Improvement of Implementation of Merkle Crypto System

Maksim Iavich [1] [0000-0002-3109-7971], Arturo Arakeliani [2] [0000-0002-3109-7971],
Giorgi Iashvili[2][0000-0002-1855-2669], Dali Magrakvelidze[3] [0000-0002-3109-7971],
Tetiana Okhrimenko[4] [0000-0001-9036-6556]

[1] Caucasus University, Tbilisi, Georgia
[2] University of Georgia, Tbilisi, Georgia
[3] Georgian Technical University, Tbilisi, Georgia
[4] National Aviation University, Kyiv, Ukraine
taniazhm@gmail.com

Abstract. Today, the information security of each country is a key issue in ensuring national security, taking into account that modern information and communication technologies (ICT) are implemented in all spheres of life. Increasing number and power of cyberattacks on ICT forcing scientists around the world to seek new methods to secure information. Traditional cryptographic methods, which are mostly used to ensure data confidentiality, do not provide protection against all currently known attacks, in particular, they are vulnerable to attacks based on quantum algorithms. Ways to solve this problem are the protocols of quantum and post-quantum cryptography. Thereby this article describes hash-based digital signature systems. These systems are safe against quantum computer attacks. Quantum computers can break existing public-key crypto systems. Quantum computer solves the discrete logarithm problem both for finite fields and elliptic curves. As it is able to efficiently calculate discrete logarithms it can easily break Diffie-Hellman key exchange protocol. Hash-based digital signature systems have performance problems. The efficiency of the scheme is analyzed in the article. The Merkle digital signature algorithm using recursion was implemented. A performance analysis was conducted. To improve the efficiency in the implementation of this algorithm, the recursion was replaced by loops. An analysis of the resulting implementation was carried out. Modified implementation showed very good results.

Keywords: Quantum Computer, Diffie-Hellman, Merkle Digital Signature.

1 Introduction

Today, in digital era, data security is top of mind for many businesses and governments to protect: financial records, medical histories, military strategy, confidential information and more. Information technologies are presented in all spheres of human life. But openness and publicity of network and Internet services, together with the evolution of cyber-attacks, significant increase of ICT users and the amount of information processed, stored and transmitted by ICT, jeopardize the information confidentiality, which is usually provided by symmetric and symmetric cryptography, that have certain shortcomings [1, 4-6].

2 Problem statement

As it is known symmetric methods, in particular, are characterized by the problem of secret keys distribution, and asymmetric methods are slow and require significant computing resources [6-10]. The problem of secret keys distribution is one of the most important problems related to security of information transmitted over telecommunications channels. Once users receive a shared secret key, cryptograms can be sent from any unprotected channel, possibly even over a channel that is prone to full passive eavesdropping (such as public announcements through the media). However, to obtain a shared secret key, two users who initially have no shared secret information must use some very reliable and secret channel. Since interception is a series of measurements performed by an eavesdropper, no matter how complex they may be from a technical point of view, any channel can be listened to. This poses a serious security threat, which is why it is important to detect an eavesdropping device. It should be emphasized that there is no classical cryptographic mechanism that would give a full guarantee that the key was not intercepted during transmission on a traditional (not quantum) communication channel.

In addition, the stability of all traditional cryptosystems depends on the computational capabilities of intruder and is based on the hypothetical impossibility of solving a certain class of mathematical problems in polynomial time [1, 6]. However, this hypothesis can be refuted with the help of multi-qubit quantum computers. Now active work all over the world is being conducted to develop and improve quantum computers. Cryptosystems, which are used in practice are vulnerable to attacks by quantum computers. The security of these systems is based on the problem of factorization of large numbers and the calculation of discrete logarithms, and a quantum computer can easily solve these problems [15, 16].

Scientists and experts are actively working on the creation of quantum computers. GOOGLE Corporation, NASA the association USRA (Universities Space Research Association) and D-Wave teamed-up to develop quantum processors [3].

Quantum computers can break existing public-key crypto systems. Quantum computer solves the discrete logarithm problem both for finite fields and elliptic curves. Being able to efficiently calculate discrete logarithms it can break Diffie-Hellman key exchange protocol [6, 10-14].

Public-key cryptography is used in different products on different platforms and in various fields. Many commercial products use public-key cryptography, the number of which is actively growing. Public-key cryptography is also widely used in operating systems from Microsoft, Apple, Sun, and Novell. It is used in secure phones, Ethernet, network cards, smart cards, and it is widely used in cryptographic hardware. Public-key technology is used in protected Internet communications, such as S /

MIME, SSL and S / WAN. It is used in government, banks, most corporations, different laboratories and educational organizations. Breaking existing public-key cryptosystems will cause complete chaos.

There are two major classes of methods to replace traditional cryptography – the first is quantum cryptography (based on the fundamental difference, the use of the unique capabilities of quantum mechanics), the second is post-quantum cryptography (for example, lattice-based cryptosystems, digital signature, syndrome-based cryptosystems and others).

Quantum cryptography can provide protection against interception of key distribution, because, unlike classical cryptography, it is based on the laws of physics, and not on the fact that successful interception would require huge computing power. Due to the above properties of quantum systems, the attacker makes several errors in the information transmitted by individual photons, which can be detected by legitimate users. Note that the laws of quantum mechanics allow not only to detect perturbations of states, but also link error rate in the measurements of legitimate users with the amount of information that could be obtained by an attacker. Quantum key distribution protocols main advantage is that they, in contrast to most classical schemes, have theoretical and informational stability, independent of the computational and other technical capabilities of the attacker [1, 4-8].

On the other hand, post-quantum methods of information security are rapidly developing and great work is being done to create cryptosystems that are protected from quantum computer attacks. Such are hash-based digital signature systems. The security of these crypto systems is based on the resistance to collisions of hash functions.

3 Lamport–Diffie one-time signature scheme

Lamport–Diffie one-time signature scheme was offered, this scheme is a hash-based digital signature. In this scheme, key generation and signature generation are efficient, but the size of the signature is quite large.

For the signature key X, 2n random lines of the size n are generated.

X= (xn-1[0], xn-1[1], …, x0[0], x0[1]) ∈ {0,1} n,2n

Verification key Y= (yn-1[0], yn-1[1], …, y0[0], y0[1]) ∈ {0,1} n,2n

The verification key is calculated as follows:

yi[j] = f(xi[j]), 0<=i<=n-1, j=0,1

f – is one way function:

f: {0,1} n →{0,1} n;

4 Winternitz one-time signature scheme

Winternitz one-time Signature Scheme was proposed to reduce the size of the signature. In this scheme was chosen the argument w ∈ N, and is calculated r =

[s/w]+[([log2 [s/w]] + 1 + w)/w]. R random numbers X1, X2, … Xr ∈{0,1}s was chosen, concatenation of which is X - private key. Are calculated Yi = h2w−1(Xi), the public key is Y = h(Y1||...||Yr). The message M is divided into s/w blocks b1, …, bs/w with the length w, if it is necessary, on the left are added zeroes. Later the checksum C =$\sum_{i=1}^{s/w} 2^{w}$-bi is calculated. It is shown in the Fig 1.

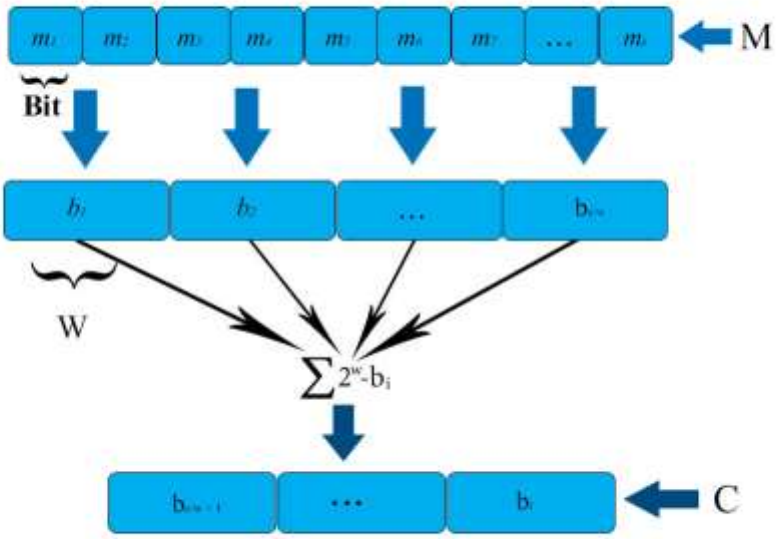

Fig. 1. Winternitz one-time Signature Scheme

The binary representation of C is divided into [([log2 [s/w]] + 1 + w)/w] blocks, $b_{s/w+1}$, ..., b_r with the length w.

Is calculated sigi = $h^{bi}(X_i)$ for i = 1, ..., r, the signature of the letter is sig = (sig_1||...||sig_r).

For signature verification are calculated b_1, ..., $b_{r.}$ For I = 1, ..., r is calculated signew$_i$= $h^{2w-1-bi}(sig_i)$ = $h^{2w-1-bi}(h^{bi}\ (X_i))$ = h^{2w-1}(Xi) = Y_i, if h(signew$_{1,\ ...,}$ signew$_r$)=Y, then the signature is correct.

The biggest problem of one-time signature schemes is the transfer of public key. It is necessary to make sure that the public key has not been changed, therefore it is necessary to use as little number of public keys as possible, and to make them shorter.

5 Merkle digital signature scheme

The Merkle crypto system was proposed to solve the problem of a one-time key pair. Merkle uses a binary tree to replace a large number of verification keys with one public key, the root of the binary tree. This cryptosystem uses a Lamport or Winternitz one-time signature scheme and a cryptographic hash function [17-20].

Key generation. The size of the tree must be H>=2 and using one public key 2H documents can be signed. Signature and verification keys are generated; Xi, Yi, 0<=i<=2H. Xi- is the signature key, Yi- is the verification key. Signature keys are hashed using the hash function h:{0,1}*→{0,1}n in order to get the leaves of the tree.

The concatenation of two previous nodes is hashed in order to get the parent node,.

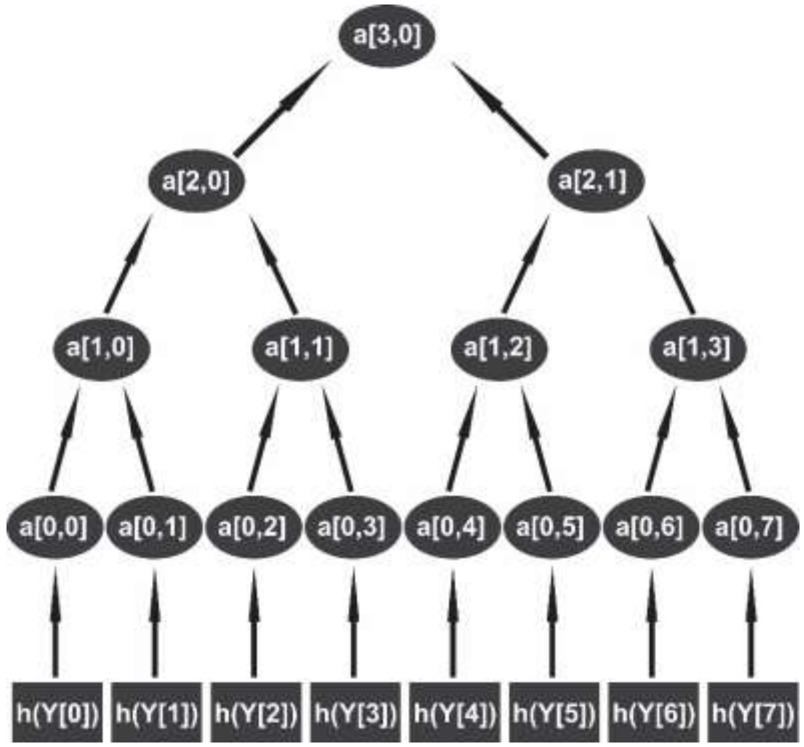

Fig. 2. Merkle tree with *H=3*.

a[i,j] are the nodes of the tree;

a[1,0]=h(a[0,0] || a[0,1])

The root of the tree is the public key of the signature - *pub*, $\boldsymbol{2^H}$ pairs of signature keys must be generated in order to calculate the public *k*, and the hash function h is used $\boldsymbol{2^{H+1}-1}$ times.

Message signature. A message of any size can be signed being transformed to size of *n* by means of hashing *h (m) = hash*, An arbitrary one-time key X_{any} is used, and the signature is a concatenation of one-time signature, one-time verification key, index of a key and all fraternal nodes according to the selected arbitrary key with the index “any”.

$Signature= (sig||any|| Y_{any}||auth_{0},\ldots,auth_{H-1})$

Signature verification. The one-time signature is checked using the selected verification key, if the verification is true, all the *a[i, j]* are calculated using "auth", index "any" and Y_{any}. The signature is verified, if the root of the tree matches the public key.

The algorithm of this system was implemented and recursion is used.

```
Recursion:
    1.  Importing necessary libs
    2.  Define class
    3.  Defining “ alt_hashes(hashes) “ method
    4.  Set list “ arr ”
    5.  If  hashes == “ ”, raise Exception
    6.  Foreach loop
        6.1.   sorting hashes and appending into arr
    7.  Length_of_block == length of arr
    8.  While loop, if length is odd, copy last element in list
        8.1.   append it into arr list
    9.  Set list “ another_arr ”
    10. Foreach loop
        10.1. For loop with range from 0 to length of “ arr “ and iteration by 2
            10.1.1.Define variable with “ sha512() “ value
            10.1.2.Hash elements that are in “ arr “ list
            10.1.3.Apennd them into new “ another_arr “ list
            10.1.4.Return this list in hex
    11. Set list “ hash_arr ”
    12. Foreach loop
        12.1. Generate Hex and put it into “ hash_arr ” list
    13. Create message put it in “ st “ variable
    14. Convert “ st “ value in binary
    15. First_secret_key = hash_arr[0]
    16. Second_secret_key = hash_arr[1]
    17. Generate “ one-time signature ”
        17.1. If st == 0
            17.1.1.Choose  “ First_secret_key “ bit
        17.2. Else
            17.2.1.Choose  “Second _secret_key “ bit
    18. First_pub_key = hash(hash_arr[0])
    19. Second_pub_key = hash (hash_arr[1])
    20. Encryption
        20.1. Concatenate “ one-time signature ” with message’s hash
    21. Verification of  “ one-time signature ”
        21.1. If bit of  “ one-time signature ” == 0
            21.1.1.Compare with “ First_secret_key “ bit
        21.2. Else
            21.2.1.Compare  with “Second _secret_key “ bit
    22. Verification of  “ signature ”
        22.1. Concatenate siblings with each other
        22.2. If this equals to public key
```

```
                    22.2.1.Sign is correct
                22.3. Else
                    22.3.1.   Sign is not correct
```

Fig. 3. Merkle Algorithm with Recursion.

This example shows the implementation of the Merkle algorithm using recursion. The public key generation time for 8 elements is 0.0159 seconds, the encryption time is 0.01684, and the verification time is 0.0288883.

Recursion was changed to loops in order to improve the efficiency.

```
Implementation using loops:
            1.  Importing necessary libs
            2.  Define class
            3.  Defining “ loop_hashes(hashes) “ method
            4.  Set list “ arr ”
            5.  If  hashes == “ ”, raise Exception
            6.  Foreach loop
                6.1.    sorting hashes and appending into arr
            7.  Length_of_block == length of arr
            8.  While loop, if length is odd, copy last element in list
                8.1.    append it into arr list
            9.  Set list “ another_arr ”
            10. Set i = 0
            11. While loop, Length_of_block > 1
                11.1. Set to hash_f sha512()
                11.2. Concatenate arr[i] and arr[i+1]
                11.3. append it into “another_arr” list
                11.4. append arr[i+1] to “auth_list” list
                11.5. i = i + 2
                11.6. if i equal to “Length_of_block”
                    11.6.1.set to “Length_of_block” “Length_of_block /  2”
                    11.6.2.i = 0
                    11.6.3.set “another_arr” to “arr”
                    11.6.4.empty “another_arr”
                11.7. return “arr”
            12. Set list “ hash_arr ”
            13. Foreach loop
                13.1. Generate Hex and put it into “ hash_arr ” list
            14. Create message put it in “ st “ variable
            15. Convert “ st “ value in binary
            16. First_secret_key = hash_arr[0]
            17. Second_secret_key = hash_arr[1]
            18. Generate “ one-time signature ”
                18.1. If st == 0
                    18.1.1.Choose  “ First_secret_key “ bit
                18.2. Else
                    18.2.1.Choose  “Second _secret_key “ bit
            19. First_pub_key = hash(hash_arr[0])
            20. Second_pub_key = hash (hash_arr[1])
            21. Encryption
                21.1. Concatenate “ one-time signature ” with message’s hash
```

```
22.  Verification of " one-time signature "
     22.1. If bit of " one-time signature " == 0
           22.1.1.Compare with " First_secret_key " bit
     22.2. Else
           22.2.1.Compare with "Second _secret_key " bit
23.  Verification of " signature "
     23.1. Concatenate siblings with each other
     23.2. If this equals to public key
           23.2.1.Sign is correct
     23.3. Else
           23.3.1.  Sign is not correct
```

Fig. 4. Merkle algorithm with loops.

This example shows the implementation of the Merkle algorithm using loops. The "public key" generation time for 8 elements is 0.0061761 seconds, the encryption time is 0.0080878, and the confirmation time is 0.0181923. Here is the graph, which reflects the results:

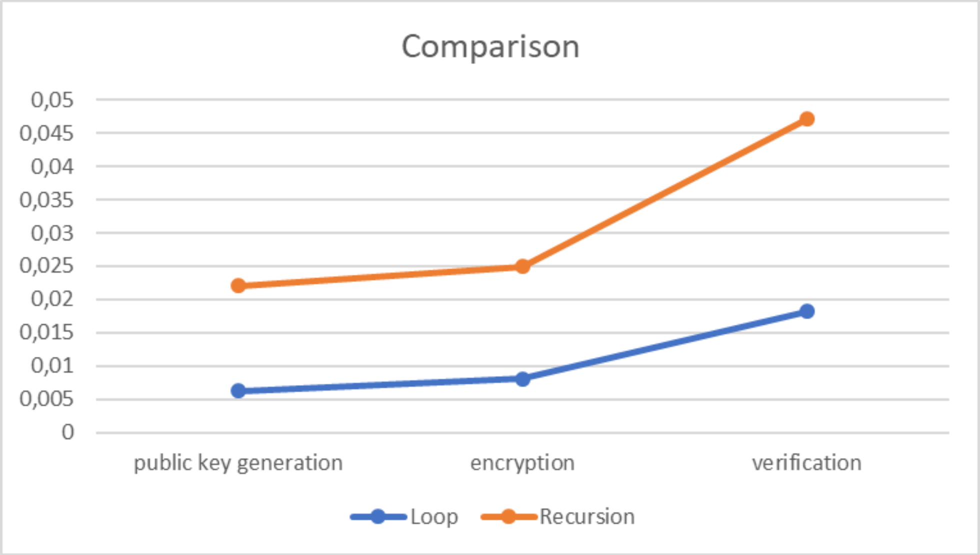

Fig. 5. A figure caption is always placed below the illustration. Short captions are centered, while long ones are justified. The macro button chooses the correct format automatically.

Thus, it could be seen, that the implementation changes gave rather good results.

6 Conclusions

In this work were described hash-based digital signature systems. These systems are safe against quantum computer attacks. Hash-based digital signature systems have performance problems. The efficiency of the scheme is analyzed in the article. The Merkle digital signature algorithm using recursion was implemented. A performance analysis was conducted. To improve the efficiency in the implementation of this algorithm, the recursion was replaced by loops. An analysis of the resulting implementation was carried out. Modified implementation showed very good results.

Acknowledgement:

The project was conducted in the frame of the research grant PHDF-19-519.

References

1. Korchenko O., Vasiliu Y., Gnatyuk S. Modern quantum technologies of information security against cyber-terrorist attacks, Aviation, Vol. 14, №3, pp. 58-69, 2010.
2. Daniele Di Tullio, Ankan Pal. A New Method for Geometric Interpretation of Elliptic Curve Discrete Logarithm Problem https://arxiv.org/pdf/1909.08901.pdf
3. Universities Space Research Association https://www.usra.edu/
4. Quantum Computation and Information. From Theory to Experiment / Imai H., Hayashi M. (eds.). — Springer-Verlag: Berlin, Heidelberg, 2006. — P. 235
5. N. Gisin, G. Ribordy, W.Tittel, H. Zbinden , Quantum cryptography, Rewiews of Modern Physics, 2002, vol. 74, №1, pp. 145-195.
6. M. Nilsen and I. Chuang, Quantum Computing and Quantum Information. Cambridge University Press, 2006, 824 p.
7. Gnatyuk S., Akhmetov B., Kozlovskyi V., Kinzeryavyy V., Aleksander M., Prysiazhnyi D. New Secure Block Cipher for Critical Applications: Design, Implementation, Speed and Security Analysis, Advances in Intelligent Systems and Computing, Vol. 1126, pp. 93-104, 2019.
8. Gnatyuk S., Kinzeryavyy V., Kyrychenko K., Yubuzova Kh., Aleksander M., Odarchenko R. Secure Hash Function Constructing for Future Communication Systems and Networks, Advances in Intelligent Systems and Computing, Vol. 902, pp. 561-569, 2019.
9. Kalimoldayev M., Tynymbayev S., Gnatyuk S., Khokhlov S., Magzom M., Kozhagulov Y. Matrix multiplier of polynomials modulo analysis starting with the lower order digits of the multiplier, News of the National Academy of Sciences of the Republic of Kazakhstan, Series of Geology and Technical Sciences, №4 (436), pp. 181-187, 2019.
10. Kalimoldayev M., Tynymbayev S., Gnatyuk S., Ibraimov M., Magzom M. The device for multiplying polynomials modulo an irreducible polynomial, News of the National Academy of Sciences of the Republic of Kazakhstan, Series of Geology and Technical Sciences, №2 (434), pp. 199-205, 2019.
11. M. Zaliskyi, R. Odarchenko, S. Gnatyuk, Yu. Petrova. A. Chaplits, Method of traffic monitoring for DDoS attacks detection in e-health systems and networks, CEUR Workshop Proceedings, Vol. 2255, pp. 193-204, 2018.
12. Hu Z., Gnatyuk S., Kovtun M., Seilova N. Method of searching birationally equivalent Edwards curves over binary fields, Advances in Intelligent Systems and Computing, Vol. 754, pp. 309-319, 2019.
13. S. Gnatyuk, A. Okhrimenko, M. Kovtun, T. Gancarczyk, V. Karpinskyi, Method of Algorithm Building for Modular Reducing by Irreducible Polynomial, Proceedings of the 16th International Conference on Control, Automation and Systems, Oct. 16-19, Gyeongju, Korea, 2016, pp. 1476-1479.

14. M. Kovtun, S. Gnatyuk, V. Kovtun, A. Okhrimenko, Development of a search method of birationally equivalent binary Edwards curves for binary Weierstrass curves from DSTU 4145-2002, Proceedings of 2nd Intern. Scientific-Practical Conf. on the Problems of Infocommunications. Science and Technology (PIC S&T 2015), Kharkiv, Ukraine, October 13-15, 2015, pp. 5-8.
15. Guang Hao Low, Artur Scherer, and Dominic W. Berry, Black-Box Quantum State Preparation without Arithmetic Yuval R. Sanders, Phys. Rev. Lett. 122, 020502 – Published 16 January 2019
16. Liu J. et al. (2019) Formal Verification of Quantum Algorithms Using Quantum Hoare Logic. In: Dillig I., Tasiran S. (eds) Computer Aided Verification. CAV 2019. Lecture Notes in Computer Science, vol 11562. Springer, Cham
17. Iavich, M., Gagnidze, A., Iashvili, G., Hash based digital signature scheme with integrated TRNG, CEUR Workshop Proceedings, 2018
18. Gagnidze, A., Iavich, M., Iashvili, G., Novel version of merkle cryptosystem, Bulletin of the Georgian National Academy of Sciences, 2017
19. M. Iavich, G. Iashvili, A.Gagnidze, S. Gnatyuk, V. Vialkova; Lattice Based Merkle; IVUS2019; CEUR-WS.org; 2019
20. Fedushko S., Davidekova M. Analytical service for processing behavioral, psychological and communicative features in the online communication. The International Workshop on Digitalization and Servitization within Factory-Free Economy (D&SwFFE 2019) November 4-7, 2019, Coimbra, Portugal. Procedia Computer Science. Volume 160, 2019, Pages 509-514. https://doi.org/10.1016/j.procs.2019.11.056
21. Anisimova O., Lukash H., Syerov Yu. Formation of the Image of the Specialist in Social Networks. CEUR Workshop Proceedings. Vol 2616: Proceedings of the 2nd International Workshop on Control, Optimisation and Analytical Processing of Social Networks (COAPSN-2020), Lviv, Ukraine, May 21, 2020. p. 39-52. http://ceur-ws.org/Vol-2616/paper4.pdf
22. Gagnidze A.G., Iavich M.P., Iashvili G.U., Analysis of Post Quantum Cryptography use in Practice, Bulletin of the Georgian National Academy of Sciences, vol. 11, no. 2, 2017, p.29-36.