Oleksandr MARUKHNENKO[1], Yevgeniy KOTUKH[2]

Opiekun naukowy: Gennady KHALIMOV[3]

ANALIZA STRUKTURY I PARAMETRÓW ALGORYTMU SPHINCS +

Streszczenie: W artykule omówiono kryptosystem SPHINCS + i jego parametry. Analizowano ich wpływ na bezpieczeństwo podpisu i kompromisy między rozmiarem a prędkością. Zaproponowano kilka zestawów parametrów dla bezpieczeństwa 384 i 512-bitowego.

Słowa kluczowe: podpis oparty na haszowaniu, SPHINCS, drzewo Merkle

ANALYSIS OF THE STRUCTURE AND PARAMETERS OF THE SPHINCS + ALGORITHM

Summary: The article considers the SPHINCS+ cryptosystem and its parameters. We analize their influence on signature security and tradeoffs between size and speed. We propose few sets of parameters for 384 and 512-bit security.

Keywords: hash-based signature, SPHINCS, Merkle tree

1. Introduction

Most modern asymmetric cryptosystems are based on calculations in rings, prime number fields and groups of points of elliptic curves, their security is based on the complexity of the solution of factorization problems (RSA), discrete logarithm in a simple field (DSA) or group of points of the elliptic curve. All of these problems can be solved using a quantum computer of sufficient computing power that can be created in the near future. In order to analyze and standardize new cryptosystems, in 2016, NIST announced the launch of an open competition. The first round introduced two hash-based digital signature algorithms - SPHINCS+[1] and Gravity-SPHINCS[2], which are independent modifications of the previously developed SPHINCS

[1] Kharkiv National University of Radioelectronics, Department of Information Technology Security, Cybersecurity: oleksandr.marukhnenko@nure.ua
[2] Ph.D., Kharkiv National University of Radioelectronics, Department of Information Technology Security, Cybersecurity: yevgenkotukh@gmail.com
[3] Ph.D., Kharkiv National University of Radioelectronics, Department of Information Technology Security, hennadii.khalimov@nure.ua

algorithm [3]. Only the SPHINCS + cryptosystem has passed the second round.

A feature of hash-based cryptosystems is the limited number of signatures that can be created using a single key. To eliminate this shortcoming, firstly a hash-tree structure which is built with one-time signature keys, then a hyper-tree structure which is built with hash-trees were proposed. The use of such a multilevel structure requires setting and justifying a number of parameters.

The purpose of the paper is to analyze the structure of the SPHINCS+ algorithm and to propose certain parameters for the given security level.

1.1. Organization of this paper

In section 2 we discus the algorithms underlying the SPHINCS+ cryptosystem. In section 3, we outline the structure of SPHINCS+ and analyze its security. In Section 4, we propose specific system parameter values that can be used to provide a given stability. Let's summarize the results and give suggestions for further research.

2. General idea

The general idea of hash-based signatures is that an array of bit strings is selected according to the message, the elements of which are selected and possibly hashed times according to the bits of the message.

The stability of this class of EDS is based on the one-way nature of hash functions and their resistance to collisions. Consider the benefits of signatures based on hash functions:

1) postquantum - modern cryptographic hash functions, such as SHA-2 and SHA-3, are resistant to quantum attacks and can be safely used even after the creation of quantum computers;

2) easy modification - EDS algorithms of this class do not determine the used hash function, which makes it easy to replace the hashing mechanism if any vulnerabilities are found in it;

3) flexibility - complements the previous feature by the fact that depending on the requirements of the system, the hash functions and system parameters which best satisfy them can be selected, and the compromise between stability, performance and memory is reached.

The disadvantages include:

1) limited number of signatures - all algorithms of this class limit the maximum number of signatures that can be created using one key pair, but in modern cryptosystems the number can be very high, so actually this limitation is removed;

2) large signature size - modern signatures have a complex structure that includes many elements.

The hash function EDS can be classified as follows:

- one-time signatures - OTS (Lamport, Winternitz);
- few-time signatures using Merkly trees;
- few-time signatures with gradual decrease in security (HORS, FORS, PORS);
- multi-time using hypertrees (SPHINCS, SPHINCS+, Gravity-SPHINCS, $XMMS^{MT}$).

2.1. Winternitz signature

The general idea of the Winternitz (WOTS) signature is as follows: the secret key is an array of random bit sequences, the public key is an array of hashed w (Winternitz parameter) times private key elements, when creating a signature a message is divided into blocks by $\log w$ bits, the checksum is added at the end, the signature is added of the private key elements cached a certain number of times, depending on the value of the respective block. During verification, the signature elements are hashed to w times and compared with the public key. Details of the algorithm can be found in [1], [4].

The following additional parameters are introduced:

$len_1 = \left\lceil \frac{m}{w} \right\rceil$ – number of blocks $\log w$ bits in a message;

$len_2 = \left\lfloor \frac{\log_2 (len_1 * (w-1))}{\log_2 w} \right\rfloor + 1$ – checksum length in blocks;

$len = len_1 + len_2$ – block length of message with checksum.

2.2. Merkle tree

The method of multiple use of hash-based signature key-pair was proposed by Merkle and is based on the use of so-called hash trees or Merkle trees [5]. A hash tree is called a complete binary tree, which leaves contain hashes from data blocks, and the inner nodes contain hashes from the concatenation of values in the child nodes. The root node of the tree contains a hash from the entire dataset, that is, the hash tree is a one-way hash function. The structure of the Merkle tree with 4 leaves is shown in Figure 1.

The leaves of a tree are the OTS public key hash values, such as the Winternitz OTS. The public key is the root of the tree. To confirm that the OTS public key used in the signature belongs to this tree, the authentication path is added to the signature - the tree elements required to pass from the given letter to the tree root, the number of the one-time key used, and the OTS public key itself.

Thus, signature verification consists of two steps - public key signature verification, key authentication. In algorithms in which the public key is fully computed from the signature (WOTS), there is no need to pass the used key, which reduces the size of the signature.

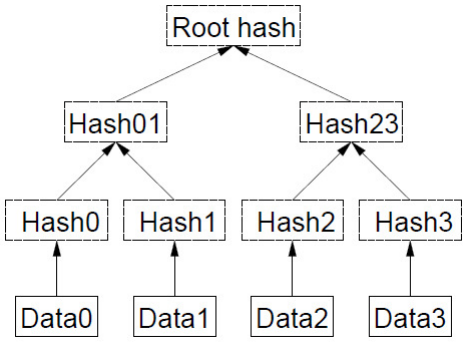

Figure 1. Merkle tree

2.3. FORS signature

The FORS signature belongs to a class of few-time signatures with a gradual decrease in security. The basic idea behind this group of algorithms is that the private key is large enough, for example, 1 megabyte, every signature contains a certain "random" part of the key, thus each new signature increases the probability of forgery. The safe usage of one key pair is determined by the system's security requirements.

The FORS (Forest Of Random Subset) algorithm is a modification of the HORST algorithm and is used in the SPHINCS+ scheme. The essence of the algorithm is that the message uniquely corresponds to a subset of elements of a given set (secret key), which becomes a signature. The special feature of FORS is that for each subscribed block, a separate array of random numbers is used.

System parameters:

$t = 2^{\tau}$ – size of key array.

k – number of elements in signature, $k\tau = n$.

Private key. Array of $k*t$ random values.

$$SK = \begin{bmatrix} sk_{0,0}, sk_{0,1}, ..., sk_{0,t-1} \\ sk_{1,0}, sk_{1,1}, ..., sk_{1,t-1} \\ ... \\ sk_{k-1,0}, sk_{k-1,1}, ..., sk_{k-1,t-1} \end{bmatrix}, sk_i = \{0,1\}^m$$

Public key. It is formed as follows: public key elements consistently form k hash-trees with t leaves in each, the public key will be a hash from the concatenation of the roots of all the trees, as shown in Figure 2.

$PK = H(Root_1 \| Root_2 \| ... \| Root_{k-1})$.

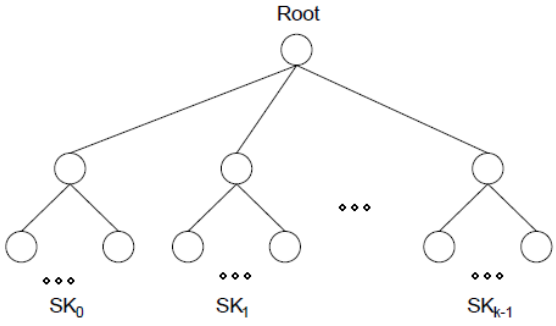

Figure 2. FORS public key computing

Signature contains k elements of private key and authentication paths for them. More details can be found in [1].

2.4. Hyper-trees

The mechanisms described earlier do not solve the problem of the limited number

of uses of one key pair. For FORS, stability is proportional to the number of key elements, so more signatures require an increase in the size of the private key (can be replaced by the HPVC initializer) and signature. For signatures based on Merkle trees, the number of uses is also limited when generating a key pair; also to build a tree on *n* signatures, it is necessary to generate *n* OTS key pairs and make *2n* hashes inside the tree, which already requires significant calculations for trees with thousands of leaves.

An alternative solution is to use a hypertree structure, which is a tree of hash trees, the leaves of trees of the upper level are used to sign the roots of trees lying below the level. The public key of the user is the root of the top tree. This scheme allows you to generate lower level trees without changing the public key. The signature of the message includes the signature itself, the authentication paths at each level, and the signatures for the intermediate nodes.

3. SPHINCS+ cryptosystem

The most promising EDS hash-based algorithms is the SPHINCS algorithm family: two independent modifications of the SPHINCS signature [3] - SPHINCS+ [1] and Gravity-SPHINCS [2], which participated in the first round of the NIST competition. The analysis of the SPHINCS + algorithm continues in the second round.

The structure of the SPHINCS+ algorithm is similar to the structure of SPHINCS, but some internal components have been changed, especially the multiple signature scheme.

This algorithm uses a hyper-tree of the following form: tree leaves at all layers contain the WOTS+ public key hash values that are used to sign the root of the corresponding tree at a lower layer, at the lowest level the keys are used to sign the FORS public keys. The FORS key pair index is selected based on the randomized message digest. The message signature is a FORS signature, the corresponding WOTS+ signature from the hyper-tree leaf, the root tree signatures, the authentication paths, and the random sequence used to generate the digest.

System parameters:

- n – security parameter, length of all elements in hyper-tree in bytes;
- h – hyper-tree height;
- d – number of layers, each one contains hash-trees with height $h' = h / d$;
- w – Winternitz parameter;
- k – number of FORS trees;
- t – number of leaves in FORS tree.

SPHINCS+ private key contains next components:

1) random *n*-bit string SK.seed, which is used for pseudorandom generation of the hyper-tree elements: WOTS+ and FORS;

2) random *n*-bit string SK.prf, which is used for computing randomized message digest;

3) SPHINCS+ public, because its components are necessary for some computations.

SPHINCS+ public key contains next components:

1) root of the single Merkle tree on the top layer of the hyper-tree PK.root;

2) random *n*-bit string PK.seed, which is used for additional randomization in some operations, for example in WOTS+ chain function.

For PRNG or keyed hash-function initialization most elements are associated with unique address of one of the following types:

- WOTS+ hash address – used in chain function;
- WOTS+ public key compression address – used in hashing of WOTS+ public key;
- hash tree address – used for generation of WOTS+ private keys;
- FORS tree address – used for generation of FORS+ private keys;
- FORS tree roots compression address – used in hashing of FORS trees roots.

The structure and features of using each type of address are described in details in [1].

3.1. SPHINCS+ security

The security of the SPHINCS+ cryptosystem is determined by the choice of the previously described hash function and PRNG used, the authors [1] propose to use as the PRNG the same hash function.

The authors of the algorithm evaluate the classical resistance to general attacks by the following formula:

$$b = -\log\left(\frac{1}{2^{8n}} + \sum_{\gamma}\left(1-\left(1-\frac{1}{t}\right)^{\gamma}\right)^{k}\binom{q}{\gamma}\left(1-\frac{1}{2^{h}}\right)^{q-\gamma}\frac{1}{2^{h\gamma}}\right).$$

The resilience of a system to quantum attacks can be calculated as

$$b = -\frac{1}{2}\log\left(\frac{1}{2^{8n}} + \sum_{\gamma}\left(1-\left(1-\frac{1}{t}\right)^{\gamma}\right)^{k}\binom{q}{\gamma}\left(1-\frac{1}{2^{h}}\right)^{q-\gamma}\frac{1}{2^{h\gamma}}\right).$$

Thus, the stability parameters are affected by the system parameters n, k, t. You must use a cryptographic hash function with an output length of n bytes to provide the specified level of security.

4. Optimal parameters for SPHINCS+

In the previous section, we determine that only part of the parameters affect the security of the cryptosystem, but all the parameters affect the size of the signature, the time of its creation and verification. The size of the signature is equal $(h + k(\log t + 1) + d * len + 1) * n$ bytes. The computational complexity is determined by the number of uses of the various variations of the hash function, which are described in detail in…

When selecting specific parameters of the algorithm, a space-time trade-off must be reached, on the one hand the signature may be shorter, but the creation and verification time will be longer, on the other hand, the creation and verification will be faster, but the signature will take more memory. The authors of SPHINCS+ offer two sets of parameters: "fast" and "small" for security levels of 128, 192 and 256 bits (Table 1).

Table 1. Parameters for 128, 192 and 256 bits security

	n	h	d	log(t)	k	w	bitsec	sigbytes
SPHINCS+-128s	16	64	8	15	10	16	133	8080
SPHINCS+-128f	16	60	20	9	30	16	128	16976
SPHINCS+-192s	24	64	8	16	14	16	196	17064
SPHINCS+-192f	24	66	22	8	33	16	194	35664
SPHINCS+-256s	32	64	8	14	22	16	255	29792
SPHINCS+-256f	32	68	17	10	30	16	254	49216

The value of the Winternitz parameter is the same for each parameter set and equals 16, which is caused by the convenient splitting of the message into tetrades.

We offer the following parameter values to provide stability levels of 384 and 512 bits (Table 2).

Table 2. Parameters for 384 and 512 bits security

	n	h	d	log(t)	k	w	bitsec	sigbytes
SPHINCS+-384s	48	64	8	16	23	16	391	59904
SPHINCS+-384f	48	60	15	10	39	16	390	94800
SPHINCS+-512s	64	66	6	19	26	16	519	87872
SPHINCS+-512f	64	65	13	12	42	16	512	148160

The selected parameters are one of the possible combinations and can vary widely depending on the system requirements. Assessment of the optimality of the proposed parameters requires further investigation.

5. Conclusions

Hash-based digital signatures are a promising area of post-quantum cryptography. The developers of the SPHINCS + cryptosystem managed to eliminate the shortcomings that were present in the previous algorithms. The complex structure of a hyper-tree requires the selection of several system parameters that affect the stability of the system, its speed of operation and the size of the signature, so it is difficult to unambiguously choose the best set of components, since depending on the specific application requirements may change. We proposed parameter values that can be applied in the SPHINCS + crypto system with 384 and 512-bit security.

REFERENCE

1. BERNSTEIN D.J., DOBRAUNIG CH., EICHLSEDER M. and others: SPHINCS+ – Submission to the NIST's post-quantum cryptography standardization process, 2017.
2. AUMASSON J.P., ENDIGNOUX G.: Gravity-SPHINCS – Submission to the NIST's post-quantum cryptography standardization process, 2017.
3. BERNSTEIN D.J., HOPWOOD D., HÜLSING A. and others: Sphincs: practical stateless hash-based signatures. Cryptology ePrint Archive, Report 2014/795, 2014.
4. HÜLSING A.: W-OTS+ – shorter signatures for hash-based signature schemes. In

Amr Youssef, Abderrahmane Nitaj, and Aboul-Ella Hassanien, editors, Progress in Cryptology – AFRICACRYPT 2013, volume 7918 of LNCS, pages 173–188. Springer, 2013.

5. MERKLE R.: A certified digital signature. In Gilles Brassard, editor, Advances inCryptology – CRYPTO ’89, volume 435 of LNCS, pages 218–238. Springer, 1990.