Review and Analysis of the Classical and Post-Quantum Ring Signature Algorithms

Anton Leevik[†], Vladislav Beliaev[†], Boris Stasenko[†], Vadim Davydov[†], Sergey Bezzateev[‡]
[†]Faculty of Secure Information Technologies, ITMO University,
[‡] Department of Information Security Technologies, Saint Petersburg University of Aerospace Instrumentation
Saint Petersburg, Russia
Email: {agleevik, v.beliaev,boris.stasenko, vadim.davydov}@niuitmo.ru, bsv@aanet.ru

***Abstract*—In this paper a review and analysis of ring signature algorithms based on discrete logarithm and code-based problems is made. The authors consider Linkable and Multilayer Linkable Spontaneous Anonymous Group Signatures which are based on the discrete logarithm problem. Taking into the account the fact that quantum computers have already been developed, we look at different variations of code-based signatures (linkable, multilayer linkable, traceable and threshold) and analyse the efficiency of their use. Finally, the analysis of these signatures is made and standard code-based algorithms are compared.**

***Index Terms*—ring signature, discrete logarithm problem, post-quantum cryptography, code-based signature**

I. Introduction

These days, there is a big variety of modern cryptographic algorithms. However, with the advent of quantum computers and a significant rise in their performance, some of the algorithms become inefficient. Since such popular public-key algorithms as RSA, El Gamal, and Rabin are based on factorisation and discrete logarithm problems, which could be now easily solved by quantum computer with Shor's algorithm, the main part of modern cryptography is in danger [1].

Fortunately, there is another direction in cryptography called post-quantum cryptography, where mathematical problems cannot be theoretically solved by quantum computer, which sounds promising for the entire cryptographic community. Nevertheless, post-quantum algorithms are vulnerable to some other attacks, and the main idea for now is to make them as much secure as possible.

One of the most used and essential primitive, which can be met in many cryptocurrency and blockchain systems, is a ring signature. One of the most well-known uses of such signature is the Monero RingCT protocol [2]. Monero is one of the most popular cryptocurrencies in the world, known for its privacy, security, and anonymity.

The main issue is that this primitive is based on the discrete logarithm problem, which can threaten all such systems in the near future. In this paper, we make a review and analysis of the ring signature and its modifications in two versions: based on the discrete logarithm problem, and hard coding problems.

In Section II, there is a short review of multilayer ring signatures and primitives which are a crucial part of such signatures and their analysis. Section III consists of five variations of ring signatures algorithms which are resistant to possible attacks on quantum computers and the full analysis of their expediency and performance. In Section IV, conclusions are made and future perspectives are considered.

This work was prepared with the Financial support of the Ministry of Science and Higher Education of the Russian Federation under the agreement No. 075-15-2019-1707 from 22.11.2019 (identifier RFMEFI60519X0189, internal number 05.605.21.0189).

II. A Short Review of Ring Signatures Based on Discrete Logarithm Problem

This Section consists of a short review of ring signatures based on the discrete logarithm problem. Starting from the essential cryptographic primitives used in ring signatures, we also cover two algorithms of Linkable and Multilayer Linkable Spontaneous Anonymous Group signatures. This Section is concluded by the analysis of effective use of ring signatures.

A. Keccak-256 Hashing Function

Choosing an optimal hashing algorithm is crucial for the secure generation of user addresses and keys. Producing the same hash output for two different inputs is known as a collision, which chance must be negligible for complex cryptosystems. Considering the effective uniqueness of hashes, they are also frequently used as identifiers in blockchain systems. Keccak hashing algorithm, which provides 32-byte hashes, is used in the multilayer ring signatures. The way of how this function works is described in [3].

B. Edwards25519 Elliptic Curve

Before moving on to the algorithms themselves, some of their main parameters should be introduced. We generate a key pair using the Edwards25519 elliptic curve, which is a curve of the Twisted Edwards family.

The Edwards25519 is birationally equivalent to the Curve25519 [4]. Both Curve25519 and Edwards25519 were introduced by D. Bernstein *et al.* [5]–[7].

From the Cryptonote whitepaper [8], we get the following constants for the Edwards25519 curve:

- q – a prime number: $q = 2^{255} - 19$;
- G – a generator point many operations start with: $G = (x; y)$, where $y = -4 \cdot 5^{-1} \mod q$;
- l – a prime order of the generator point G (or base point): $l = 2^{252}+27742317777372353535851937790883648493$;

- d – an element of $\mathbb{F}_q$ used in the curve equation below: $d = -121665 \cdot 121666^{-1} \mod q$;
- E – an elliptic curve equation: $-x^2 + y^2 = 1 + dx^2y^2$;
- Let $\mathcal{H}_p$ be defined as $\mathcal{H}_p(x) = keccak(x) \cdot G$;
- Let h_s be defined as $h_s(x) = keccak(x) \mod l$.

C. LSAG Signature

Linkable Spontaneous Anonymous Group signature (LSAG) is a ring signature over a set of u users.

- Let u be a number of users in the ring signature.

From [9], [10], we obtain the following steps for the signature generation and verification algorithms:

1) **Key generation**
   Consider a set of public keys PK:
   $$PK = (P_1, P_2, \ldots, P_i, \ldots, P_u),\ \forall i \in \{1, 2, \ldots, u\}$$
   - Let $\mathbb{F}_l^* \subset \mathbb{F}_q$;
   - Let G be the Edwards25519 base point;
   - Let the signer be with index π, $\pi \in \{1, 2, \ldots, u\}$.

   The signer with index π knows a secret key x_π such that $P_\pi = x_\pi \cdot G$. The secret key is randomly chosen from $\mathbb{F}_l^*$.

   The rest of the public and private keys for all users are generated in the same way, but the ring signature algorithm uses only the signer's private key.

   - Let I_π be the key image for the given index π such that $I_\pi = x_\pi \cdot \mathcal{H}_p(P_\pi)$.
2) **Signature generation**
   Suppose some user with index π, $\pi \in \{1, 2, \ldots, u\}$ signs a message on behalf of the ring. This user has:
   - Public key: P_π;
   - Secret key: x_π;
   - Image key: I_π.

   We will use the auxiliary values to generate a signature:
   - Let L, R be the Edwards25519 points;
   - Let c, s be the values of $\mathbb{F}_l^*$.

   With the given keys, the next algorithm will be the following:

   a) Let M be a given message;
   b) Let α be randomly chosen from $\mathbb{F}_l^*$. Compute:
      - $L_\pi = \alpha \cdot G$
      - $R_\pi = \alpha \cdot \mathcal{H}_p(P_\pi)$
   c) Let $c_{\pi+1} = h_s(M, L_\pi, R_\pi)$
   d) $\forall i \in \{\pi + 1, \pi + 2, \ldots, u, 1, 2, \ldots, \pi - 2, \pi - 1\}$, replacing $u + 1 \to 1$:
      - Let s_i be randomly chosen from $\mathbb{F}_l^*$. Compute:
        i) $L_i = s_i \cdot G + c_i \cdot P_i$
        ii) $R_i = s_i \cdot \mathcal{H}_p(P_i) + c_i \cdot I_\pi$
      - $c_{i+1} = h_s(M, L_i, R_i)$
   e) $s_\pi = \alpha - c_\pi \cdot x_\pi \mod l$

   The algorithm outputs a signature:
   $\sigma = (I_\pi, c_1, s_1, \ldots, s_u)$
3) **Signature verification**
   Given a message M, a public key set PK:
   $$PK = (P_1, P_2, \ldots, P_i, \ldots, P_u),\ \forall i \in \{1, 2, \ldots, u\}$$
   and a signature:
   $\sigma = (I, c_1, s_1, \ldots, s_u)$

   With the given values, the next algorithm will be the following:

   a) Let $c_1' = c_1$.
   b) $\forall i \in \{1, 2, \ldots, u\}$, replacing $u + 1 \to 1$:
      - $L_i' = s_i \cdot G + c_i \cdot P_i$
      - $R_i' = s_i \cdot \mathcal{H}_p(P_i) + c_i \cdot I$
      - $c_{i+1}' = h_s(M, L_i', R_i')$
   c) Check that $c_1 = c_{u+1}'$

   If the equality in (c) is true, the signature is valid.

D. MLSAG Signature

Multilayer Linkable Spontaneous Anonymous Group signature (MLSAG) is a ring signature over a set of u key-vectors, where each key-vector is a collection of a public keys $\bar{P}_i = (P_{i,1}, P_{i,2}, \ldots, P_{i,a})$, $\forall i \in 1, 2, \ldots, u$.

- Let u be a number of users in the ring signature;
- Let a be a number of one-time addresses for each user.

From [2], [10], [11], we obtain the following steps for the signature generation and verification algorithms:

1) **Key generation**
   Consider an $a \times u$ matrix of public keys:
   $$PK = \begin{bmatrix} P_{1,1} & P_{2,1} & \ldots & P_{\pi,1} & \ldots & P_{u,1} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ P_{1,a} & P_{2,a} & \ldots & P_{\pi,a} & \ldots & P_{u,a} \end{bmatrix}$$
   - Let $\mathbb{F}_l^* \subset \mathbb{F}_q$;
   - Let G be the Edwards25519 base point;
   - Let the signer be with index π, $\pi \in \{1, 2, \ldots, u\}$.

   The signer with index π knows the secret keys $x_{\pi,j}$ such that $P_{\pi,j} = x_{\pi,j} \cdot G$, $\forall j \in \{1, 2, \ldots, a\}$.The secret key values are randomly chosen from $\mathbb{F}_l^*$.

   The rest of the public and private keys are generated in the same way, but the ring signature algorithm uses only the signer's private keys.

   - Let $\bar{I} = (I_1, I_2, \ldots, I_j, \ldots, I_a)$ be the key image vector for the given index π, $\pi \in \{1, 2, \ldots, u\}$ such that $I_j = x_{\pi,j} \cdot \mathcal{H}_p(P_{\pi,j})$, $\forall j \in \{1, 2, \ldots, a\}$.
2) **Signature generation**

Suppose some user with index π, $\pi \in \{1, 2, \dots, u\}$ signs a message on behalf of the ring. This user has:

- Public key-vector: $(P_{\pi,1}, P_{\pi,1}, \dots, P_{\pi,a})$;
- Secret key-vector: $(x_{\pi,1}, x_{\pi,1}, \dots, x_{\pi,a})$;
- Image key-vector: $(I_1, I_2, \dots, I_a)$.

We use the auxiliary values to generate a signature:

- Let L, R be the Edwards25519 points;
- Let c, s be the values of $\mathbb{F}_l^*$.

With the given keys, we follow the next algorithm:

a) Let M be a given message;

b) Let α_j, $\forall j \in \{1, 2, \dots, a\}$ be randomly chosen values from $\mathbb{F}_l^*$. Compute:
   - $L_{\pi,j} = \alpha_j \cdot G$
   - $R_{\pi,j} = \alpha_j \cdot \mathcal{H}_p(P_{\pi,j})$

c) Let $c_{\pi+1} = h_s(M, L_{\pi,j}, R_{\pi,j}, \dots, L_{\pi,a}, R_{\pi,a})$

d) $\forall i \in \{\pi+1, \pi+2, \dots, u, 1, 2, \dots, \pi-2, \pi-1\}$, replacing $u+1 \to 1$:
   - Let $s_{i,j}$, $\forall j \in \{1, 2, \dots, a\}$ be randomly chosen values from $\mathbb{F}_l^*$. Compute:
     i) $L_{i,j} = s_{i,j} \cdot G + c_i \cdot P_{i,j}$
     ii) $R_{i,j} = s_{i,j} \cdot \mathcal{H}_p(P_{i,j}) + c_i \cdot I_j$
   - $c_{i+1} = h_s(M, L_{i,1}, R_{i,1}, \dots, L_{i,a}, R_{i,a})$

e) $s_{\pi,j} = \alpha_j - c_\pi \cdot x_{\pi,j} \mod l$, $\forall j \in \{1, 2, \dots, a\}$

The algorithm outputs a signature:

$\sigma = (I_1, \dots, I_a, c_1, s_{1,1}, \dots, s_{1,a}, \dots, s_{u,1}, \dots, s_{u,a})$

3) **Signature verification**
Given message M and public key matrix PK:

$$PK = \begin{bmatrix} P_{1,1} & P_{2,1} & \dots & P_{u,1} \\ \dots & \dots & \dots & \dots \\ P_{1,u} & P_{2,a} & \dots & P_{u,a} \end{bmatrix}$$

and a signature:

$\sigma = (I_1, \dots, I_a, c_1, s_{1,1}, \dots, s_{1,a}, \dots, s_{u,1}, \dots, s_{u,a})$

With the given values, we follow the next algorithm:

a) Let $c'_1 = c_1$.

b) $\forall i \in \{1, 2, \dots, u\}$, replacing $u+1 \to 1$:
   - $\forall j \in \{1, 2, \dots, a\}$ compute:
     i) $L'_{i,j} = s_{i,j} \cdot G + c_i \cdot P_{i,j}$
     ii) $R'_{i,j} = s_{i,j} \cdot \mathcal{H}_p(P_{i,j}) + c_i \cdot I_j$
   - $c'_{i+1} = h_s(M, L'_{i,1}, R'_{i,1}, \dots, L'_{i,a}, R'_{i,a})$

c) Check that $c_1 = c'_{u+1}$

If the equality in (c) is true, the signature is valid.

E. Signatures Comparison

In the process of creating LSAG and MLSAG signatures, auxiliary values are generated. These values c_1, s_i for LSAG, and $s_{i,j}$ for MLSAG, where $\forall i \in \{1, 2, \dots, u\}$ and $\forall j \in \{1, 2, \dots, a\}$, are necessary to check the signature for correctness.

The Edwards25519 elliptic curve provides small private (32 bytes) and public (32 bytes) keys. Public and image keys are usually represented as 64-byte points on elliptic curve, but could be compressed up to 32 bytes [2]. Elements of $\mathbb{F}_l^*$ are also represented as 32-byte values.

TABLE I
COMPARISON OF RING SIGNATURES BASED ON THE DISCRETE LOGARITHM PROBLEM

Signatures / Parameters	LSAG	MLSAG
Public key size	u	$a \times u$
Secret key size	1	a
Image key size	1	a
Signature size	$2 \times u + 2$	$2 \times (a \times u) + a + 1$

Table I shows the number of elements for each signature parameter. As mentioned above, each element is stored as a 32-byte value.

An increase in the MLSAG ring size by one user can be seen to correspond to an increase in the signature size by $2 \times a$ elements.

F. Analytics

In order to generate or verify the signature, we need to use key-vectors in MLSAG algorithm, which limits the use of large ring sizes in practice. This drawback can affect the algorithm speed and the allocated memory for storing signatures. The LSAG algorithm is faster because each user has only one public and one secret key. Accordingly, the generated signatures need less space.

Let us consider a ring where the number of users is gradually increased by one user, starting from $u = 4$ to $u = 100$. Each user has a fixed number of public keys in MLSAG algorithm, e.g. $a = 5$.

For the experiment LSAG and MLSAG signatures were modeled in Python 3.8 and computed on the machine with the following characteristics:

- Intel Core i7-7700K (4 cores, 4.2-4.5 GHz);
- DDR4 RAM, 4200 MHz, 8 GB.

The results of the execution time of the LSAG algorithm are demonstrated in Figures 1-2. The results of the execution time of the MLSAG algorithm are demonstrated in Figures 3-4.

Excluding small deviations, we can observe how the time for creating and verifying signatures increases linearly (Figures 1-4). The LSAG algorithm is about 5 times faster, because the MLSAG signature is modeled with 5 layers. Consequently, if the system uses the MLSAG signature, it should not be chosen a big number of users, as well as the number of public keys for each user. Otherwise, the execution time can be too long and the generated signatures can occupy a lot of memory space.

To sum up, it can be concluded that the classical ring signature algorithms are based on the complexity of the Elliptic Curve Discrete Logarithm Problem (ECDLP). It has

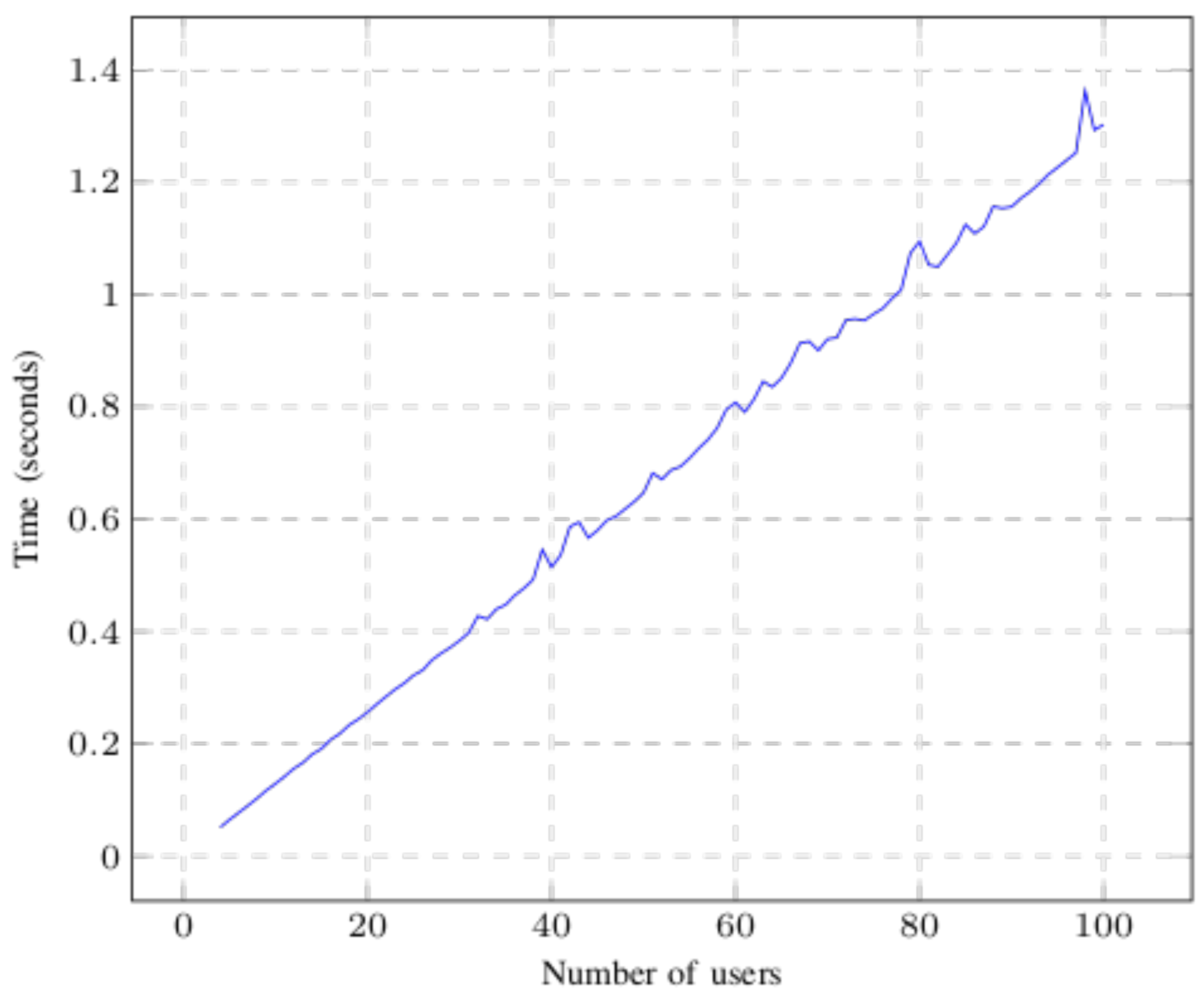

Fig. 1. Time dependence on the number of users in the LSAG signing algorithm.

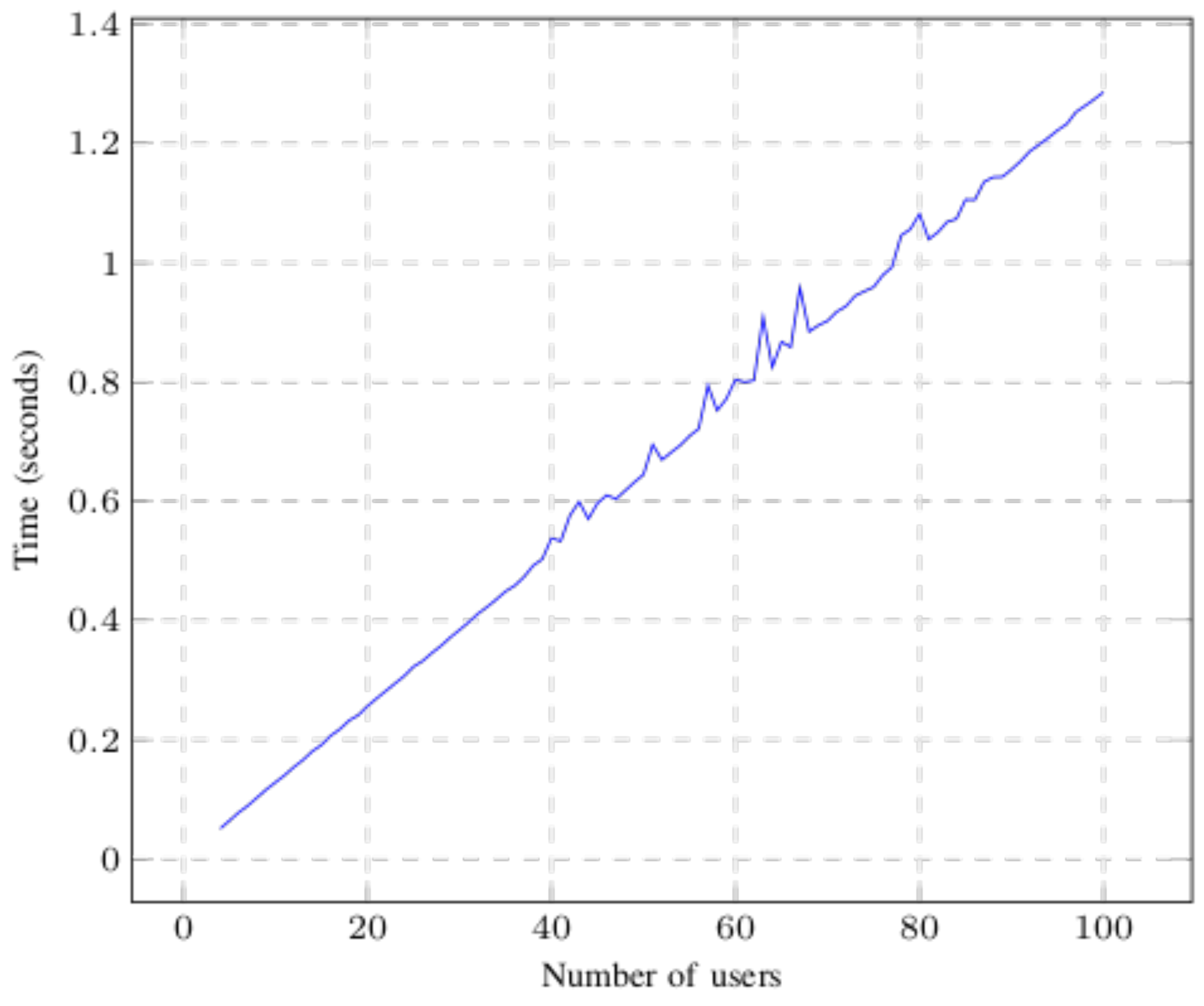

Fig. 2. Time dependence on the number of users in the LSAG verification algorithm.

been experimentally verified that multilayer ring signatures have one major drawback – the dependence on the number of users and the number of public keys for each user, which can be an obstacle for large systems.

III. Code-based Ring Signatures

In this Section the different types of code-based ring signatures are described. These types are based on the syndrome decoding problem which is proven to be NP-complete in [12]. In addition, the comparison of code-based ring signatures schemes and modeling results are presented.

The schemes of ring signatures are presented below, and the following notation for this purpose is used:

- n, k, t – the parameters of the used codes:
  - n – codeword length;
  - k – code dimension;
  - t – maximum correctable errors number;
- $w(x)$ – the Hamming weight of x;
- M – the message;
- u – the number of users in the ring.

A. Ring Signature

The ring signature scheme, which is an extension of T. Courtois et al.'s scheme [13], is described in [14] and its algorithm is presented below.

1) **Key generation**
   - Let the parameters $n, k, t \in \mathbb{N}$
   - For each user $\mathcal{P}_i$ where $i = 0, 1, \ldots, u-1$:
     - Choose a generating matrix G_i^0 and a parity-check matrix H_i^0.

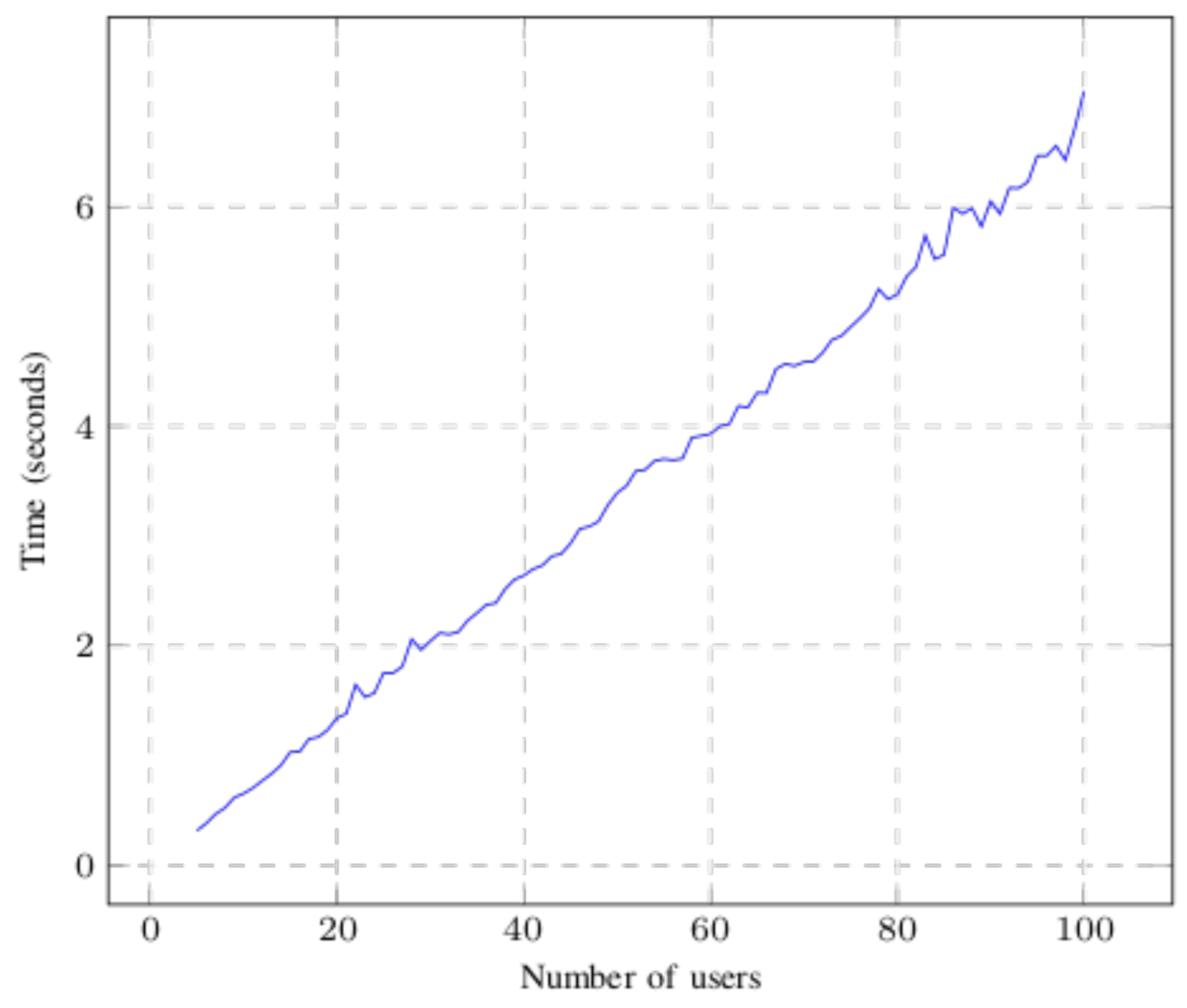

Fig. 3. Time dependence on the number of users in the MLSAG signing algorithm.

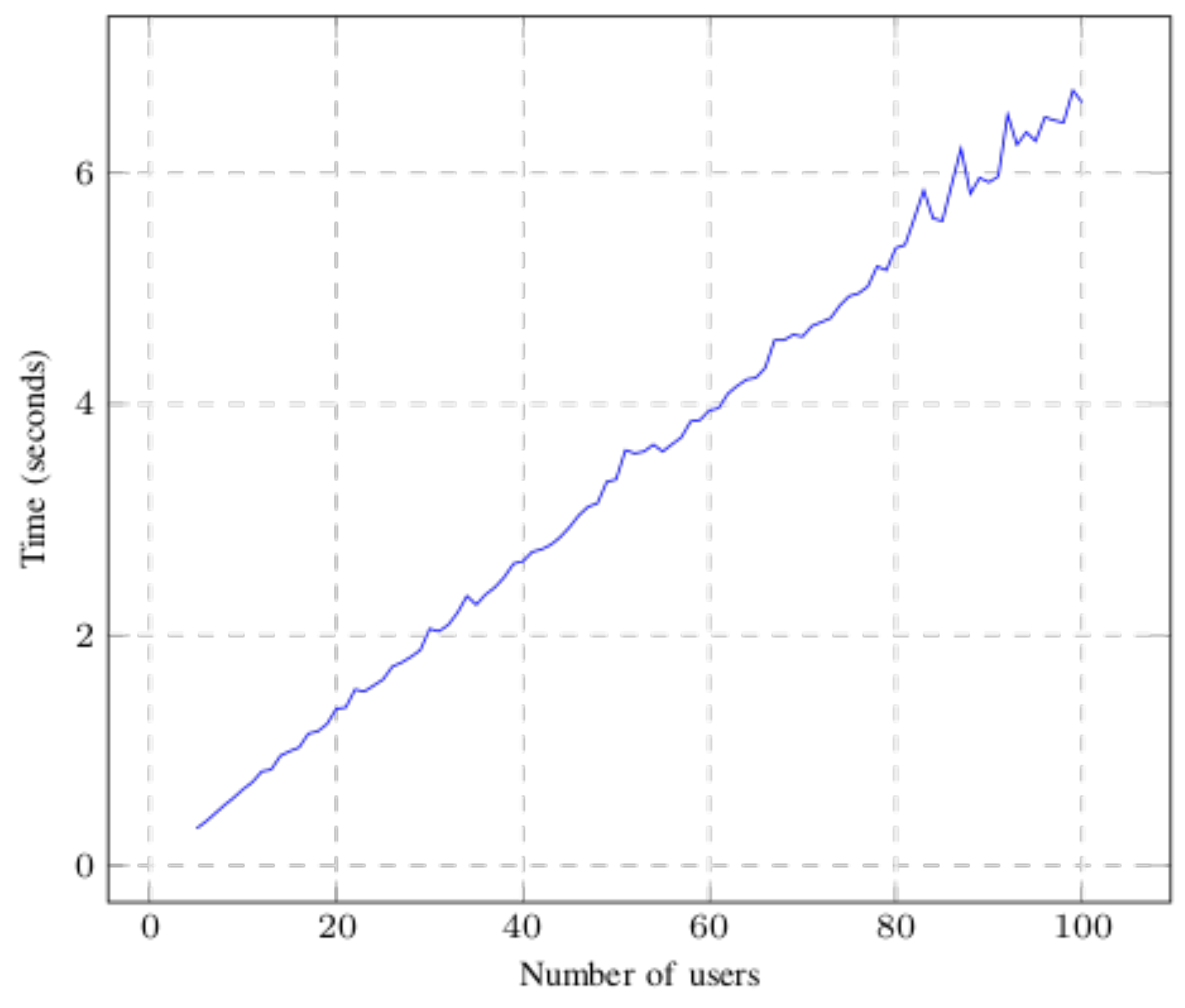

Fig. 4. Time dependence on the number of users in the MLSAG verification algorithm.

- Choose random $k \times k$ non-singular matrix U_i, $(n-k) \times (n-k)$ non-singular matrix V_i and $n \times n$ permutation matrix P_i.
- Compute $G_i = U_i G_i^0 P_i$ and $H_i = V_i H_i^0 P_i$.
- Public key is H_i, secret key is (G_i, U_i, V_i, P_i).

2) **Signature generation**
The operations in the algorithm are described for signer r and identical for all signers.
   a) Choose random vectors $\bar{s}_q \in F_2^{n-k}$ and compute $s_{r+1,q} = h(u|h(M)|\bar{s}_q)$ for $q = 0, 1, 2, \ldots$, where $h : \mathbb{F}_2^* \to \mathbb{F}_2^{n-k}$ is a hash function.
   b) Choose random vectors $z_{i,q} \in F_2^n$, $w(z_{i,q}) = t$, and compute $s_{i+1,q} = h(u|h(M)|H_i z_{i,q}^T \oplus s_{i,q})$ for $i = r+1, r+2, \ldots, u-1, 0, 1, \ldots, r-1$.
   c) Find $\bar{q}$ which is the smallest value of q such that $s_{r,\bar{q}} \oplus \bar{s}_{\bar{q}}$ is decodable by any designed algorithms.
   d) Set z_r such that $H_r z_r^T = s_{r,\bar{q}} \oplus \bar{s}_{\bar{q}}$.
   e) Compute the indexes I_{z_i} of z_i as $I_{z_i} = (p_1, p_2, \ldots, p_t)$, where $p_1 < p_2 < \cdots < p_t$ denote the positions of non-zero bits of z_i.
   f) Output $\sigma = (s_0, I_{z_0}, I_{z_1}, \ldots, I_{z_{u-1}})$ as the signature, where $s_0 = s_{0,\bar{q}}$.

3) **Signature verification**
   a) Recover z_i's from the indexes I_{z_i} for $i = 0, 1, \ldots, u-1$.
   b) Compute $s_{i+1} = h(u|h(M)|H_i z_i^T \oplus s_i)$ for $i = 0, 1, \ldots, u-1$: accept if $s_u = s_0$ and reject, otherwise.

B. Linkable Ring Signature

The linkable ring signature is a ring signature type which allows detecting if different messages are signed with the same private key.

The linkable ring signature scheme is described in [15], and its algorithm is presented below.

1) **Key generation**
   - Let the parameters $n, k, t \in \mathbb{N}$.
   - Choose matrices A, B from $\mathbb{F}_q^{(n-k)\times n}$, where $q = 2^m$, $m \in \mathbb{N}$.
   - Pick for each user $\mathcal{P}_i$ $e_i \in \mathbb{F}_q^n$ such that $w(e_i) \le t$ and compute $PK_i^T = Ae_i^T$, where $i = 0, 1, \ldots, u-1$.
   - The public key is PK_i, the secret key is e_i.

2) **Signature generation**
   a) For signer $\mathcal{P}_l$, calculate $T_l^T = Be_l^T$.
   b) Choose $v \in \mathbb{F}_q^n$ such that $w(v) \le t$ and calculate $d_{l+1} = h(L, T_l, M, Av^T, Bv^T)$ where $L = \{PK_i | i = 0, 1, \ldots, u-1\}$, $h : \mathbb{F}_q^* \to \mathbb{F}_q$ is a hash function.
   c) For $i = l+1, l+2, \ldots, u-1, 0, 1, \ldots, l-1$:
      - choose $c_i \in \mathbb{F}_q$,$r_{i1}, r_{i2} \in \mathbb{F}_q^n$ such that $w(r_{ij}) \le t, (j = 1, 2)$
      - calculate $r_{z,i} = r_{i1} + c_i r_{i2}$.
      - compute $d_{i+1} = h(L, T_l, M, t_{i,1}, t_{i,2})$, where $t_{i,1}^T = Ar_{z,i}^T - d_i PK_i^T$, $t_{i,2}^T = Br_{z,i}^T - d_i T_l^T$.
      - calculate $r_{z,l} = v + d_l e_l$.
   d) Output $\sigma = (d_0, (r_{z,i}), T_l)$ as the signature.

3) **Signature verification**
   a) For $i = 0, 1, \ldots, u-1$, compute
$$t_{i,1}'^T = Ar_{z,i}^T - d_i PK_i^T,$$
$$t_{i,2}'^T = Br_{z,i}^T - d_i T_l^T,$$
$$d_{i+1}' = h(L, T_l, M, t_{i,1}', t_{i,2}').$$
   b) Accept if $d_0 = h(L, T_l, M, t_{u-1,1}', t_{u-1,2}') = d_u$, reject, otherwise.

4) **Signature linking**
For two valid signatures for different message σ and σ', check if $T_l = T_l'$. If equality is respected, then both messages are signed by one user.

C. Multilayer Linkable Ring Signature

The multilayer linkable ring signature is a modification of the linkable ring signature which is applied for a signing with multiple inputs.

The multilayer linkable ring signature scheme is described in [15], and its algorithm is presented below.

1) **Key generation**
   - Let the parameters $n, k, t \in \mathbb{N}$.
   - Choose matrices A, B from $\mathbb{F}_q^{(n-k)\times n}$, where $q = 2^m$, $m \in \mathbb{N}$.
   - For each user $\mathcal{P}_i$, $i \in \{0, 1, \ldots, u-1\}$, pick $e_{ij} \in \mathbb{F}_q^n$ for $j \in \{0, 1, \ldots, K-1\}$, where K is a number of signature layers, such that $w(e_{ij}) \le t$ and compute $PK_{ij}^T = Ae_{ij}^T$.
   - The public key is $PK_i = \{PK_{ij} | 0 \le j \le K-1\}$, the secret key is $e_i = \{e_{ij} | 0 \le j \le K-1\}$.

2) **Signature generation**
   a) For signer $\mathcal{P}_l$ calculate $T_{l,j}^T = Be_{l,j}^T$ for all $j \in \{0, 1, \ldots, K-1\}$.
   b) Choose $v_j \in \mathbb{F}_q^n$ such that $w(v_j) \le t$ and calculate $d_{l+1} = h(L, T_{l,j}, M, Av_0^T, Av_1^T, \ldots, Av_{K-1}^T, Bv_0^T, Bv_1^T, \ldots, Bv_{K-1}^T)$ where $L = \{PK_{ij}\}$, $h : \mathbb{F}_q^* \to \mathbb{F}_q$ is a hash function.
   c) For $i = l+1, l+2, \ldots, u-1, 0, 1, \ldots, l-1$:
      - choose $c_{ij} \in \mathbb{F}_q$,$r_{ij}^1, r_{ij}^2 \in \mathbb{F}_q^n$ such that $w(r_{ij}^h) \le t, (h = 1, 2)$
      - calculate $r_{i,j} = r_{ij}^1 + c_{ij} r_{ij}^2$, $i \ne l$.
      - compute $d_{i+1} = h(L, T_{l,j}, M, t_{i,j}^1, t_{i,j}^2)$, where $(t_{i,j}^1)^T = Ar_{i,j}^T - d_i PK_{ij}^T$, $(t_{i,j}^2)^T = Br_{i,j}^T - d_i T_{l,j}^T$.
   d) For $i = l$ calculate $r_{l,j} = v_j + d_l e_{l,j}$.
   e) Output $\sigma = (d_0, (r_{i,j}), T_{l,j})$ as the signature

3) **Signature verification**
   a) For $i = 0, 1, \ldots, u-1$, compute
$$(t_{i,j}'^{\,1})^T = Ar_{i,j}^T - d_i PK_{ij}^T,$$
$$(t_{i,j}'^{\,2})^T = Br_{i,j}^T - d_i T_{l,j}^T,$$
$$d_{i+1}' = h(L, T_{l,j}, m, t_{i,j}'^{\,1}, t_{i,j}'^{\,2}).$$

b) Check if $d'_0 = d_0$.

4) **Signature linking**
For two valid signatures for different message σ and σ', check if $T_{l,j} = T'_{l,j}$. If equality is respected, then both signatures are computed using the same private key.

D. Traceable Ring Signature

The traceable ring signature is a ring signature type which allows detecting if different messages are signed with the same secret key, and also identifying this signer.

The traceable ring signature scheme is described in [16], and its algorithm is presented below.

1) **Key generation**
- Let parameters $n, k, t \in \mathbb{N}, k = 3n/4$,
- Choose the code parity-check matrix H.
- Each user $\mathcal{P}_i$ chooses e_i from $\{0,1\}^n$, $w(e_i) = t$, and computes $s_i^T = He_i^T$, where $i = 0, 1, \ldots, u-1$.
- Public key of user $\mathcal{P}_i$ is (H, s_i) and secret key is e_i.

2) **Signature generation**
To sign message M, user $\mathcal{P}_i$:
a) computes $\tilde{H} = g(L)$ and $r_i^T = \tilde{H}e_i^T$, where $g : \mathbb{F}_2^* \to \mathbb{F}_2^{(n-k)\times n}$ is a random oracle hash function;
b) sets $A_0 = r_i + f(M) + \cdots + f^i(M)$, where $f : \mathbb{F}_2^* \to \mathbb{F}_2^{n-k}$ – a random oracle hash function, f^i – applying the function f i times on its input;
c) computes $r_j = A_0 + f(M) + f^2(M) + \cdots + f^j(M)$, for $j \neq i$;
d) applies the Fiat-Shamir transform to $\binom{u}{1}$-GStern's protocol, which is Stern's protocol modification described in [16], on input $(H, \bar{s}, \tilde{H}, \bar{r})$ where $\bar{s} = (s_1, \ldots, s_u)$ and $\bar{r} = (r_1, \ldots, r_u)$:
- computes the commitments Com according to $\binom{u}{1}$-GStern's protocol;
- simulates the verifier's challenge as $Ch = \bar{f}(Com, M)$, where $\bar{f}$ – a random oracle hash function;
- computes the corresponding responses $Resp$ according to $\binom{u}{1}$-GStern's protocol;
- outputs the transcript $T = (Com, Ch, Resp)$.

e) outputs the signature $\sigma = (A_0, Com, Resp)$.

3) **Signature verification**
To verify the message signature, the verifier:
a) computes $r_j = A_0 + f(M) + f^2(M) + \cdots + f^u(M)$;
b) computes $Ch = \bar{f}(Com, M)$;
c) verifies $T = (Com, Ch, Resp)$ is a valid transcript, according to $\binom{u}{1}$-GStern's protocol.

4) **Signature tracing**
Given two structures (L, M, σ) and (L', M', σ') where $\sigma = (A_0, Com, Resp)$ and $\sigma' = (A'_0, Com', Resp')$ such that $Ver(L, M, \sigma) = 1$ and $Ver(L, M', \sigma') = 1$, the verifier:
a) computes $r_j = A_0 + f(M) + f^2(M) + \cdots + f^j(M)$ and $r'_j = A'_0 + f(M') + f^2(M') + \cdots + f^j(M')$ for all j;
b) checks if $r_j = r'_j$. If this happens, it stores pk_j in a list $traceList$, which is initially empty for all j;
c) outputs the only $pk_i \in traceList$ if $|traceList| = 1$; else if $traceList = (pk_1, \ldots, pk_N)$ it outputs $linked$; else it outputs $indep$.

E. Threshold Ring Signature

The (r, u)-threshold ring signature is a ring signature type which requires at least r out of u users to sign a message.

The threshold signature scheme is described in [17], and its algorithm is presented below.

1) **Key generation**
- Let the parameters $n, k, t \in \mathbb{N}$.
- For each user $\mathcal{P}_i$ in the ring $R = \{\mathcal{P}_i | 1 \leq i \leq u\}$, choose a parity-check matrix H_i, a random binary $(n-k) \times (n-k)$ non-singular matrix Q_i and a random $n \times n$ permutation matrix P_i.
- Public key of user $\mathcal{P}_i$ $pk_i = \tilde{H}_i$, where $\tilde{H}_i = Q_i H_i P_i$, secret key $sk_i = (Q_i, H_i, P_i)$, The ring public key is $PK = (pk_{1\leq i\leq N})$.

2) **Signature generation**
a) Choose a leader signer $\mathcal{P}_l$ randomly from a signers' subset size r.
b) For each signer $\mathcal{P}_i, 1 \leq i \neq l \leq r$, randomly choose $e_{ij} \in \mathbb{F}_2^n, j = r+1, r+2, \ldots, u$, and compute $s_i^T = h(M)^T + \Sigma_{j=r+1}^u \tilde{H}_j e_{ij}^T$, where M is a message, $h : \mathbb{F}_2^* \to \mathbb{F}_2^{n-k}$ is a one-way collision-resistant hash function.
c) For each signer $\mathcal{P}_i, 1 \leq i \neq l \leq r$, compute $Q_i^{-1} s_i^T$. If $Q_i^{-1} s_i^T$ is a decodable syndrome, obtain a vector e'_i such that $H_i e_i'^T = Q_i^{-1} s_i^T$; otherwise, recompute s_i in the previous step.
d) Compute the permutations $e_i^T = P_i^T e_i'^T, 1 \leq i \neq l \leq r$.
e) For $r+1 \leq j \leq u$, choose a random e_{lj} under the condition $w(e_{lj} + \Sigma_{i=1, i\neq l}^r e_{ij}) = t$.
f) Set $e_j = \Sigma_{i=1}^r e_{ij}, r+1 \leq j \leq u$.
g) Compute s_l as $s_l^T = h(M)^T + \Sigma_{j=r+1}^u \tilde{H}_j e_{lj}^T$ if r is odd, or as $s_l^T = \Sigma_{j=r+1}^u \tilde{H}_j e_{lj}^T$, otherwise.
h) Compute $Q_l^{-1} s_l^T$. If $Q_l^{-1} s_l^T$ is a decodable syndrome, obtain a vector e'_l such that $H_l e_l'^T = Q_l^{-1} s_l^T$; otherwise, choose another e_{lj}.
i) Compute the permutation $e_l^T = P^T e_l'^T$.
j) Output $\sigma = (e_1, e_2, \ldots, e_u)$ as the signature.

3) **Signature verification**
If $w(e_i) = t$ for each e_i and $h(M)^T = \sum_{i=1}^u \tilde{H}_i e_i^T$, accept the signature, and reject, otherwise.

F. Signatures Comparison

The comparison of the ring signature types by the main parameters is presented in Table II.

In Table II the following notation is used:
- n – codeword length;
- k – code dimension;
- t – maximum correctable errors number;

TABLE II
CODE-BASED RING SIGNATURES COMPARISON

Parameters \ Signatures	Basic	Linkable	Multilayer Linkable	Traceable	Threshold
Public key size	$n(n-k)$	$n-k$	$K(n-k)$	$(n+1)(n-k)$	$n(n-k)$
Secret key size	$2n^2-kn+2k^2$	n	Kn	n	$2n^2-3nk+k^2$
Signature size	$n-k+ut\lceil\log_2 n\rceil$	$m(n(u+1)-k+1)$	$m(K(nu+n-k)+1)$	$n-k+uK(3t+2n)$	Nn

- m – $\mathbb{F}_{2^m}$ field expansion ratio;
- u – number of users.

Coefficient K in Table II has two different meanings: for the multilayer linkable signature, K is a number of signature layers; for the traceable signature, K is a number of GStern's algorithm rounds required to reduce the algorithm cheating probability.

G. Analytics

During the experiment, the multilayer linkable signature is modeled in SageMath 9.0 and generated with the Goppa code with the parameters n = 3488, m = 12, t = 64 which are recommended in [18].

To evaluate the generation and the validation time, the experiment is conducted with the number of users from $u = 5$ to 100 and the number of signature layers $K = 5$. The computings were performed on the machine with the characteristics given in Section II. The results of the experiment are presented in Figures 5, 6.

Comparing these results with the results in Section II, it can be concluded that execution time for the code-based multilayer signature is longer than the time for its elliptic curves analogue and makes the use of such signature options in real systems ineffective. However, despite this fact, the main advantage of the code-based signatures is their resistance to Shor's algorithm, and with the improvements in quantum computers development, makes these options prioritised and recommended for use.

IV. CONCLUSION

This paper provides an overview of modern cryptographic methods for creating a ring signature. A comparison of traditional and post-quantum algorithms based on error-correcting codes is presented. To demonstrate a significant difference in the implementation complexity of two approaches considered, the results of modeling the classical and post-quantum algorithms using the SageMath 9.0 software package are presented. However, it should be noted that since the algorithms have been modelled on personal computers with characteristics shown above, the calculations are limited by their computing power.

Based on the results obtained, the future research directions can be considered as the reduction of the complexity of code-based ring signature algorithms, as well as the size of signature keys and signatures.

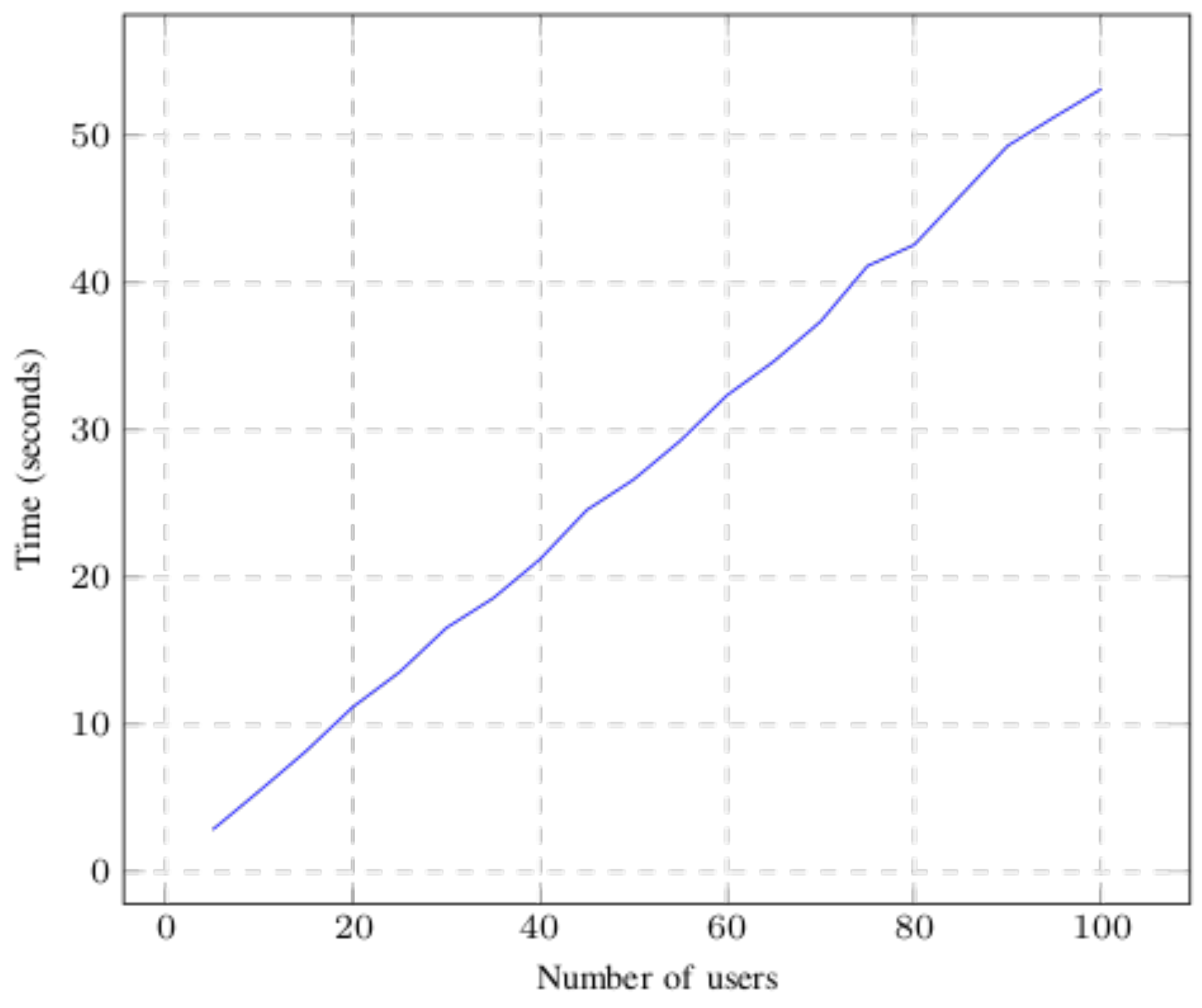

Fig. 5. Time dependence on the number of users in the signing algorithm in Multilayer Linkable signature.

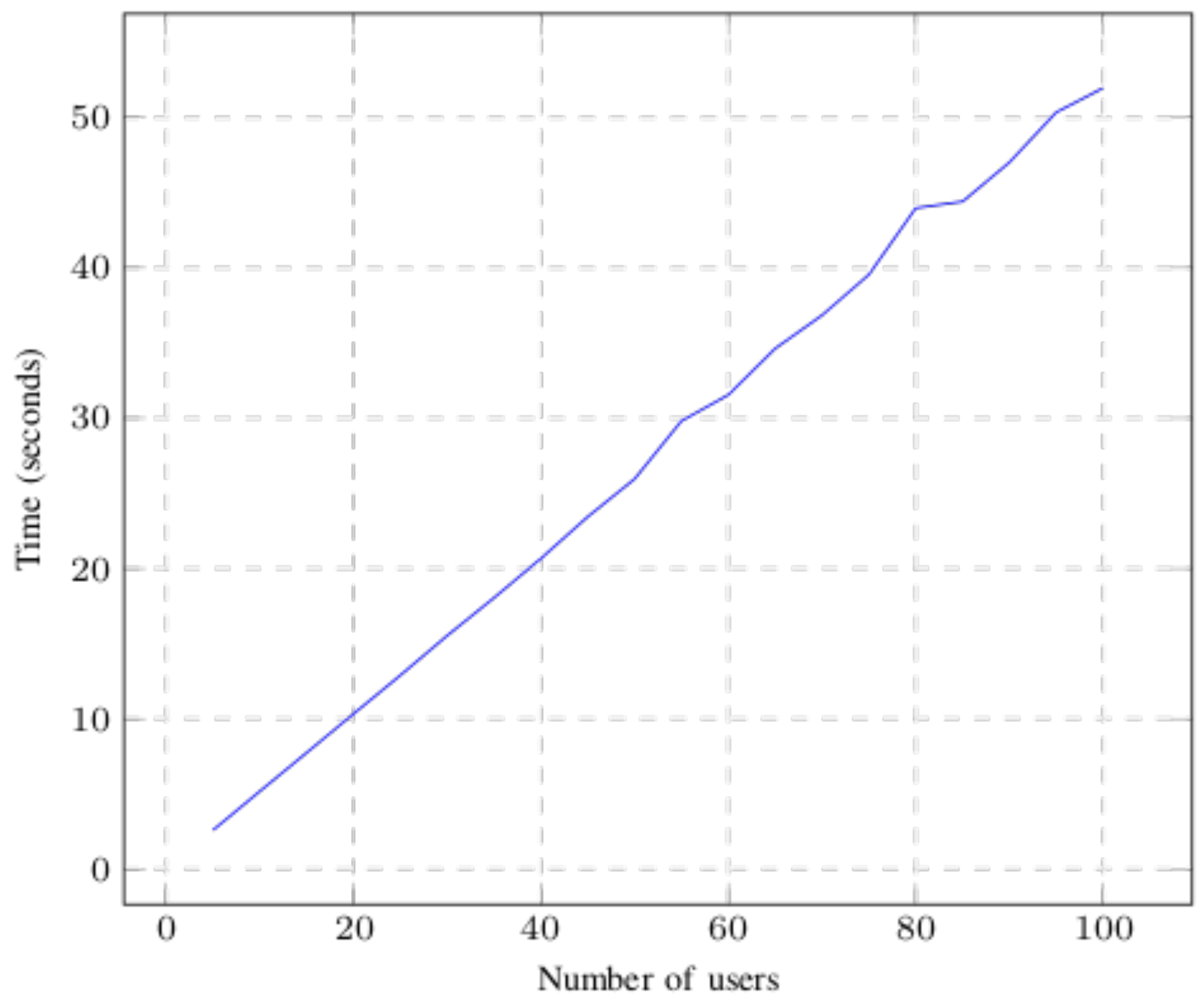

Fig. 6. Time dependence on the number of users in the verification algorithm in Multilayer Linkable signature.

REFERENCES

[1] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016, vol. 12.

[2] S. N. koe, K. M. Alonso, “Zero to monero: Second edition,” 2020, [Accessed 14-September-2020]. [Online]. Available: https://web.getmonero.org/library/Zero-to-Monero-2-0-0.pdf

[3] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, “Keccak sponge function family main document,” *Submission to NIST (Round 2)*, vol. 3, no. 30, pp. 320–337, 2009.

[4] D. J. Bernstein, “Curve25519: new diffie-hellman speed records,” in *International Workshop on Public Key Cryptography*. Springer, 2006, pp. 207–228.

[5] D. J. Bernstein, P. Birkner, M. Joye, T. Lange, and C. Peters, “Twisted edwards curves,” in *International Conference on Cryptology in Africa*. Springer, 2008, pp. 389–405.

[6] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, “High-speed high-security signatures,” *Journal of cryptographic engineering*, vol. 2, no. 2, pp. 77–89, 2012.

[7] D. J. Bernstein and T. Lange, “Faster addition and doubling on elliptic curves,” in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2007, pp. 29–50.

[8] N. Van Saberhagen, “Cryptonote v 2.0,” 2013, [Accessed 14-September-2020]. [Online]. Available: https://decred.org/research/saberhagen2013.pdf

[9] B. E. K. Seguias, “Monero’s building blocks part 6 of 10–linkable spontaneous anonymous group (lsag) signature scheme,” 2018, [Accessed 14-September-2020]. [Online]. Available: https://delfr.com/wp-content/uploads/2018/05/Monero_Building_Blocks_Part7.pdf

[10] S. Vijayakumaran, “Monero ring signatures,” 2018, [Accessed 14-September-2020]. [Online]. Available: https://www.ee.iitb.ac.in/ sarva/-courses/EE465/2018/slides/MoneroRingSignatures.pdf

[11] S. S.Noether, A. Mackenzie *et al.*, “Ring confidential transactions,” *Ledger*, vol. 1, pp. 1–18, 2016.

[12] S. Barg, “Some new np-complete coding problems,” *Problemy Peredachi Informatsii*, vol. 30, no. 3, pp. 23–28, 1994.

[13] N. T. Courtois, M. Finiasz, and N. Sendrier, “How to achieve a mceliece-based digital signature scheme,” in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2001, pp. 157–174.

[14] D. Zheng, X. Li, and K. Chen, “Code-based ring signature scheme.” *IJ Network Security*, vol. 5, no. 2, pp. 154–157, 2007.

[15] Y. Ren, Q. Zhao, H. Guan, and Z. Lin, “On design of single-layer and multilayer code-based linkable ring signatures,” *IEEE Access*, vol. 8, pp. 17 854–17 862, 2020.

[16] P. Branco and P. Mateus, “A traceable ring signature scheme based on coding theory,” in *International Conference on Post-Quantum Cryptography*. Springer, 2019, pp. 387–403.

[17] G. Zhou, P. Zeng, X. Yuan, S. Chen, and K.-K. R. Choo, “An efficient code-based threshold ring signature scheme with a leader-participant model,” *Security and Communication Networks*, vol. 2017, pp. 1 915 239:1–1 915 239:7.

[18] D. J. Bernstein, T. Chou, T. Lange, R. Misoczki, R. Niederhagen, E. Persichetti, P. Schwabe, J. Szefer, and W. Wang, “Classic mceliece: conservative code-based cryptography 30 march 2019,” 2019, [Accessed 14-September-2020]. [Online]. Available: https://classic.mceliece.org/nist/mceliece-20190331.pdf