**Гульнур Алпыскызы Тюлепбердинова** – кандидат физико-математических наук, ассоциированный профессор кафедры искусственного интеллекта и Big Data, Казахский национальный университет имени аль-Фараби, Казахстан; e-mail: tiulepberdynova.g@kaznu.kz. ORCID: https://orcid.org/0000-0002-4322-8983.

**Авторлар туралы мәліметтер**

**Олжас Аслзадаевич Умаров** – 3-курс студенті, Astana IT University, Қазақстан; e-mail: 230017@astanait.edu.kz. ORCID: https://orcid.org/0009-0001-8683-2216.

**Жанар Орымбаевна Оралбекова*** – PhD, қауымдастырылған профессор, Жасанды интеллект және деректер ғылымы мектебінің профессоры м.а., Astana IT University, Қазақстан; e-mail: zh.oralbekova@astanait.edu.kz. ORCID: https://orcid.org/0000-0002-3726-8882.

**Шырын Сарыбайевна Туткышбаева** – PhD, Бағдарламалық инженерия мектебінің ассистент профессоры, Astana IT University, Қазақстан; e-mail: sh.tutkyshbayeva@astanait.edu.kz. ORCID: https://orcid.org/0000-0002-1419-2571.

**Дина Орымбаевна Оралбекова** – PhD, қауымдастырылған профессор, Ақпараттық және есептеуіш технологиялар институтының аға ғылыми қызметкері, Қазақстан; e-mail: dinaoral@mail.ru, ORCID: https://orcid.org/0000-0003-4975-6493.

**Гульнур Алпыскызы Тюлепбердинова** – физика-математика ғылымдарының кандидаты, Жасанды интеллект және Big Data кафедрасының қауымдастырылған профессоры, Әл-Фараби атындағы Қазақ ұлттық университеті, Қазақстан; e-mail: tiulepberdynova.g@kaznu.kz. ORCID: https://orcid.org/0000-0002-4322-8983.

**Information about the authors**

**Olzhas Umarov** – third-year student, Astana IT University, Kazakhstan; e-mail: 230017@astanait.edu.kz. ORCID: https://orcid.org/0009-0001-8683-2216.

**Zhanar Oralbekova*** – PhD, associate professor, acting Professor of the School of AI and Data Science, Astana IT University, Kazakhstan; e-mail: zh.oralbekova@astanait.edu.kz. ORCID: https://orcid.org/0000-0002-3726-8882.

**Shyryn Tutkyshbayeva** – PhD, assistant professor of the School of Software Engineering, Astana IT University, Kazakhstan; e-mail: sh.tutkyshbayeva@astanait.edu.kz. ORCID: https://orcid.org/0000-0002-1419-2571.

**Dina Oralbekova** – PhD, associate professor, Senior Researcher of Institute of information and computational technologies, Kazakhstan; e-mail: dinaoral@mail.ru, ORCID: https://orcid.org/0000-0003-4975-6493.

**Gulnur Tyulepberdinova** – candidate of physical and mathematical sciences, associate professor of the Department of Artificial Intelligence and Big Data, Al-Farabi Kazakh National University, Kazakhstan; e-mail: tiulepberdynova.g@kaznu.kz. ORCID: https://orcid.org/0000-0002-4322-8983.

*Received 01.05.2026*
*Revised 23.06.2026*
*Accepted 24.06.2026*

https://doi.org/10.53360/2788-7995-2026-3(23)-7

IRSTI: 81.96.00

CC BY 4.0
Check for updates

**Ye. Seitkulov[*], R. Ospanov[1], B. Yergaliyeva[1], K. Utebayev[2]**
[1]L.N.Gumilyov Eurasian National University, 010008, Republic of Kazakhstan, Astana
[2]Almaty branch of the National Research Nuclear University MEPhI,
050040, Republic of Kazakhstan, Almaty
[*]e-mail: yerzhan.seitkulov@gmail.com

# IMPLEMENTATION AND COMPARATIVE ANALYSIS OF SPHINCS+-BASED DIGITAL SIGNATURE SCHEMES

***Annotation:*** *In post-quantum cryptography, the focus is on the design of algorithms capable of withstanding threats posed by quantum computers, which requires the adoption of new approaches to constructing their internal architecture. Significant attention in this field is devoted to digital signatures, as they provide authentication and integrity protection for data in digital environments. One of the most promising*

©Ye. Seitkulov, R. Ospanov, B. Yergaliyeva, K. Utebayev, 2026

*directions is considered to be hash-based digital signature schemes. They are characterized by high efficiency and proven cryptographic security, which makes them one of the key post-quantum solutions for electronic signatures. This paper presents a comparison of the main components, as well as the sizes of private keys, public keys, and signatures (in bytes), for several SPHINCS and SPHINCS+-based schemes, including the TANBA-SPHINCS+ scheme, a new variant of SPHINCS+. This scheme employs a cryptographic hash function based on a modified Sponge construction with multiple internal functions. The proposed approach extends the classical SPHINCS+ architecture and allows flexible configuration of internal transformations. Such a design increases the structural diversity of the algorithm and may contribute to improved resistance against certain types of cryptanalytic attacks. For experimental evaluation, a software implementation was developed. The paper also presents comparative experimental performance results.*

***Key words:*** *cryptographic algorithm, digital signature, cryptographic hash function, post-quantum cryptography, information security.*

## Introduction

Recently, we have witnessed the active development of quantum technologies. Modern research and development in quantum computing and quantum computers are leading to new challenges in information security. Post-quantum cryptography has become a strategic area of information security. The question of whether to transition to post-quantum cryptography is no longer relevant. The pressing question is how and when to do so. For the past two decades, the international community has been actively transitioning to post-quantum cryptography. Research in post-quantum cryptography is aimed not only at studying existing approaches to quantum-resistant cryptographic design, but also at developing new methods capable of ensuring strong security with reasonable computational overhead. At present, several major classes of post-quantum digital signature schemes can be identified, each relying on different mathematical assumptions and construction principles.

The first class includes hash-based digital signature schemes. The security of these schemes relies on the properties of cryptographic hash functions, particularly their resistance to preimage attacks, which are currently believed to remain secure even in the presence of quantum computers. When implemented correctly, the overall security of such schemes is determined primarily by the strength of the underlying hash function. Representative examples of this class include XMSS, LMS and SPHINCS+.

The second class comprises lattice-based signature schemes. Their security is connected with the hardness of lattice problems such as SVP, CVP, SIS, and LWE. These problems are widely considered computationally infeasible for both classical and quantum computing systems. Well-known examples of lattice-based signatures are CRYSTALS-Dilithium, FALCON, and Raccoon signature schemes.

The third category consists of code-based digital signature schemes. Their security relies on the difficulty of encoding and decoding error-correcting codes, including Goppa codes, linear codes, polar codes, and cyclic codes. Representative algorithms in this area include CROSS, Enhanced pqsigRM and LESS.

Another important class involves multivariate polynomial-based signature schemes. The security of these algorithms is based on the computational complexity of solving systems of multivariate polynomial equations over finite fields, which are generally considered NP-complete problems. Examples of such schemes are Rainbow, GeMSS, and DME-Sign.

A separate area of research is formed by isogeny-based digital signature schemes. These approaches employ isogenies between elliptic curves, while their security depends on the difficulty of constructing an isogeny between two given curves, a problem for which no efficient quantum algorithm is currently known. Examples of this category include SIKE, CSI-FiSh and SQIsign.

In addition to the approaches described above, there are also other post-quantum signature algorithms that do not directly belong to these categories, such as KAZ-SIGN, ALTEQ.

An additional advantage of hash-based schemes lies in their flexibility and adaptability. Such constructions can operate with various secure cryptographic hash functions, and if vulnerabilities are discovered in the currently used hashing algorithm, it can be replaced without fundamentally altering the overall structure of the signature scheme. Owing to these properties, hash-based digital signatures are regarded as one of the most practical and reliable directions in the development of post-quantum cryptography.

SPHINCS+ is a framework for constructing a family of digital signature algorithms [1], [2]. Initially, SPHINCS+ considered implementations using the SHA2, SHAKE, and Haraka algorithms, while the version of the framework outlined in the FIPS 205 standard, i.e., SLH-DSA, retained the SHA2 and SHAKE implementations. There are several implementation examples of the SPHINCS+ scheme using various cryptographic hash functions. For example, the paper [3] proposes a variant of the SPHINCS+ scheme that uses the Stribog hash function, described in the Russian standard GOST R 34.11-2018 [4]. K-SPHINCS+ [5], [6] is another variant of SPHINCS+ that uses the Korean cryptographic algorithms LSH, CHAM, and LEA.

Previously, a new modification of the Sponge scheme, a modern scheme for constructing cryptographic hash functions, was developed in [7]. The Sponge construction used for hash computation consists of several consecutive stages. First, the input message undergoes a padding procedure in which additional bits are appended so that the resulting message length becomes divisible by the predefined block size. Afterward, the internal state of the algorithm, represented by a fixed-length bit sequence, is initialized with a predefined value. The message is then processed during the absorbing phase. At this stage, message blocks of fixed size are sequentially combined with part of the internal state using a bitwise operation, followed by the application of an internal permutation function. This procedure is repeated until all message blocks have been processed. Once the absorption phase is completed, the algorithm proceeds to the squeezing phase, where fragments of the final hash value are extracted from the internal state. If the required output length exceeds the available number of bits, the permutation function is applied repeatedly to generate additional output fragments. The central component of the scheme is the internal function, which is either a permutation or a transformation of some fixed length.

Based on the modified Sponge construction, a new cryptographic hash function called TANBA was developed. In this approach, the internal function applied at each stage is determined by a dedicated selection mechanism that chooses a function from a predefined set of internal functions. The proposed implementation incorporates three different internal functions. The first function follows a generalized design approach derived from AES principles. The second is built using the word ring feedback with carry shift registers. The third function is conceptually close to the Keccak permutation; however, several structural modifications distinguish it from the original Keccak design. One of the advantages of the proposed construction is its scalability. By modifying the selection mechanism, the scheme can be adapted to operate with a different number of internal functions, including configurations with two or four functions. The construction can also be implemented using only a single internal function when required.

In [8] a new variant of SPHINCS+, replacing the hash functions from SPHINCS+ with the TANBA hash function, was proposed. The proposed TANBA-SPHINCS+ scheme uses the architectural principles of SPHINCS+, but differs in the use of an alternative hash function TANBA. Therefore, the design under consideration represents a research variant of the SPHINCS+ scheme, intended for analyzing possible modifications to the underlying cryptographic primitives.

The contribution of this paper includes the implementation of the TANBA-SPHINCS+ scheme in Python and its experimental evaluation against selected SPHINCS+ variants. Additionally, the study analyzes the impact of replacing the hash function while preserving the overall structure of the scheme. The aim of this work is to study the implementation of the previously proposed TANBA-SPHINCS+ scheme and to evaluate the effect of replacing the underlying hash function within the SPHINCS+ framework. The focus is placed on implementation aspects and experimental analysis rather than on developing a new cryptographic construction. This paper provides a comparison of main components (one-time signatures, few-time signatures, cryptographic hash functions), sizes of private keys, public keys, and signatures (in bytes) for some variants of SPHINCS and SPHINCS+-based schemes, including the TANBA-SPHINCS+ scheme. To conduct the computational experiments, a software implementation was written in Python 3.8.10. The paper presents comparative experimental results in terms of performance.

**Materials and methods**

The bases for the development of hash-based digital signatures are the so-called one-time signatures. They appeared back in the late 1970s. The simplest one-time signature scheme can be described as follows. For each bit of the message, pairs of random secret values are used. Their hashes are published. When signing a message, some of the secret values are revealed. Signature verification is performed through re-hashing. More formally, this can be described as follows.

Let $M$ be an $n$-bit message to be signed: $M = a_1 a_2 \dots a_n$, $a_i \in \{0,1\}$. Also suppose that there is some cryptographic hash function $H$ that computes hash values of length $n$.

First, a key pair is calculated: a private key $K_{priv}$ and a public key $K_{pub}$. The private key is obtained using a cryptographically secure pseudorandom number generator. For this, $n$ pairs of pseudorandom $n$-bit numbers are generated. Let us denote these pairs as $(x_i, y_i)$, $i = 1, \dots, n$. These pairs of numbers represent the private key $K_{priv}$. Then, using the hash function $H$, the hash values for all these numbers are calculated. Let us denote these pairs as $(H(x_i), H(y_i))$, $i = 1, \dots, n$. These pairs of numbers represent the public key $K_{pub}$.

Next, a signature $Sig(M) = s_1 s_2 \dots s_n$ is generated using the private key $K_{priv}$: $s_i = \begin{cases} x_i, \ if \ a_i = 0 \\ y_i, \ if \ a_i = 1 \end{cases}$. Thus, these $n$ $n$-bit numbers represent the digital signature of message $M$. Further verification of the signature is also performed using a hash function $H$. To do this, the hash values of the $n$ numbers that make up the signature $Sig(M) = s_1 s_2 \dots s_n$ are first calculated: $H(s_1), H(s_2), \dots, H(s_n)$. The resulting hash values are then compared with the corresponding numbers that make up the public key: for each bit $a_i$ of message $M = a_1 a_2 \dots a_n$, if $a_i = 0$, the equality $H(s_i) = H(x_i)$ is checked; if $a_i = 1$, the equality $H(s_i) = H(y_i)$ is checked. If all equalities are true, then the signature verification is considered successful.

This is the first known digital signature scheme (Lamport's scheme) that demonstrated the possibility of constructing secure signatures not based on factorization or discrete logarithm, but solely on the properties of cryptographic hash functions. Furthermore, this scheme can be considered the foundation of post-quantum cryptography. It is resistant to attacks on quantum computers. Its security comes down to the security of the cryptographic hash function used in the scheme. The name "one-time signature (OTS)" follows from the fact that such a signature is secure only when forming a signature for one message. This is one of the disadvantages of this scheme. Another drawback is the large size of the keys and signatures. For example, for a 256-bit message, the private key consists of 512 random numbers, the public key also consists of 512 256-bit hash values, and the signature consists of 256 256-bit values. Clearly, as the message size increases, this becomes impractical. Using such a signature, one cannot simply sign multiple messages. Later, more sophisticated and efficient one-time signatures were developed (for example, W-OTS and W-WOTS+: the main improvement made in these schemes over Lamport's scheme is that the bits are signed in groups, i.e. one signature element is created for a group of bits.). And also, it was from one-time signatures that few-time signatures (FTS), Merkle schemes, and then modern hash-based signatures emerged and developed.

Modern hash-based digital signature schemes can be divided into stateful and stateless schemes. Stateful signature schemes store information about used keys. This means there is a set of one-time keys. Each key can only be used once. Each time a signature is performed, a counter is incremented, and the key used is marked. Stateless signature schemes do not store any state at all. Each signature can be used independently, and no memory of previous signatures is required. In other words, in stateful schemes, the use of each key is controlled. In stateless schemes, the system is designed in such a way that even reuse does not compromise security.

Examples of a stateful scheme are XMSS and LMS, standardized by NIST and IETF. The first stateless hash-based signature scheme was proposed in [9]. However, to ensure sufficient security and a reasonable number of signatures, parameters resulting in a fairly large signature size were required, making this scheme impractical. The first practical stateless hash-based signature scheme, capable of achieving reasonable signature sizes while computing hundreds of signatures per second on modern processors, was the SPHINCS scheme [10]. This scheme combined the one-time signature (OTS) binary authentication tree from [9, x6.4.2] and the HORS few-time signature scheme with two key optimizations. First, the internal tree nodes are not one-time signatures, but Merkle trees whose leaves are W-OTS signatures. This allows each node to sign up to $2^x$ child nodes, where $x$ is the height of the corresponding tree. This leads to the use of a hypertree and a reduction in signature size due to the inclusion of fewer OTSs, but increases the signature time, since all Merkle trees on the path to a leaf must be generated anew for each signature. Second, the hypertree leaves are not one-time signatures, but few-time HORST signatures – a variant of the HORS scheme with a binary tree for public key compression, which allows a single leaf to sign

multiple messages, improves resilience to path collisions, and thereby reduces the required hypertree height. A specific instantiation of the scheme, called SPHINCS-256, was also proposed. The name of the instantiation is determined by the security parameter n = 256 (bitlength of hashes in HORST and W-OTS). Hash functions used in SPHINCS-256 are BLAKE256, BLAKE512 and function based on ChaCha12.

The SPHINCS framework subsequently became the basis for subsequent work conducted within the NIST Post Quantum Cryptography standardization process.

A new variant of the SPHINCS signature scheme, called SPHINCS-Simpira, was presented in [11]. The scheme is based on the Simpira family of cryptographic permutations, presented in [12].

Several instantiations of the SPHINCS scheme with various cryptographic hash functions on modern high-performance processors was examined in [13]. Algorithms used included SHA256, Keccak, Simpira, Haraka, and ChaCha.

In [14], a new variant of SPHINCS, called Gravity-SPHINCS, was proposed with shorter keys, shorter signatures, and faster signing and verification procedures while maintaining the same level of security as SPHINCS. These improvements are based on a number of optimizations proposed in the same paper. The paper presents a new, more secure variant of the HORS few-time signature scheme, called PORS. It also introduces secret key caching to speed up signing and reduce signature size; batch signatures to amortize signature time and reduce signature size; mask-free constructions to reduce key size; and a method for eliminating redundancy in authentication paths in Merkle trees, called Octopus. Hash functions used in SPHINCS-256 are Haraka-v2-256, Haraka-v2-512, SHA-256 and function based on AES-256-CTR.

In [1] a new refinement and generalization of the SPHINCS, called SPHINCS+, was presented. One of several improvements proposed in SPHINCS+ is the use of a new FORS few-time signature scheme. Another distinctive feature is the use of publicly verifiable index selection as a method for selecting leaf nodes. These changes ultimately make it more difficult to attack SPHINCS+ using a few-time signature scheme, thereby allowing the selection of smaller parameters. Also so-called tweakable hash functions introduced in [1]. SPHINCS+ is a scheme for constructing a family of digital signature algorithms. Initially, SPHINCS+ considered implementation options using the SHA2, SHAKE, and Haraka algorithms, while the version of the scheme described in the FIPS 205 standard, i.e., SLH-DSA, retained the SHA2 and SHAKE options.

The SPHINCS+ digital signature scheme is a combination of the following algorithms: a key pair generation algorithm, a digital signature generation algorithm, and a digital signature verification algorithm [1], [2].

Let $M = m_1 m_2 \ldots m_k$, $m_i \in \{0,1\}$ is a $k$-bit message.

The key pair generation algorithm (generation of secret and public keys) is defined as follows.

Let $G$ be a cryptographically strong pseudorandom number generator. Using generator $G$, we obtain $n$-bit $sk_{seed}$, $pk_{seed}$ and $sk_{prf}$. Then the private key will consist of $sk_{seed}$, $sk_{prf}$, $pk_{seed}$ and $pk_{root}$ ($sk = sk_{seed} \parallel sk_{prf} \parallel pk_{seed} \parallel pk_{root}$ ), and the public key will consist of $pk_{seed}$ and $pk_{root}$ ($pk = pk_{seed} \parallel pk_{root}$). Here, $pk_{root}$ is the hypertree's public key, i.e., the public key (root node) of the single XMSS tree on the top layer.

The digital signature generation algorithm is defined as follows. The input data is a message $M$ and a secret key $sk = sk_{seed} \parallel sk_{prf} \parallel pk_{seed} \parallel pk_{root}$.

First, an $n$ -bit randomization string $R = PRF_{msg}(sk_{prf}, opt, M)$ is generated, where $PRF_{msg}: \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ is a pseudorandom function, and $opt$ is some additional random $n$ -bit value. Then, using $R$ , $dg = H_{msg}(R, pk_{seed}, pk_{root}, M)$ is calculated, where $H_{msg}: \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^m$ is the keyed hash function. Next, using the calculated value of $dg$, $md_{temp}$ is determined as the first $\lfloor (k \log_2 t + 7)/8 \rfloor$ bytes of $dg$, $ti_{temp}$ is the next $\lfloor (\mathrm{h} - \mathrm{h/d} + 7)/8 \rfloor$ bytes of $dg$ , $li_{temp}$ is the next $\lfloor (\mathrm{h/d} + 7)/8 \rfloor$ bytes of $dg$ . Next, $md$ is determined as the first $k \log_2 t$ bits of $md_{temp}$, $ti$ is the first $\mathrm{h} - \mathrm{h/d}$ bits of $ti_{temp}$, $li$ is the first $\mathrm{h/d}$ bits of $li_{temp}$. Using the obtained values, $\sigma_{FORS}$ (the FORS signature) and $\sigma_{HT}$ (the hypertree signature) are sequentially determined. The signature will then consist of the obtained $\sigma_{FORS}$ and $\sigma_{HT}$ ($\sigma = R \parallel \sigma_{FORS} \parallel \sigma_{HT}$).

The digital signature verification algorithm is defined as follows.

The input data are the message $M$, the signature $\sigma = R \| \sigma_{FORS} \| \sigma_{HT}$, and the public key $pk = pk_{seed} \| pk_{root}$.

First, $dg = H_{msg}(R, pk_{seed}, pk_{root}, M)$ is calculated, $md_{temp}$ is determined as the first $\lfloor (k \log_2 t + 7)/8 \rfloor$ bytes of $dg$, $ti_{temp}$ is the next $\lfloor (\mathrm{h} - \mathrm{h/d} + 7)/8 \rfloor$ bytes of $dg$, $li_{temp}$ is the next $\lfloor (\mathrm{h/d} + 7)/8 \rfloor$ bytes of $dg$, then $md$ is determined as the first $k \log_2 t$ bits of $md_{temp}$, $ti$ is the first $\mathrm{h} - \mathrm{h/d}$ bits of $ti_{temp}$, $li$ is the first $\mathrm{h/d}$ bits of $li_{temp}$. Next, the FORS public key candidate is calculated and the hypertree signature on this public key is verified.

The paper [3] considers the implementation of SPHINCS+ using the STREEBOG cryptographic hash function, described in the Russian standard GOST R 34.11-2018 [4]. The papers [5], [6] propose a hashing-based digital signature scheme, K-SPHINCS+, in which the hash functions from SPHINCS+ are replaced with the Korean cryptographic algorithms LSH, CHAM, and LEA.

In [15], an improvement to the SPHINCS+ scheme, called SPHINCS-α, was proposed. The proposed optimization is primarily based on optimizations to the one-time signature scheme. The optimized one-time signature scheme, CS-W-OTS+ (constant-sum W-OTS+), increases the message-to-signature-size ratio compared to the original W-OTS+ scheme, thereby reducing the one-time signature size. The W-OTS+ component in SPHINCS+ and XMSS is replaced by CS-W-OTS+. As a result, the parameters of the hash-based signature scheme can be retuned to reduce the signature size without increasing computation time.

Reference [16] introduces the stateless hash-based signature scheme SPHINCS+C, based on SPHINCS+. It relies on updated one-time and few-time signature primitives, W-OTS+C and FORS+C, respectively, both also introduced in [16]. W-OTS+C reduces the number of chains in the original W-OTS construction, which leads to smaller signatures and faster verification, while leaving signing and key generation costs essentially unchanged. FORS+C further decreases the signature size and provides a slight improvement in verification efficiency, again without affecting the main computational costs of signing and key generation.

A related scheme, Hypericum, proposed for standardization in Russia, is presented in [17], [18]. It is based on SPHINCS+ and SPHINCS+C and employs W-OTS+C and FORS+C as core components. In addition, it uses the STREEBOG hash function defined in the Russian standard GOST R 34.11-2018, with a 256-bit output.

In [19], a variant of SPHINCS+ called F-SPHINCS+ is introduced. It was designed as a basis for constructing a group signature scheme using only symmetric primitives. The scheme incorporates a modified version of FORS, referred to as M-FORS (Merkle FORS). Its overall structure is organized as a hypertree built from M-FORS trees.

In [20], a new few-time signature scheme, PORS+FP, is used within SPHINCS+. It is an optimization of the PORS few-time signature scheme using the forced pruning technique (PORS plus forced pruning). The resulting improvements led to shorter signatures and faster verification with minimal increase in signing time.

In [21], new stateless hashing-based digital signature schemes (CEDRUS+, CEDRUS-α, CEDRUS+C) were developed using the SPHINCS+ framework. The authors extended the hypertree structure underlying the SPHINCS+ framework, allowing trees of different heights to appear at different levels. By allowing for different hash chain lengths, they generalized one-time signatures based on hash chains and connected them to the hypertree. The resulting structural generalizations led to expanded design capabilities and a more nuanced analysis of tradeoffs between various signature scheme parameters. New schemes were identified that outperform several modern stateless hashing-based signature schemes (SPHINCS+, SPHINCS-α, and SPHINCS+C) in signature size, signature efficiency, and verification.

**Results and discussion**

The TANBA-SPHINCS+ algorithm proposed in [8] is an implementation of the SPHINCS+ scheme, replacing the hash functions from SPHINCS+ with the TANBA hash function. TANBA-SPHINCS+ uses the same parameters and structures as SPHINCS+. Specifically, the algorithm uses the following functions.

$PRF: \{0,1\}^n \times \{0,1\}^{256} \to \{0,1\}^n$ is a pseudorandom function for generating pseudorandom keys, defined as $PRF(seed, adrs) = TANBA(seed||adrs, 8n)$.

$PRF_{msg}: \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^n$ is a pseudorandom function for generating random numbers for message compression, defined as

$PRF_{msg}(sk_{prf}, opt, M) = TANBA(sk_{prf} ||opt||M, 8n)$.

$H_{msg}: \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^m$ is an additional keyed hash function that can handle messages of arbitrary length, defined as

$H_{msg}(R, pk_{seed}, pk_{root}, M) = TANBA(R||pk_{seed}||pk_{root}||M, 8m)$.

$T_l: \{0,1\}^n \times \{0,1\}^{256} \times \{0,1\}^{ln} \rightarrow \{0,1\}^n$ , $F = T_1: \{0,1\}^n \times \{0,1\}^{256} \times \{0,1\}^n \rightarrow \{0,1\}^n$ , $H = T_2: \{0,1\}^n \times \{0,1\}^{256} \times \{0,1\}^{2n} \rightarrow \{0,1\}^n$ are tweakable hash functions, defined as

$F(pk_{seed}, adrs, M_1) = TANBA(pk_{seed}||adrs||M_1, 8n)$,

$H(pk_{seed}, adrs, M_1||M_2) = TANBA(pk_{seed}||adrs||M_1||M_2, 8n)$,

$T_l(pk_{seed}, adrs, M) = TANBA(pk_{seed}||adrs||M, 8n)$, respectively. Here, $adrs$ is the address of one of the seven types defined in [2].

The following Table 1 below summarizes the main components (one-time signatures, few-time signatures, cryptographic hash functions) of the above-mentioned algorithms and the TANBA-SPHINCS+. Table 2 provides a comparison of the sizes of the private keys, public keys, and signatures (in bytes) for some variants of SPHINCS and SPHINCS+-based schemes, including the TANBA-SPHINCS+.

Table 1 – Main components of SPHINCS and SPHINCS+-based schemes

| Scheme | OTS | FTS | Hash functions |
|---|---|---|---|
| SPHINCS (2015) [10] | W-OTS+ | HORST | BLAKE256, BLAKE512, based on ChaCha12 |
| SPHINCS-Simpira (2017) [11] | W-OTS+ | HORST | based on Simpira |
| Instantiation of SPHINCS (2017) [13] | W-OTS+ | HORST | SHA256, Keccak, Haraka, based on Simpira, based on ChaCha |
| Gravity-SPHINCS (2018) [14] | W-OTS | PORST | Haraka-v2-256, Haraka-v2-512, SHA-256, based on AES-256-CTR |
| SPHINCS+ (2019) [1] | W-OTS+ | FORS | SHA2, SHAKE, Haraka |
| STREEBOG-SPHINCS+ (2020) [3] | W-OTS+ | FORS | STREEBOG |
| K-SPHINCS (2022) [5], [6] | W-OTS+ | FORS | LSH256, based on CHAM256, based on LEA256 |
| SPHINCS-α (2022) [15] | CS-W-OTS+ | FORS | SHA2-256, SHA2-512, SHAKE-256 |
| SPHINCS+C (2023) [16] | W-OTS+C | FORS+C | SHAKE |
| Hypericum (2023) [17],[18] | W-OTS+C | FORS+C | STREEBOG |
| F-SPHINCS (2024) [19] | W-OTS+ | M-FORS | Based on LowMC |
| SPHINCS+PORS+FP (2025) [20] | W-OTS+ | PORS-FP | SHA-3, SHAKE |
| CEDRUS+ (2025) [21] | Inhomogeneous W-OTS+ | FORS | SHAKE-128, SHAKE-192, SHAKE-256 |
| CEDRUS-α (2025) [21] | Inhomogeneous CS-W-OTS+ | FORS | SHAKE-128, SHAKE-192, SHAKE-256 |
| CEDRUS+C (2025) [21] | Inhomogeneous W-OTS+C | FORS+C | SHAKE-128, SHAKE-192, SHAKE-256 |
| TANBA-SPHINCS+ [8] | W-OTS+ | FORS | TANBA |

Table 2 – Private key, public key and signature sizes (in bytes) of some instantiations for SPHINCS and SPHINCS+-based schemes

| Scheme | Private key size | Public key size | Signature size |
|---|---|---|---|
| *1* | *2* | *3* | *4* |
| SPHINCS-256 | 1088 | 1056 | 41000 |
| SPHINCS-Simpira | 1088 | 1056 | 41000 |
| Gravity-SPHINCS (NIST-fast) | 64 | 32 | 35168 |
| Gravity-SPHINCS (NIST) | 64 | 32 | 26592 |
| Gravity-SPHINCS (NIST-slow) | 64 | 32 | 22304 |
| Gravity-SPHINCS (fast) | 64 | 32 | 28928 |
| Gravity-SPHINCS (batched) | 64 | 32 | 20032 |
| Gravity-SPHINCS (small) | 64 | 32 | 12640 |
| SPHINCS+-128s | 64 | 32 | 7 856 |

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| SPHINCS+-128f | 64 | 32 | 17 088 |
| SPHINCS+-192s | 96 | 48 | 16 224 |
| SPHINCS+-192f | 96 | 48 | 35 664 |
| SPHINCS+-256s | 128 | 64 | 29 792 |
| SPHINCS+-256f | 128 | 64 | 49 856 |
| K-SPHINCS | 128 | 64 | 49 856 |
| SPHINCS-α-128s | 64 | 32 | 6880 |
| SPHINCS-α-128f | 64 | 32 | 16720 |
| SPHINCS-α-192s | 96 | 48 | 14568 |
| SPHINCS-α-192f | 96 | 48 | 34896 |
| SPHINCS-α-256s | 128 | 64 | 27232 |
| SPHINCS-α-256f | 128 | 64 | 49312 |
| SPHINCS+C-128s | 64 | 32 | 6304 |
| SPHINCS+C-128f | 64 | 32 | 14904 |
| SPHINCS+C-192s | 96 | 48 | 13776 |
| SPHINCS+C-192f | 96 | 48 | 33016 |
| SPHINCS+C-256s | 128 | 64 | 26096 |
| SPHINCS+C-256f | 128 | 64 | 46884 |
| Hypericum-s | 128 | 64 | 18292 |
| Hypericum-f | 128 | 64 | 58460 |
| Hypericum-u | 128 | 64 | 28068 |
| SPHINCS+PORS+FP-128s1 | 64 | 32 | 7332 |
| SPHINCS+PORS+FP-128f1 | 64 | 32 | 16420 |
| SPHINCS+PORS+FP-192s1 | 96 | 48 | 15316 |
| SPHINCS+PORS+FP-192f1 | 96 | 48 | 34540 |
| SPHINCS+PORS+FP-256s1 | 128 | 64 | 28260 |
| SPHINCS+PORS+FP-256f1 | 128 | 64 | 48132 |
| SPHINCS+PORS+FP-128s2 | 64 | 32 | 6096 |
| SPHINCS+PORS+FP-128f2 | 64 | 32 | 14504 |
| SPHINCS+PORS+FP-192s2 | 96 | 48 | 13104 |
| SPHINCS+PORS+FP-192f2 | 96 | 48 | 32104 |
| SPHINCS+PORS+FP-256s2 | 128 | 64 | 25,168 |
| SPHINCS+PORS+FP-256f2 | 128 | 64 | 45412 |
| CEDRUS+-128s | 64 | 32 | 7840 |
| CEDRUS+-128f | 64 | 32 | 16528 |
| CEDRUS+-192s | 96 | 48 | 16176 |
| CEDRUS+-192f | 96 | 48 | 35280 |
| CEDRUS+-256s | 128 | 64 | 29600 |
| CEDRUS+-256f | 128 | 64 | 49632 |
| CEDRUS-α-128s | 64 | 32 | 6960 |
| CEDRUS-α-128f | 64 | 32 | 16560 |
| CEDRUS-α-192s | 96 | 48 | 14760 |
| CEDRUS-α-192f | 96 | 48 | 35640 |
| CEDRUS-α-256s | 128 | 64 | 27104 |
| CEDRUS-α-256f | 128 | 64 | 49696 |
| CEDRUS+C-128s | 64 | 32 | 6252 |
| CEDRUS+C-128f | 64 | 32 | 14452 |
| CEDRUS+C-192s | 96 | 48 | 13384 |
| CEDRUS+C-192f | 96 | 48 | 31620 |
| CEDRUS+C-256s | 128 | 64 | 25228 |
| CEDRUS+C-256f | 128 | 64 | 46500 |
| TANBA-SPHINCS+ | 128 | 64 | 49 856 |

Existing implementations of the SPHINCS+ scheme use a limited set of cryptographic hash functions, such as SHA2, SHA3, SHAKE, and apply the same function to various operations (generating pseudorandom functions, computing Merkle tree nodes, hashing messages). This limits the flexibility of the scheme, complicates the replacement of individual components, and limits the

experimental exploration of alternative hash function configurations. The modified scheme proposed in this paper, in which standard hash functions are replaced with a hash function based on the modified Sponge scheme with multiple internal functions, has the potential to provide a more flexible and modular architecture. Despite existing implementations, there is virtually no research in the literature on the use of Sponge-constructions with multiple internal functions for building a flexible and modular scheme.

The proposed TANBA-SPHINCS+ scheme uses the same security parameters as the SPHINCS+. Therefore, the public key, private key, and signature sizes remain identical to those of the original scheme. This is because the modification only affects the hash function used, while the hypertree parameters and WOTS+ structure remain unchanged.

To conduct the computational experiments, a software implementation was written in Python 3.8.10. The experiments were conducted on a computer with the following specifications: processor – Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz, RAM – 4.00 GB, video adapter – NVIDIA GeForce MX330 (2 GB), Intel(R) UHD Graphics (128 MB), system type – 64-bit operating system, x64 processor, operating system – Windows 10 Pro (version 22H2).

Following Table 3 presents comparative experimental results in terms of performance.

Table 3 – Experimental results

| Algorithm | Keygen | Sign | Verify |
|---|---|---|---|
| SPHINCS+-SHA2-256f | 12.9 ms | 257 ms | 2860 ms |
| SPHINCS+-SHAKE-256f | 13.4 ms | 261 ms | 2921 ms |
| TANBA-SPHINCS+ | 15.2 ms | 283 ms | 3012 ms |

The experimental results show that TANBA-SPHINCS+ achieves performance comparable to the reference SPHINCS+ implementations while preserving identical key and signature sizes. The observed differences in execution time are mainly due to the computational cost of the underlying hash function. This work focuses on the implementation and experimental evaluation of a previously proposed scheme rather than on improving performance or designing a new cryptographic construction. In this context, the results confirm that modifying the hash function does not affect the overall structure of the SPHINCS+ scheme, but also does not lead to performance gains. The computational cost of TANBA-SPHINCS+ is dominated by hash function evaluations within the SPHINCS+ framework. Key generation depends on the hypertree structure, while signing involves multiple operations in the WOTS+ and FORS components. Verification follows a similar structure but generally requires fewer computations. Performance is measured in milliseconds, which depends on the hardware and software environment. Therefore, the reported values should be treated as approximate. More hardware-independent metrics, such as cycles per byte, were not used and remain a subject for future work. The reported execution times are based on a single measurement for each operation. Consequently, statistical measures such as standard deviation and confidence intervals were not evaluated. This limitation will be addressed in future work through repeated experiments and statistical analysis. The current implementation is limited to a single parameter set corresponding to SPHINCS+-256f. Although other configurations were not evaluated, the scheme can be extended to different parameter sets without changes to its overall structure.

**Conclusion**

This paper presents an implementation of the TANBA-SPHINCS+ digital signature scheme, which was previously introduced in our earlier work, together with its experimental evaluation against selected SPHINCS+ variants. The implementation was developed in Python 3.8.10 and tested in a standard desktop environment. The experimental study covered the main stages of the algorithm, including key generation, signing, and verification. The obtained results show that TANBA-SPHINCS+ preserves the same key and signature sizes as the SPHINCS+-256f configuration. In terms of execution time, the scheme demonstrates performance comparable to standard SPHINCS+ implementations, with no noticeable improvement in computational speed. These results indicate that the substitution of the underlying hash function does not affect the overall structure of the SPHINCS+ framework. At the same time, it also does not lead to a reduction in computational cost, since performance remains dominated by hash-based operations. The main contribution of this work lies in the practical implementation of the TANBA-SPHINCS+ scheme and the experimental verification of its compatibility with the original SPHINCS+ design. The study confirms that the proposed modification can be integrated without changing the structural properties of the scheme.

The work is limited to a single parameter set and a Python-based implementation. Hardware-specific optimizations and alternative parameter configurations were not considered, and the reported timing results should therefore be treated as indicative. Future research should focus on a formal security analysis of the TANBA-based construction, a more detailed study of computational complexity, and an evaluation under different parameter settings.

## References


1. The SPHINCS+ Signature Framework. In Proceedings of the 2019 ACM SIGSAC Conference / D.J. Bernstein et al. – New York, NY, USA, P. 2129-2146. https://doi.org/10.1145/3319535.3363229.
2. Aumasson, et al. SPHINCS+ - Submission to the NIST post-quantum project, v.3.1. 10 June 2022. https://sphincs.org/data/sphincs+-r3.1-specification.pdf.
3. Sphincs+ post-quantum digital signature scheme with Streebog hash function / E. Kiktenko et al // AIP Conference Proceedings. – 2020. – vol. 2241. – P. 020014. https://doi.org/10.1063/5.0011441.
4. GOST 34.11-2018. Mezhgosudarstvennyi standart. Informatsionnaya tekhnologiya. Kriptograficheskaya zashchita informatsii. Funktsiya kheshirovaniya. (In Russian).
5. K-XMSS and K-SPHINCS+: Hash based signatures with korean cryptography algorithms / M. Sim et al // Cryptology ePrint Archive, 2022. https://eprint.iacr.org/2022/152.pdf.
6. K-XMSS and K-SPHINCS+: Enhancing Security in Next-Generation Mobile Communication and Internet Systems with Hash Based Signatures Using Korean Cryptography Algorithms / M. Sim et al // Sensors. – 2023. – № 23(17). – P. 7558. https://doi.org/10.3390/s23177558.
7. A cryptographic hash function based on a modified Sponge scheme / R.M. Ospanov et al // Eurasian Journal of Mathematical and Computer Applications. – 2022. – № 10(2). – P. 55-70. https://doi.org/10.32523/2306-6172-2022-10-2-55-70.
8. TANBA-SPHINCS+ - post-quantum cryptographic hash-based digital signature algorithm. / R.M. Ospanov et al // Bulletin Abai KazNPU. Series of Physics & Mathematical Sciences. – 2025. – № 1(89). – P. 235-246. https://doi.org/10.51889/2959-5894.2025.89.1.020. (In Russian).
9. Goldreich O. Foundations of Cryptography: Basic Applications, volume 2. Cambridge University Press, Cambridge, UK, 2004.
10. SPHINCS: Practical Stateless Hash-based Signatures / D.J. Bernstein et al // In: Osvald, E., Fischlin, M. (eds), Advances in Cryptology, Lecture Notes in Computer Science. – 2015. – vol 9056. – P. 368-397. https://doi.org/10.1007/978-3-662-46800-5_15.
11. Gueron S. Sphincs-Simpira: Fast Stateless Hash-Based Signatures with Post-Quantum Security / S. Gueron, N. Mouha // Cryptology ePrint Archive. – 2017. – P. 2017/645. https://eprint.iacr.org/2017/645.
12. A Family of Efficient Permutations Using the AES Round Function / S. Gueron et al // Advances in Cryptology ASIACRYPT. – 2016. – vol. 10031. – P. 95-125. https://doi.org/10.1007/978-3-662-53887-6_4 .
13. Kölbl S. Putting Wings on SPHINCS / S. Kölbl // Cryptology ePrint Archive. – 2017. – P. 2017/898. https://eprint.iacr.org/ 2017/898.
14. Aumasson J.-P. Improving stateless hash-based signatures / J.-P. Aumasson, G. Endignoux // In: Smart, N. (eds) Topics in Cryptology – CT-RSA. – 2018. – vol 10808. – P. 219-242. https://doi.org/10.1007/978-3-319-76953-0_12.
15. Zhang K. Sphincs-α: A compact stateless hash-based signature scheme / K. Zhang, H. Cui, Y. Yu // Cryptology ePrint Archive. – 2022. – P. 2022/059. https://eprint.iacr.org/2022/059.
16. SPHINCS+C: Compressing SPHINCS+ with (Almost) No Cost / A. Hulsing et al // In: Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 22-24 May 2023, P. 1435-1453. https://doi.org/10.1109/SP46215.2023.10179381.
17. Grebnev S. Hypericum – a quantum-resistant digital signature project for standardization in Russia, RusCrypto 2023. Available online: https://ruscrypto.ru/resource/archive/rc2023/files/02_grebnev.pdf. (In Russian).
18. Turchenko O. Hypericum: a post-quantum digital signature for standardization in Russia / O. Turchenko, S. Grebnev // J Comput Virol Hack Tech. – 2024. – № 20. – P. 455-483. https://doi.org/10.1007/s11416-024-00520-6.
19. Sphinx-in-the-Head: Group Signatures from Symmetric Primitives / L. Chen et al // ACM Trans. Priv. Sec. – 2024. – Vol. 27, Issue 1, Article 11. – P. 1-35. https://doi.org/10.1145/3638763.
20. Abri M. Shorter Hash-Based Signatures Using Forced Pruning / M. Abri, J. Katz // Cryptology ePrint Archive. – 2025. – P. 2025/2069. https://eprint.iacr.org/2025/2069.

21. Qin Z. Extending the SPHINCS+ Framework: Varying the Tree Heights and Chain Lengths / Z. Qin, S. Sun // Cryptology ePrint Archive. – 2025. – P. 2025/2236. https://eprint.iacr.org/2025/2236.

***Acknowledgments***

*This research was funded by Ministry of Science and Higher Education of the Republic of Kazakhstan, grant number AP23486901.*

**Е.Н. Сейткулов*, Р.М. Оспанов[1], Б.Б. Ерғалиева[1], К.А. Утебаев[2]**
[1]Л.Н.Гумилев атындағы Еуразия ұлттық университеті, 010008, Қазақстан Республикасы, Астана,
[2]MEPhI Ұлттық зерттеу ядролық университетінің Алматы филиалы,
050040, Қазақстан Республикасы, Алматы
*e-mail: yerzhan.seitkulov@gmail.com

**SPHINCS+ НЕГІЗІНДЕГІ ЦИФРЛЫҚ ҚОЛТАҢБА СХЕМАЛАРЫН ІСКЕ АСЫРУ ЖӘНЕ САЛЫСТЫРМАЛЫ ТАЛДАУ**

*Посткванттық криптографияда кванттық компьютерлер тарапынан туындайтын қауіптерге төтеп бере алатын алгоритмдерді құру мәселелері қарастырылады, бұл олардың ішкі архитектурасын жобалауда жаңа тәсілдерді қолдануды талап етеді. Бұл салада цифрлық қолтаңбаларға ерекше назар аударылады, себебі олар цифрлық ортада деректердің түпнұсқалығын растауды және тұтастығын қамтамасыз етеді. Ең перспективалы бағыттардың бірі ретінде хештеуге негізделген цифрлық қолтаңба схемалары қарастырылады. Олар жоғары тиімділігімен және дәлелденген криптографиялық беріктігімен ерекшеленеді, сондықтан электрондық қолтаңба үшін постквантты шешімдердің негізгі түрлерінің қатарына жатады. Осы жұмыста SPHINCS және SPHINCS+ негізіндегі кейбір схемалардың, соның ішінде SPHINCS+-тың жаңа нұсқасы болып табылатын TANBA-SPHINCS+ схемасының негізгі компоненттері, жабық және ашық кілттердің, сондай-ақ қолтаңбалардың (байтпен) өлшемдері салыстырылды. Бұл схема бірнеше ішкі түрлендіру функцияларын қамтитын модификацияланған Sponge схемасына негізделген криптографиялық хеш-функцияны қолданады. Ұсынылған тәсіл SPHINCS+ классикалық архитектурасын кеңейтеді және ішкі түрлендірулерді икемді түрде баптауға мүмкіндік береді. Мұндай құрылым алгоритмнің құрылымдық әртүрлілігін арттырады және криптоаналитикалық шабуылдардың белгілі бір түрлеріне қарсы тұрақтылықты күшейтуге ықпал етуі мүмкін. Есептеу эксперименттерін жүргізу үшін бағдарламалық іске асыру әзірленді. Мақалада өнімділік бойынша салыстырмалы эксперименттік нәтижелер ұсынылған.*

***Түйін сөздер:*** *криптографиялық алгоритм, сандық қолтаңба, криптографиялық хэш функциясы, постквантты криптография, ақпараттық қауіпсіздік.*

**Е.Н. Сейткулов*, Р.М. Оспанов[1], Б.Б. Ергалиева[1], К.А. Утебаев[2]**
[1]Евразийский национальный университет им. Л.Н.Гумилева, 010008, Республика Казахстан, Астана
[2]Алматинский филиал Национального исследовательского ядерного университета «МИФИ»,
050040, Республика Казахстан, Казахстан

**РЕАЛИЗАЦИЯ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ СХЕМ ЦИФРОВОЙ ПОДПИСИ НА ОСНОВЕ SPHINCS+**

В постквантовой криптографии рассматриваются вопросы создания алгоритмов, способных устойчивость к угрозам со стороны квантовых компьютеров, что предполагает применение новых подходов к построению их внутренней архитектуры. Значительное внимание в этой области уделяется цифровым подписям, поскольку они обеспечивают подтверждение подлинности и сохранность данных в цифровой среде. Одним из наиболее перспективных направлений считаются схемы цифровой подписи, основанные на хешировании. Они отличаются высокой эффективностью и обладают доказанной криптографической стойкостью, благодаря чему относятся к числу ключевых постквантовых решений для электронной подписи. В данной работе проведено сравнение основных компонентов, размеров закрытых ключей, открытых ключей и подписей (в байтах) для некоторых вариантов схем на основе SPHINCS и SPHINCS+, включая схему TANBA-SPHINCS+, новый вариант SPHINCS+. В этой схеме применяется криптографическая хеш-функция на основе модифицированной схемы Sponge с несколькими внутренними функциями. Предложенный подход расширяет классическую архитектуру SPHINCS+ и позволяет гибко настраивать внутренние преобразования. Такая конструкция повышает структурное разнообразие алгоритма и может способствовать повышению устойчивости к определенным типам криптоаналитических атак. Для проведения вычислительных экспериментов была написана программная реализация. В статье представлены сравнительные экспериментальные результаты по производительности.

**Ключевые слова:** криптографический алгоритм, цифровая подпись, криптографическая хеш-функция, постквантовая криптография, информационная безопасность.

**Information about the authors**

**Yerzhan Seitkulov*** – PhD, L.N. Gumilyov Eurasian National University, Republic of Kazakhstan; e-mail: yerzhan.seitkulov@gmail.com. ORCID: https://orcid.org/0000-0002-5172-8339.

**Ruslan Ospanov** – Researcher, Gumilyov Eurasian National University, Republic of Kazakhstan; e-mail: ospanovrm@gmail.com. ORCID: https://orcid.org/0000-0002-0771-575X.

**Banu Yergaliyeva** – Researcher, Gumilyov Eurasian National University, Republic of Kazakhstan; e-mail: banu.yergalieva@gmail.com. ORCID: https://orcid.org/0000-0002-1252-857X.

**Kuat Utebayev** – Master student, Almaty Branch, National Research Nuclear University MEPhi, Almaty 050040, Kazakhstan; e-mail: utebayevkuat@gmail.com. ORCID: https://orcid.org/0009-0004-3032-5049.

**Авторлар туралы ақпарат**

**Ержан Сейтқұлов*** – PhD, Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Қазақстан Республикасы; e-mail: yerzhan.seitkulov@gmail.com. ORCID: https://orcid.org/0000-0002-5172-8339.

**Руслан Оспанов** – Ғылыми қызметкер, Гумилев атындағы Еуразия ұлттық университеті, Қазақстан Республикасы; e-mail: ospanovrm@gmail.com. ORCID: https://orcid.org/0000-0002-0771-575X.

**Бану Ерғалиева** – Ғылыми қызметкер, Гумилев атындағы Еуразия ұлттық университеті, Қазақстан Республикасы; e-mail: banu.yergalieva@gmail.com. ORCID: https://orcid.org/0000-0002-1252-857X.

**Қуат Өтебаев** – магистрант, MEPhi Ұлттық зерттеу ядролық университетінің Алматы филиалы, Алматы 050040, Қазақстан; e-mail: utebayevkuat@gmail.com. ORCID: https://orcid.org/0009-0004-3032-5049.

**Информация об авторах**

**Ержан Сейткулов*** – PhD, Л.Н. Евразийский национальный университет имени Гумилева, Республика Казахстан; e-mail: erzhan.seitkulov@gmail.com. ORCID: https://orcid.org/0000-0002-5172-8339.

**Руслан Оспанов** – научный сотрудник Евразийского национального университета имени Гумилева, Республика Казахстан; e-mail: ospanovrm@gmail.com. ORCID: https://orcid.org/0000-0002-0771-575X.

**Бану Ергалиева** – научный сотрудник Евразийского национального университета имени Гумилева, Республика Казахстан; e-mail: Banu.yergalieva@gmail.com. ORCID: https://orcid.org/0000-0002-1252-857X.

**Куат Утебаев** – магистрант, Алматинский филиал Национального исследовательского ядерного университета МИФИ, Алматы Казахстан; e-mail: utebayevkuat@gmail.com. ORCID: https://orcid.org/0009-0004-3032-5049.

*Received 04.05.2026*
*Revised 19.06.2026*
*Accepted 22.06.2026*

https://doi.org/10.53360/2788-7995-2026-3(23)-8

IRSTI: 28.23.25

CC BY 4.0
Check for updates

**A. Zhakanova[1], Zh. Oralbekova[1]*, D. Oralbekova[2]**
[1]Astana IT University,
010000, Republic of Kazakhstan, Astana, 55/11 Mangilik El avenue, C1 block
[2]Institute of Information and Computational Technologies,
050010, Republic of Kazakhstan, Almaty, 28 Shevchenko street
*e-mail: zh.oralbekova@astanait.edu.kz

# DEVELOPMENT OF A MODEL OF KAZAKH SIGN LANGUAGE FOR CLOSED-SET PHRASE RECOGNITION AND TRAINING

***Annotation:*** *Kazakh Sign Language (KSL) remains underrepresented in current sign language processing research due to a lack of large, annotated datasets and specialized computational tools. This paper presents a deep learning-based framework for phrase-level recognition of sign language using visual keypoint sequences, in which each recognized phrase is mapped to its gloss label and a verified reference translation in Russian. The approach employs the FluentSigners-50 dataset as a benchmark corpus and extracts pose*

©A. Zhakanova, Zh. Oralbekova, D. Oralbekova, 2026