	0.029118	170.475	4.91e-32
9	27.565	0.001126	21.602	0.010226
10	17.2289	0.045249	21.298	0.01139

From Table 1 and Fig. 2, 3, it is possible to conclude that for $\log_2 n$=10 the distributions satisfy the Kruskal-Wallis criterion for a significance level of 0.01. However, it should be noted that in general the distributions for different keys are different and the expected number of signatures depends on the key.

To calculate the expected number of signatures for a randomly selected key, it is possible to calculate the expected value for all keys and average it. The mean values of the distributions are expected to be normally distributed, so it is possible to apply the Student's t-test to the obtained values to estimate the most likely value of this parameter.

Confidence intervals for each of the keys were calculated using the Student's t-test. The expected average values are listed in Table 2.

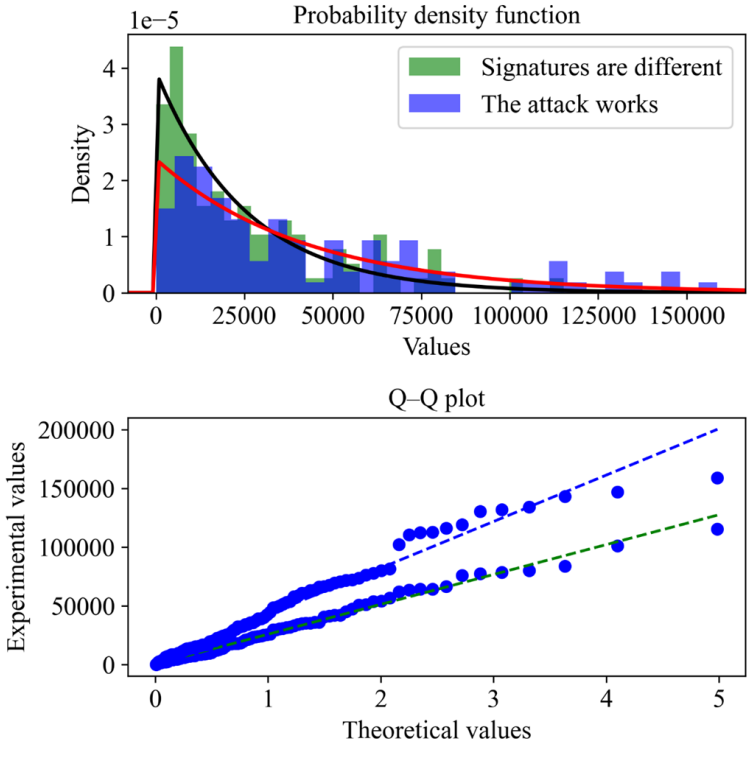

Fig. 1. Density function and Q–Q plot for $\log_2 n$=9

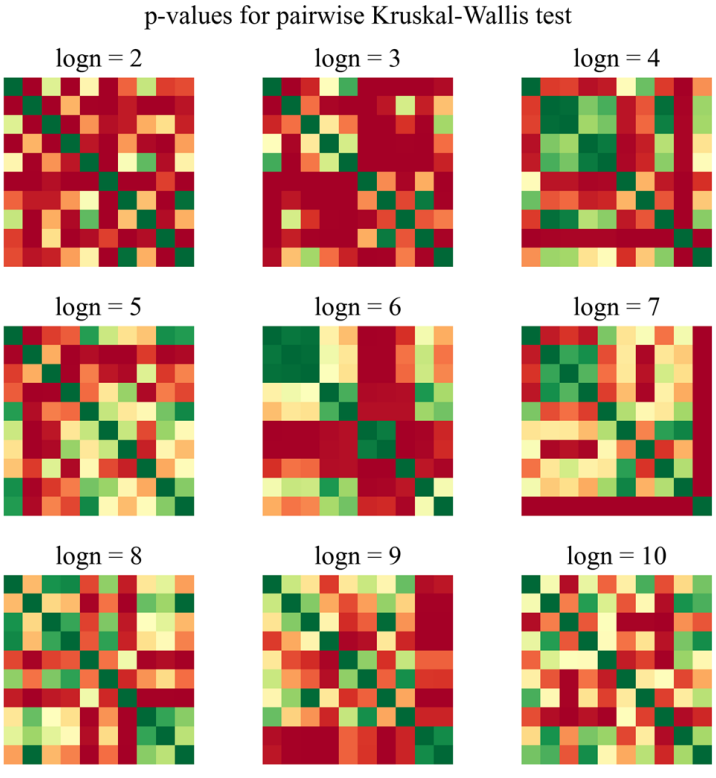

Fig. 2. The p-value for the Kruskal-Wallis pairwise test for the difference-in-signs statistic

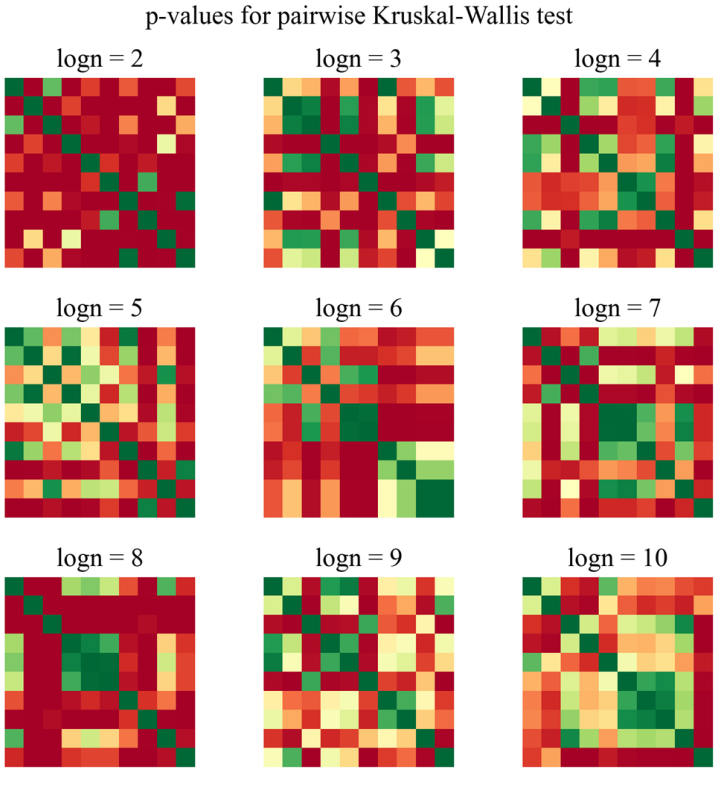

Fig. 3. The p-value for the Kruskal-Wallis pairwise test for the successful attack statistic

Fig. 4, 5 show 95 % confidence intervals for the expected number of signatures of the corresponding statistics.

It is quite interesting that the dependence of the number of signatures on $\log_2 n$ is not monotonic. However, this can be explained by an insufficiently large amount of statistical material.

Linear, quadratic, and cubic regression models were applied to the obtained estimates, and the coefficient of determination R^2 was calculated to assess the quality of the models. Fig. 6, 7 show the corresponding models; Table 3 gives the obtained estimate of the average number of signatures.

Table 2

Expected average number of signatures

Key	$\log_2 n$=2	$\log_2 n$=3	$\log_2 n$=4	$\log_2 n$=5	$\log_2 n$=6	$\log_2 n$=7	$\log_2 n$=8	$\log_2 n$=9	$\log_2 n$=10
Statistics of differences in signatures									
1	40386	52424	25984	28728	27704	34612	34276	33828	33222
2	23202	31346	33877	46674	29318	26374	39661	28854	29877
3	47154	44637	37725	35368	29782	28839	34582	28055	22245
4	26125	47474	33517	25336	30066	26100	33067	25390	30593
5	39963	49803	34086	32891	31642	36444	24367	37016	28000
6	65627	22292	26366	29067	45190	34893	32164	35695	40353
7	33151	24164	27038	27029	45219	38335	25691	29825	30165
8	36703	32555	29579	34887	26207	30686	33477	36803	42870
9	48764	28125	46406	28709	29673	39975	38210	44872	31468
10	33968	42215	32540	32820	35353	65048	37617	44665	34608
Statistics of a successful attack									
1	76945	109577	49630	48050	59888	55900	50185	47478	50286
2	183296	89302	55542	48871	52282	81730	175649	41240	54094
3	78895	86359	89954	51729	69551	43473	102241	57725	39488
4	135106	142008	56461	41662	58987	80722	40079	42062	35699
5	58165	84637	58854	54557	71641	51648	44177	43061	45783
6	45930	62435	74446	54927	68080	53771	42342	67335	46750
7	92645	113691	70199	43453	47131	48301	57978	54718	40465
8	44826	116400	52626	76233	44067	59914	75243	50474	47772
9	164514	100128	108978	55696	51824	53938	45177	55511	45871
10	86542	80268	54442	78320	51824	38122	34715	36177	62573

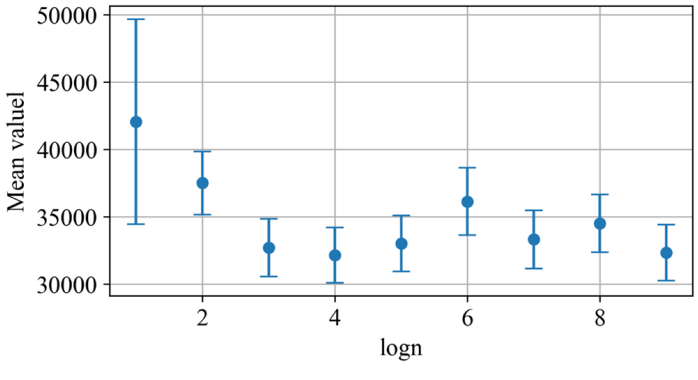

Fig. 4. The 95 % confidence intervals for the expected number of signatures for the signature difference statistic

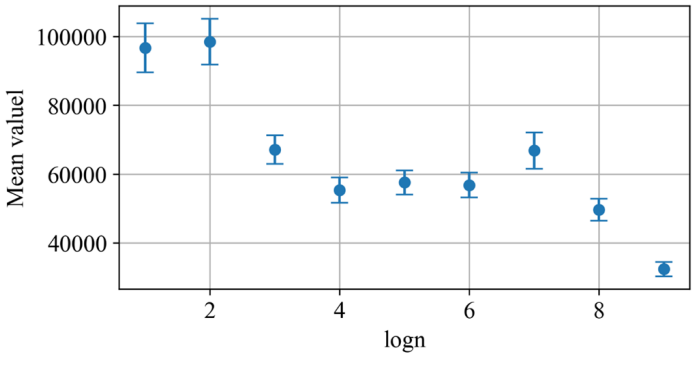

Fig. 5. The 95 % confidence intervals for the expected number of signatures for the success attack statistics

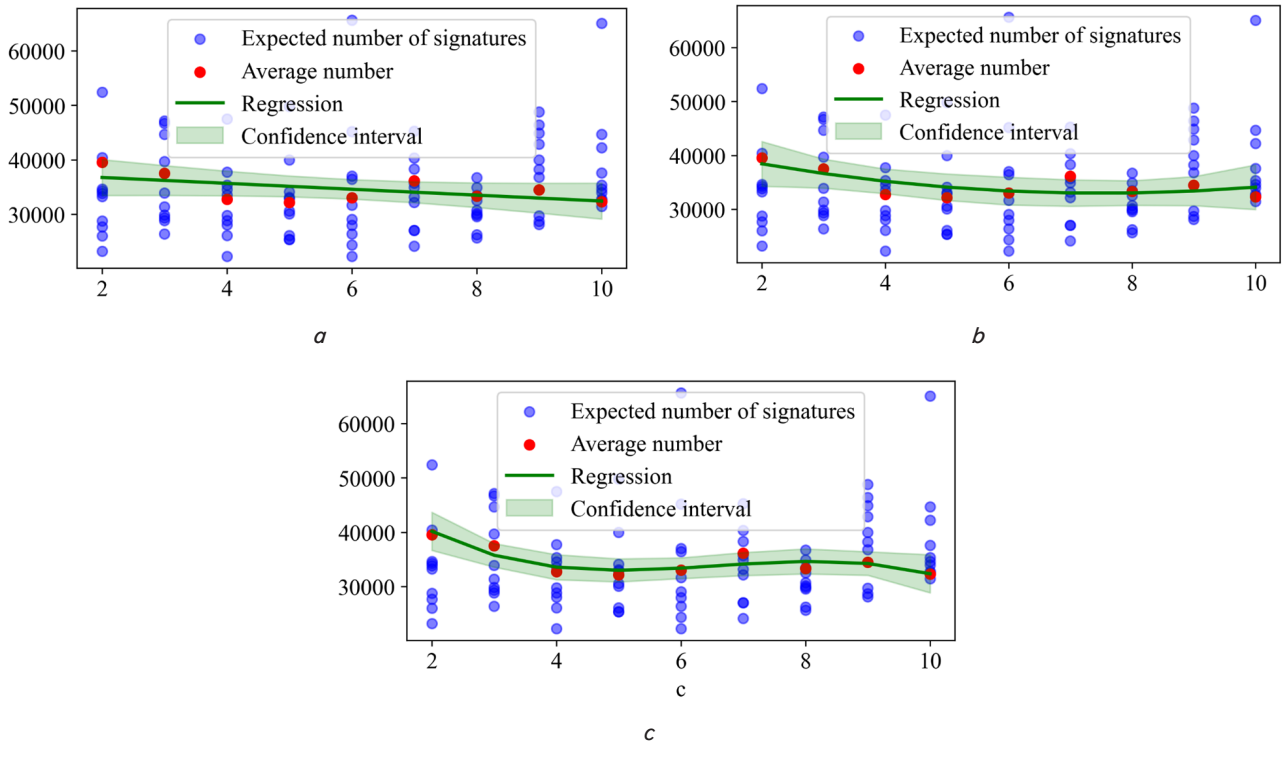

Fig. 6. Application of regression models to statistics of differences in signatures: *a* – linear model; *b* – quadratic model; *c* – cubic model

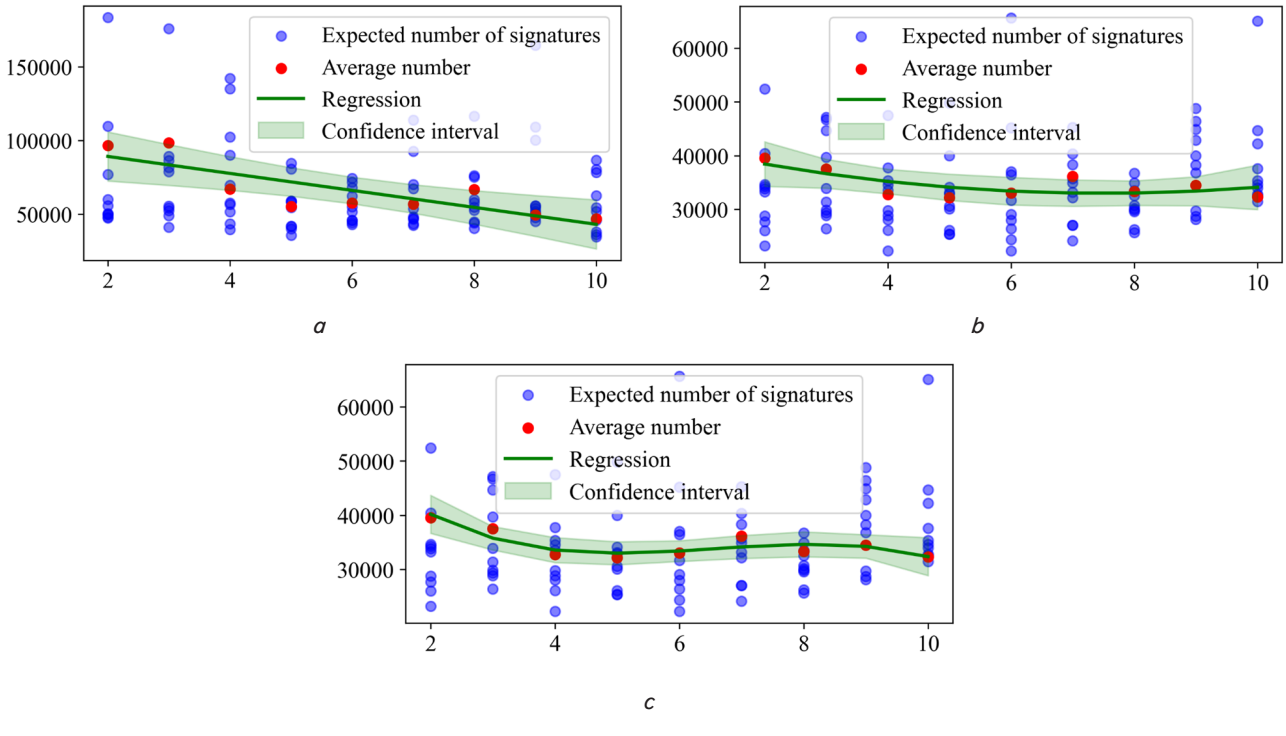

Fig. 7. Application of regression models to the statistics of a successful attack: *a* – linear model; *b* – quadratic model; *c* – cubic model

From Fig. 6, 7 and Table 3, it is possible to draw the following conclusions:

– statistics of differences in signatures are well described only by the cubic model. The linear model has a coefficient of determination of 0.3277, which shows a poor fit to the data, the quadratic model has a coefficient of determination of 0.5138, which, although considered satisfactory, is at the very limit. The cubic model describes the data well;

– it is worth noting that for cryptographically significant parameters (N=9, N=10) the estimates of all models coincide, despite different coefficients of determination.

Table 3

Expected number of signatures according to regression models

Statistics	Statistics of differences in signatures			Statistics of a successful attack		
Model	Linear	Quadratic	Cubic	Linear	Quadratic	Cubic
R^2	0.3277	0.5138	0.7980	0.6838	0.7909	0.8368
N=2	36740	38421	40158	89141	98523	103660
N=3	36199	36619	35750	83388	85733	83164
N=4	35657	35177	33563	77634	74953	70183
N=5	35116	34095	32978	71881	66184	62882
N=6	34574	33374	33374	66127	59426	59426
N=7	34033	33012	34129	60374	54677	57980
N=8	33491	33011	34625	54620	51940	56710
N=9	32950	33370	34239	48867	51212	53781
N=10	32408	34089	32351	43113	52495	47358

For the statistics of a successful attack, all models show high coefficients of determination, however, considering that the peak for N=8 occurs for many keys independently, it is advisable to use the cubic model.

5. 2. Investigating the reference software implementation of ES Falcon

Different signatures can be obtained for two variants of ES calculation without recomputing the LDL tree and recomputing if the reference implementation is used. The reason for the appearance of different signatures is the use of floating point when calculating ES. When performing ES calculations for both options, mathematically equivalent operations, but different in calculation accuracy, were used.

Analysis of the code revealed that there are discrepancies in the order of execution of operations and replacement of operations when implementing the functions ffSampling_fft_dyntree and ffSampling_fft at log$n$=2. Namely, this happens when the split (poly_split_fft function) and merge (poly_merge_fft function) operations are implemented in the ffSampling_fft_dyntree function and their sweep in the ffSampling_fft function.

When implementing the split operation in the function ffSampling_fft_dyntree for log$n$=2, the operation of multiplying complex numbers is performed. This implementation requires 4 multiplication operations and two addition operations, followed by a halving of the multiplication result. This means that when multiplying the complex numbers $x+iy$ and $z+iu$ we shall get the number $v+iw$, where $v=x\cdot z-y\cdot u$, $w=x\cdot u+y\cdot z$, and after dividing in half $v=v/2$, $w=w/2$. For log$n$=2 values are $z=\frac{1}{\sqrt{2}}$, $u=-\frac{1}{\sqrt{2}}$.

In the ffSampling_fft function, it is taken into account that the second complex number $z+iu$ for log$n$=2 is $1/\sqrt{2}-i/\sqrt{2}$. Then as a result of multiplication we get $v=x/\sqrt{2}+y/\sqrt{2}$ and $w=y/\sqrt{2}-x/\sqrt{2}$, and taking into account division in half we get: $v=(x+y)/\sqrt{8}$, $w=(y-x)/\sqrt{8}$. This means that instead of 4 multiplication operations, two operations of multiplication by a constant are used. In terms of the number of operations, the second option is better because the 4 multiplication operations in the first option are replaced by two multiplication operations.

Below, these options are considered from the point of view of the accuracy of calculations on the example of calculating the value of v for both options. It can be considered that the values of x, y, and $u=-z$ are the same for both options. The prediction is based on the fact that the same algorithm for calculating x, y, u, z values is used for previous iterations for both options.

According to the first option, $v=x^*z+y^*z$. When calculating the products x^*z, y^*z, rounding is performed, that is, an addition operation is performed for rounded values, in this case, the loss of a large number of correct digits of the result is possible ([16], Catastrophic cancellation). This option is used by the reference function poly_split_fft.

According to the second option, $v=x^*z+y^*z=z\,(x+y)$. According to our prediction, the values of x, y for both options are the same, so when calculating $x+y$, the error is minimal and, after multiplication, we shall get an error that can be neglected ([16], benign cancellation), that is, a reduction in the calculation error. This option is used by the proposed function poly_split1_fft, in which the branch at log$n$=2 is processed separately.

Similarly, for the merge operation (which is implemented in the reference function poly_merge_fft), instead of multiplying two complex numbers, it is taken into account that the second complex number is $1/\sqrt{2}+i/\sqrt{2}$. The result of multiplication in this case $v=(x-y)/\sqrt{2}$, $w=(x+y)/\sqrt{2}$, (function poly_merge1_fft), instead of 4 multiplication operations, two multiplication operations by a constant are used with the same x, y values for both options.

The following are examples of obtaining different accuracy of calculations for log$n$=2:

Functions poly_split_fft and poly_split1_fft.
Input polynomial A:
2.1043620407812096, 3.8282371709651688, 2.0947742733457018, 0.34234852196635635
Etalon function poly_split_fft
Output polynomials:
B: 2.9662996058731892, 1.2185613976560290
C: 0.010094168927173075, 1.2290579633990930
Function poly_split1_fft
Output polynomials:
B: 2.9662996058731892, 1.2185613976560290
C: 0.010094168927173020, 1.2290579633990930
In polynomial C for the first coefficient last 2 digits are different!
Functions poly_merge_fft and poly_merge1_fft
Input polynomials:
A: -0.57360274860716420, 0.87702594501647990
B: 1.4920231256032801, 0.53999771191485768
Function poly_merge_fft
Output polynomial:
C: 0.099580877273847457, -1.2467863744881758, 2.3138816587379227, -0.55982976870496293

Function poly_merge1_fft
Output polynomial:
C:0.099580877273847568,-1.2467863744881760, 2.3138816587379227, -0.55982976870496293.

Subsequently, this difference leads to obtaining different signatures.

We have measured the performance of the reference and proposed functions for log$n$=2. Results are as follows:

– reference functions poly_split_fft and poly_merge_fft (log$n$=2): 20 and 22 cycles, respectively;

– functions poly_split1_fft and poly_merge1_fft (log$n$=2): 15 and 15 cycles, respectively. For all other values of log$n$, the functions are the same, so using the poly_split1_fft and poly_merge1_fft functions instead of the reference functions does not reduce performance.

After using the poly_split1_fft, poly_merge1_fft functions instead of the reference poly_split_fft, poly_merge_fft functions, the effect of obtaining different ESs disappeared, which means that the attack on the recovery of secret keys is impossible [7].

5. 3. Determining the possibility of using a fixed point instead of a floating point at the stage of generating an electronic signature

The use of floating point, as the previous point also shows, leads to problems related to the fact that the precision changes dynamically due to the changed amount of the non-integer part of the number. Also, floating precision is not available on all computing systems. That is why in [8] it is proposed to use a fixed point instead of a floating point, it is described how to do it, and the possibility of using this method is shown to calculate keys for the Falcon algorithm; but only keys, the calculation of a digital signature for the Falcon algorithm was not considered.

In this work, this method is applied to calculate a digital signature, and it is verified that in the case of using a fixed point with the scale proposed by the authors of [8], it is impossible to calculate ES. Similarly to work [8], the fixed point is implemented according to the IEEE Std 754TM-2008 standard [17].

The idea of the method is that instead of the usual floating point, a scaled number is set. Work [8] serves as a starting point for research on the use of a fixed point. In [8], it was proposed to use the value 2^{32} as a scale (this scale will be indicated by the number of bits for the non-integer part SCALE=32). To convert to a scaled number, it is multiplied by 2^{SCALE}. In the same work, a set of functions was proposed for the simplest operations on real and complex data for such numbers.

Thus, 32 bits are allocated for assigning a non-integer part of a number, which determines the accuracy of assigning a number. 31 bits are allocated for the task of the whole part, one bit is allocated for the sign of the number. For the successful application of such a task, a necessary condition is the use of numerical values that do not exceed 2^{31} in modulo, including all intermediate results that must satisfy this condition. To generate ES, it is necessary to calculate the LDL tree [1, 18]. Unfortunately, when calculating the LDL tree, the intermediate results when working with complex data exceed the permissible values. Because of this, the application of this task method for the generation of ES is not possible. Therefore, it is necessary to increase the length of the whole part of the number, which is possible through:

– transformation of algorithms to reduce the maximum number;

– decrease in number accuracy;

– increasing the total length of the number.

An increase in the total length of the number (more than 64 bits) was not considered in the work, as it would lead to a significant decrease in performance; further, the solution to the problem was considered by transforming the algorithms and reducing the length of the non-integer part of the number.

Let's transform the algorithm for dividing complex numbers, which is used in the reference implementation. Instead of the formula:

$$\frac{a+bi}{c+di}=\frac{(a+bi)(c-di)}{c^2+d^2}. \tag{6}$$

We use the following formula:

$$\frac{a+bi}{c+di}=(a+bi)\left(\frac{c}{c^2+d^2}-\frac{d}{c^2+d^2}i\right). \tag{7}$$

To calculate the new scale, we experimentally determined the maximum value by modulus for all floating-point calculations when creating ESs, formed using the reference implementation, taking into account the transformations defined above. For the first 1000 keys of the reference implementation, the maximum value is less than 2^{37}, that is, the length of the integer part is 37, the length of the non-integer part is 64–38=26 bits, SCALE=26.

Due to the fact that different scales are used for key generation and ES generation, a library of functions for the simplest operations on real and complex data for such numbers has been designed. It is suitable for use for arbitrary scaling under conditions where the total length of the number remains 64 bits.

The most computationally intensive basic operations of the library are the multiplication and division functions for fixed-point numbers.

Multiplication.

$c=a\cdot b$, where a, b are fixed-point data, 64 bits long, specified with SCALE scale; c is the result of the multiplication, fixed-point data, 64 bits long, specified with SCALE.

It is guaranteed that the input data and the output do not exceed the permissible limits.

To multiply 64-bit x, y, 3 multiplication operations are used (Karatsuba's algorithm). The result is a 128-bit number.

The same algorithm is used to multiply complex numbers.

The number is shifted towards the least significant bits by SCALE bits.

64 bits of the result remain.

Since SCALE<=32, the senior 32 bits of the 128-bit result are not used, so you can ignore the carryover to the last 32 bits.

When using floating point data, one floating point operation is used.

The multiplication operation does not increase the execution time compared to using floating point.

Division.

$c=a/b$, where a, b ($b\neq 0$) are fixed-point data, 64 bits long, specified with the SCALE scale; c is the result of the division, fixed-point data, 64 bits long, specified with SCALE.

It is guaranteed that the input data and the output do not exceed the permissible limits.

The whole part is if $a \geq b$.
For the whole part:

```
while (a>=b) {
        uint64_t c1=1;
        b1=b;
        while (a>=b1)
        {
                b1=b1*2;
                c1=c1*2;
        }
        b1=b1/2;
        c1=c/2;
        c=c+c1;
        a=a–b1;
}
c=c<<SCALE.
```

For the fractional part:

```
uint64_t ost=a, drob_part=0;
for (int i=0; i<SCALE; ++i){
ost2=ost*2;
        d=~((int64_t)(ost2–b)>>63);
        drob_part=(drob_part<<1)–d;
        ost=ost2 (d & b);
}
c=c+drob_part.
```

To determine the effect of using a fixed point on the performance of the ES calculation operation, the average value of the complexity of the operations (in CPU cycles) was calculated for 1000 operations with random arguments. Standard multiplication and division operations for 64-bit floating-point data were used for comparison. The results for floating-point data and fixed-point data are given in Table 4:

Table 4

Calculation results for floating-point data and for fixed-point data

Operation	Floating-point data	Fixed-point data
Multiplication	35.604	34.879
Division	24.68	24.309

For the calculations given in Table 4 we used an 11th Gen Intel(R) Core (TM) i7-1165G7 @ 2.80 GHz processor.

The determined accuracy has made it possible to generate an electronic signature for 1000 random keys of the reference implementation.

6. Discussion of results based on the study of methods for countering the attack on the Falcon ES

The probability of an attack is inversely proportional to the expected number of signatures. Table 3 demonstrates that the expected number of signatures for a successful attack is 51212-53781 for Falcon512 and 47358-52495 for Falcon1024, which is one and a half times more than the estimate of 30000 in paper [7]. This becomes possible thanks to a larger statistical sample and taking into account the peculiarities of probability distributions. Preliminary estimates were obtained by simply averaging the required number of signatures for different keys without considering probability distributions.

As the log$n$ parameter increases, the probability of an attack increases. This is because the number of floating-point calculations is increasing. From Fig. 4, 5 it follows that the obtained dependence of the required number of signatures on log$n$ is not monotonically increasing. However, since the difference between the quadratic and cubic regression estimates falls within the 95 % confidence interval for the mean values, we can neglect this effect and assume that the probability increases monotonically.

The obtained estimate of the number of signatures can be used as a limit on the number of signatures on one pair of keys in a public key infrastructure to protect against an attack. This will increase the security of the Falcon electronic signature in a qualitative sense – the attack from [7] becomes impossible with restrictions on the number of generated signatures.

An attack on Falcon is possible because there are discrepancies in the order of operations when implementing the functions ffSampling_fft_dyntree and ffSampling_fft at log$n$=2. In the ffSampling_fft_dyntree function, split (poly_split_fft function) and merge (poly_merge_fft function) operations are applied to all log$n$ values. In the function ffSampling_fft for log$n$=2, the corresponding operations are used instead of functions. If these operations are taken into account when implementing the poly_split_fft and poly_merge_fft functions for log$n$=2, then the attack becomes impossible. In contrast to work [7], the place of occurrence of discrepancies in the signatures is localized more precisely. Applying the proposed changes to the reference implementation makes it possible to prevent the attack from being implemented. This becomes possible due to the agreement of the order of calculations at log$n$=2.

Moreover, the proposed changes to the implementation make it possible to speed up the implementation. The reference functions run at 20–22 machine clocks on x86_64 processors, while the modified functions run at 15 clocks.

Since no differences between signatures are observed when using the modified functions, the security of the Falcon electronic signature increases in a qualitative sense – the attack from [7] becomes impossible in the modified implementation.

To increase the security of the implementation, a minimum scale for fixed-point calculations of 2^{26} was also found. This estimate was obtained experimentally. The theoretical justification of the assessment is the subject of further research. Previous works [8, 19] considered only the application of fixed point to key generation. The scale found in this study allows the use of fixed-point computations for signature operations and not just for key generation, in contrast to work [8]. The application of fixed-point arithmetic to signature verification operations was not considered because the algorithm does not apply floating-point operations.

The scale found allows using fixed-point computations not only for key generation but also for signature generation. An implementation with fixed-point instead of floating-point will not completely eliminate the problem. However, it will reduce their manifestations. That is, in a qualitative sense, it increases the security of the Falcon electronic signature.

As can be seen from Table 4, the average values of the execution time of basic operations practically do not differ. That is, using a fixed point instead of a floating point should not affect the performance of the ES calculation operation.

The obtained estimates of the number of signatures for an attack allow us to set limits for the reference implementation under which the probability of an attack will be arbitrarily small. The proposed changes in the implementation code eliminate the possibility of an attack since differences between signatures do not occur. At the same time, the obtained estimates of the allowable scale factor for fixed-point calculations allow protection against other attacks on floating-point calculations.

A limitation of our results is that they are obtained for one reference implementation of the Falcon ES. For other implementations, research must be conducted separately.

The disadvantage of the results is the limitation of the test sample to the set of keys of the reference implementation. A sample size that is too small can theoretically lead to the possibility of an attack with a number of signatures that differs from the obtained estimates.

For the further development of this direction, it is important to investigate the vectors of attacks on the Falcon ES scheme updated after the implementation of countermeasures.

7. Conclusions

1. The expected number of signatures for a successful attack has been specified, which is one and a half times more than the previous estimates in work [7] and is 51212-53781 for Falcon512 and 47358-52495 for Falcon1024. It was noticed that the increase in the probability of an attack, which is associated with an increase in the log$n$ parameter, is not monotonic. However, due to the entry of the difference between the quadratic and cubic regression estimates into the confidence interval for the mean values (95 %), this effect can be neglected and the increase in probability can be considered monotonic. The obtained estimate of the number of signatures can be used as a limit on the number of signatures on one pair of keys in the public key infrastructure.

2. An attack on the features of the floating point implementation becomes possible due to differences in the order of operations when implementing the functions ffSampling_fft_dyntree and ffSampling_fft at log$n$=2. The split (poly_split_fft function) and merge (poly_merge_fft function) operations in the ffSampling_fft_dyntree function and their sweeps in the ffSampling_fft function are problematic from this point of view. To prevent the attack, it is necessary to eliminate this discrepancy. For this purpose, a modification to the implementation was proposed, which also makes it possible to accelerate the implementation. So, the modified functions work in 15 cycles compared to 20–22 machine cycles of the reference implementation on processors with the x86_64 architecture. It was observed that the functions modified in the proposed way eliminate the occurrence of differences between signatures. Thus, the attack from [7] becomes impossible in the modified implementation, which increases the security of the Falcon ES in a qualitative sense.

3. A scale of 2^{26} was derived experimentally, which allows the application of fixed-point calculations for the formation of ES and other transformations in the reference implementation of the Falcon ES. Using fixed-point operations does not change the time of using floating-point operations. That is, replacing the floating point with a fixed one will not greatly affect the performance. At the same time, the use of fixed-point operations allows one to reduce the problems associated with the use of floating-point, which qualitatively increases the security of the software implementation.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

The data will be provided upon reasonable request.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

Acknowledgments

The authors express their gratitude for the support provided in the preparation of this paper, as well as for useful comments and suggestions, to the research team of JSC “IIT”.

References

1. Fouque, P., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T. et al. (2020). Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. Available at: https://falcon-sign.info/falcon.pdf
2. Post-Quantum Cryptography PQC. NIST. Available at: https://csrc.nist.gov/projects/post-quantum-cryptography
3. Prest, T. (2017). Sharper Bounds in Lattice-Based Cryptography Using the R nyi Divergence. Advances in Cryptology – ASIACRYPT 2017, 347–374. https://doi.org/10.1007/978-3-319-70694-8_13
4. Pornin, T. (2019). New Efficient, Constant-Time Implementations of Falcon. ePrint IACR. Available at: https://eprint.iacr.org/2019/893
5. Karabulut, E., Aysu, A. (2021). FALCON Down: Breaking FALCON Post-Quantum Signature Scheme through Side-Channel Attacks. 2021 58th ACM/IEEE Design Automation Conference (DAC). https://doi.org/10.1109/dac18074.2021.9586131
6. Guerreau, M., Martinelli, A., Ricosset, T., Rossi, M. (2022). The Hidden Parallelepiped Is Back Again: Power Analysis Attacks on Falcon. IACR Transactions on Cryptographic Hardware and Embedded Systems, 141–164. https://doi.org/10.46586/tches.v2022.i3.141-164

7. Potii, O., Kachko, O., Kandii, S., Kaptol, Y. (2024). Determining the effect of a floating point on the Falcon digital signature algorithm security. Eastern-European Journal of Enterprise Technologies, 1 (9 (127)), 52–59. https://doi.org/10.15587/1729-4061.2024.295160
8. Pornin, T. (2023). Improved Key Pair Generation for Falcon, BAT and Hawk. Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2023/290
9. Gentry, C., Peikert, C., Vaikuntanathan, V. (2007). Trapdoors for Hard Lattices and New Cryptographic Constructions. Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2007/432
10. Albrecht, M., Ducas, L. (2021). Lattice Attacks on NTRU and LWE: A History of Refinements. Cryptology ePrint Archive. Available at: https://eprint.iacr.org/2021/799
11. Prest, T. (2015). Gaussian Sampling in Lattice-Based Cryptography. THALES. Available at: https://tprest.github.io/pdf/pub/thesis-thomas-prest.pdf
12. Ducas, L., Prest, T. (2016). Fast Fourier Orthogonalization. Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation. https://doi.org/10.1145/2930889.2930923
13. Fisher, R. A. (1922). On the Interpretation of 2 from Contingency Tables, and the Calculation of P. Journal of the Royal Statistical Society, 85 (1), 87. https://doi.org/10.2307/2340521
14. Simard, R., L'Ecuyer, P. (2011). Computing the Two-Sided Kolmogorov-Smirnov Distribution. Journal of Statistical Software, 39 (11). https://doi.org/10.18637/jss.v039.i11
15. Wilk, M. B., Gnanadesikan, R. (1968). Probability plotting methods for the analysis for the analysis of data. Biometrika, 55 (1), 1–17. https://doi.org/10.1093/biomet/55.1.1
16. What Every Computer Scientist Should Know About Floating-Point Arithmetic. Available at: https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html
17. IEEE Std 754TM-2008. IEEE Standard for Floating-Point Arithmetic. IEEE Computer Society. Available at: https://iremi.univ-reunion.fr/IMG/pdf/ieee-754-2008.pdf
18. Pornin, T., Prest, T. (2019). More Efficient Algorithms for the NTRU Key Generation Using the Field Norm. Public-Key Cryptography – PKC 2019, 504–533. https://doi.org/10.1007/978-3-030-17259-6_17
19. [FALCON OFFICIAL] Keygen implementation. Available at: https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/bjVkrZmI9VM