A study on fitting SPHINCS$^+$ to blockchain usage

Antônio Unias de Lucena, Marco Aurélio Amaral Henriques

[1]Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
13083-852 – Campinas, SP, Brasil

{alucena,marco}@dca.fee.unicamp.br

Abstract. *The advent of a large-scale quantum computer will make insecure the current leading blockchains' digital signature schemes. Among the quantum-resistant options for signature schemes, SPHINCS$^+$ stands out due to its security (based solely on hash functions) and due to being stateless. However, when used in a blockchain environment, its signature size considerably reduces the number of transactions stored per block, impacting the blockchain operation and economics. This paper presents a study on fitting SPHINCS$^+$ to blockchain usage. The results show that, without compromising the security, a careful choice of SPHINCS$^+$ parameters can reduce both the signature size and the signature creation time, since the maximum number of signatures supported by a given key is limited to 16 million, a number considered more than sufficient for a blockchain environment.*

1. Introduction

The blockchain technology made it possible for digital environments to simultaneously exhibit the following three characteristics: security, decentralization, and distribution. Built initially to be the Bitcoin's ledger, the blockchain soon called the attention of other applications such as supply chain and logistics monitoring, data storage, and smart contracts. However, a critical element that guarantees blockchain security, the digital signature, will be under threat if a large-scale quantum computer is built.

Digital signatures allow the blockchain to become decentralized since users can trust them without a centralizing element. The security of the currently most used digital signature schemes relies on mathematical operations such as large integers factorization or discrete logarithm calculation of big integer numbers. These operations are infeasible for classical computers but straightforward for a large-scale quantum computer. This fact leads to a race to build new digital signature schemes as the main ones presently used, RSA and ECDSA, are not quantum-resistant.

The main existing quantum-resistant signing algorithms are lattice-based, code-based, multivariate-system based, and hash-based. In this study, we focus on the hash-based signature scheme SPHINCS$^+$. It was chosen because it is stateless: No data related to the keys already used is needed to be stored, and its security is well defined and based solely on hash function resistance to quantum attacks.

Modifying the digital signature scheme used by a blockchain to a quantum-resistant one is not straightforward. The main impact is on the signature size stored in a transaction, reducing the number of transactions stored in a block, influencing over

miners' fees and revenue, and leading to a demand for increasing the blockchain's block size, what changes the entire blockchain environment.

There are blockchains already using post-quantum signing algorithms. The IOTA [IOTA 2020] uses the Winternitz one-time signature scheme and the Quantum-Resistant Ledger [QRL 2020] uses the stateful XMSS as its digital signature scheme.

This paper presents a study on SPHINCS$^+$: a digital signature scheme that offers solutions to the limitations of using WOTS (capable of performing only one signature for a given public key) and XMSS (not ideal to multi-platform applications) in a blockchain environment at the expense of a larger signature size. To overcome this negative point, we present a study on fitting SPHINCS$^+$ to blockchain usage. The results show that the signature size can be halved at the expense of reducing the maximum number of possible signatures with a given public key.

This paper is organized as follows: Section 2 introduces the concept of blockchain, section 3 details how a quantum computer makes present blockchains insecure and lists the existing post-quantum signing algorithms, section 4 presents the SPHINCS$^+$ signature scheme, and section 5 shows the results of modifying its parameters to get a smaller signature size. Lastly, we discuss some conclusions and proposals for future work.

2. Blockchain

Blockchain is an append-only, resistant to tampering, decentralized, and distributed database. Designed to be Bitcoin's cryptocurrency digital ledger [Nakamoto 2009], it stores all transactions performed by its users, preventing double-spending, i.e., using a monetary value more than once.

The blockchain is resistant to tampering because its sequential blocks are firmly connected. In Bitcoin's blockchain, each block header, except the first one, stores the hash value of the previous block data. In addition, all transactions are organized as a Merkle tree, and its root is stored in the block header, making any undetectable tampering in a transaction content infeasible. So any change in data previously stored is easily visible by the blockchain users, and no data stored in the blockchain can be modified without detection.

Decentralization is possible because the blockchain users can check the integrity and authenticity of a transaction without a central authority. That occurs thanks to the use of digital signatures: A user is identified by an address, derived from his public key, at the blockchain network, and his transactions are authenticated by means of digital signatures.

Finally, the blockchain is distributed because there is no central repository for its content, and any user can store and distribute its data. Thus, it is not possible to modify a blockchain's content without changing all copies of it.

Figure 1 shows how Bitcoin's blockchain operates. Any activity performed by a user at Bitcoin's blockchain occurs by publishing a transaction on the blockchain network. A transaction contains the following data: (1) The user's public key, (2) the transferred amount of digital asset, (3) the destination address, (4) a digital signature, proving the user is the owner of both the public key and address associated with that digital asset. After the block containing this transaction receives six confirmations, the digital asset ownership can be considered permanently transferred to the addressee.

The block body stores the transactions, organized in a Merkle tree, and the root of this tree (Merkle root) is stored at the block header alongside other data such as the previous block hash.

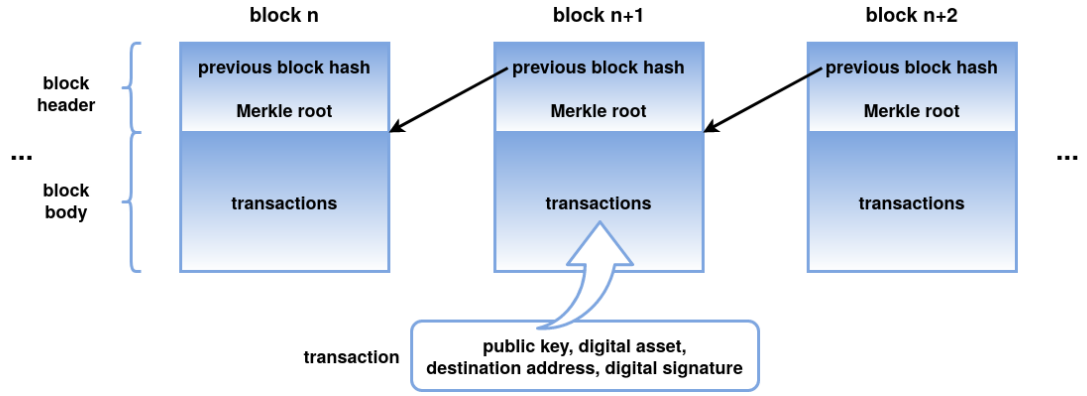

Figure 1. Bitcoin's blockchain operation

There are some bottlenecks, mainly on data throughput and storage capacity, when using a blockchain in applications such as supply chain and logistics monitoring, data storage, and smart contracts. To overcome these limitations, new forms of building and organizing a blockchain have been proposed. Initially, the blockchain was conceived to be public and open to the participation of any user. However, nowadays, there are private (centralized) and consortium (managed by a group of users) blockchains.

From the moment a user publishes its signature and its public key in a transaction, it is susceptible to cryptanalysis. So a large-scale quantum computer can turn insecure blockchains using signing algorithms such as ECDSA (Elliptic Curve Digital Signature Algorithm), leading to the need for upgrading the used signing algorithm to a quantum-resistant.

Nonetheless, changing the digital signature causes considerable impacts in the blockchain operation. In the case of Bitcoin's blockchain, as the existing quantum-resistant signing algorithms have a bigger size in bytes for their signature and key pairs, the number of transactions stored in a block is reduced. This fact impacts miners' fees and revenue, leading to the demand for increasing the blockchain's block size. There are also impacts on transactions' creation and verification times. The previous work of de Lucena and Henriques [de Lucena and Henriques 2018] details how the modification of the Bitcoin's blockchain signature scheme impacts its operation.

The next section explains how some algorithms for quantum computers will make the classical asymmetric cryptography obsolete, and also presents a list of the existing quantum-resistant options for signing algorithms.

3. Post-quantum signing algorithms

The emergence of a large-scale quantum computer will modify how certain computational problems are solved. One example is the calculation of discrete logarithms and factorization of large integer numbers that, thanks to Shor's algorithm [Shor 1994], will have polynomial complexity in quantum computers rather than exponential complexity as in classical ones. As the difficulty in doing such calculations is the mathematical basis

that grants security to the public-key algorithms ECDSA (Elliptic Curve Digital Signature Algorithm) and RSA (Rivest-Shamir-Adleman), respectively, these algorithms are fated to be broken by quantum computers.

The Grover's algorithm [Grover 1996] halves the brute force search space for symmetric keys and hashes. Consequently, hash functions or symmetric cryptographic algorithms such as AES (Advanced Encryption Standard) will have their bit security halved. The simple solution to deal with such a threat is to duplicate the number of bits of the used hash function or symmetric algorithm.

As of this writing, this is the current status of quantum computing. The most powerful quantum computers are the Google Britlescone with 72-qubits and the IonQ 79-qubit machine [Nam and Maslov 2019]. On breaking public-key cryptography, researchers estimated that 2048-bit RSA keys could be factorized in 8 hours using 20 million noisy qubits [Gidney and Ekerå 2019] and showed that a 160-bit elliptic curve cryptographic key could be broken on a quantum computer using around 1000 qubits while factoring a 1024-bit RSA would require about 2000 qubits [Proos and Zalka 2003].

To anticipate the quantum computer threat, the scientific community has been developing post-quantum signing (PQS) algorithms. The main ones are (a) code-based cryptosystems, based on decoding random linear code; (b) lattice-based cryptosystems, built on the hardness of the shortest-vector problem; (c) multivariate-based cryptosystems, relying on solving multivariate quadratic equations and (d) hash-based signatures, whose security is based on the pre-image and second pre-image resistance of cryptographic hash functions.

Table 1 [Fernández-Caramès and Fraga-Lamas 2020] summarizes the PQS algorithms with 256 bits of security that passed to the NIST second round Post-Quantum Cryptography Standardization Process [NIST 2020a]. The GeMSS 256 [GEMSS 2020], LUOV Level (chacha8) [LUOV 2020] and Rainbow Vc [Ding and Schmidt 2005] are multivariate-based algorithms, while the PICNIC2 L5-FS [PICNIC 2020] and SPHINCS$^+$ [Bernstein et al. 2019] are hash-based signatures.

Table 1. NIST second round PQS algorithms with 128-bit security level against quantum computers.

Algorithm	public-key size	private-key size	signature size
GeMSS 256	3,040.70 KB	75.89 KB	576 bits
LUOV Level 5 (Chacha 8)	82.0 KB	32 bytes	440 bytes
PICNIC2 L5-FS	64 bytes	32 bytes	54,732 bytes
Rainbow Vc	1,705.5 KB	1,227.1 KB	1,632 bits
SPHINCS$^+$	64 bytes	32 bytes	29,792 bytes

In a blockchain environment, a transaction stores both public key and signature. Therefore a good candidate for a signing algorithm is one that leads to a small sum in bytes of these two values. Hence, this study focuses on the SPHINCS$^+$ signature scheme. Even though it is not among the finalists, it is listed as an alternate candidate by the Third Round of the NIST PQC Standardization Process [NIST 2020b]. The next section discusses hash-based signatures in more detail. The evaluation of other signing algorithms options is left for future work.

Hash-based signatures Hash-based signatures (HBS) are digital signatures that make use of cryptographic hash functions. The first HBS was designed by Lamport [Lamport 1979], but, due to its large signature size compared to RSA and ECDSA signatures, it was not widely adopted. Merkle modified Lamport's scheme and developed WOTS (Winternitz One-Time Signature) signature [Merkle 1979]. This signature scheme has smaller signatures and key sizes but demands more hash calculations. Lamport and WOTS signing algorithms reveal part of the private key when a signature is verified. That is why a private key can not sign more than one message, and they are called one-time signature (OTS) schemes.

Cryptographic hash functions are one-way functions that transform a message of arbitrary length into a fixed-length array of bits. To be useful to cryptography, they must hold the following properties: (1) It is not possible to find a message M from its hash H(M) (non-invertible function or pre-image resistance); (2) Given a message M and its hash H(M) it is computationally infeasible to find a message M' such as H(M) = H(M') (resistance to second pre-image); (3) It is computationally infeasible to find two messages M and M' such that H(M) = H(M') (resistance to collision). Examples of hash functions that share such properties are the SHA (Secure Hash Algorithm) family and RIPEMD-160.

Among the HBS schemes, the most well known are XMSS [Buchmann et al. 2011] and SPHINCS$^+$. Both are built using WOTS signatures in a way it is possible to perform more than one signature for a given public key. The second scheme is stateless: A user does not need to store information about the private keys already used in signatures.

The XMSS scheme is built resembling a Merkle tree whose leaves store a public-private OTS key pair used at each new signature. Each XMSS leaf stores WOTS$^+$ [Hülsing 2013] keys that can only be used once. In contrast to SPHINCS [Bernstein et al. 2015] and SPHINCS$^+$, this scheme is stateful.

The SPHINCS signature scheme is built in a similar way to XMSS, but instead of being constituted by only one tree, it is built as a set of trees, forming a hypertree organized in various layers. In each of these layers, except the lowest one, the leaves are roots of trees storing WOTS$^+$ keys. In the lowest layer, the leaves are roots of HORST trees, storing HORS (Hash to Obtain Random Subset) [Reyzin and Reyzin 2002] keys. HORS signature schemes allow a key being used more than once to sign a message and, therefore, they are known as FTS (few-times signatures) schemes.

SPHINCS$^+$ is a modification of SPHINCS that utilizes FORS (Forest of Random Subset) trees rather than HORS ones. SPHINCS$^+$ is one of the submissions to the NIST Post-Quantum Cryptography Standardization Process. Another submission to NIST, GRAVITY-SPHINCS [Aumasson and Endignoux 2017] is also based on SPHINCS. However, it uses PORS (PRNG to Obtain a Random Subset) trees instead of HORS ones. The next section further discusses the details of the SPHINCS$^+$ signing algorithm.

4. SPHINCS$^+$

Figure 2 summarizes a SPHINCS$^+$ hypertree. It is composed of FTS and OTS nodes representing FTS and OTS trees, respectively. The hypertree is composed of hash nodes,

forming the root of a tree inside a layer. Each root is connected to an OTS node, storing a WOTS$^+$ leaf. The objects in gray correspond to the authentication path of each select OTS node. The root of the SPHINCS$^+$ hypertree, *pk*, is the signature scheme's public key.

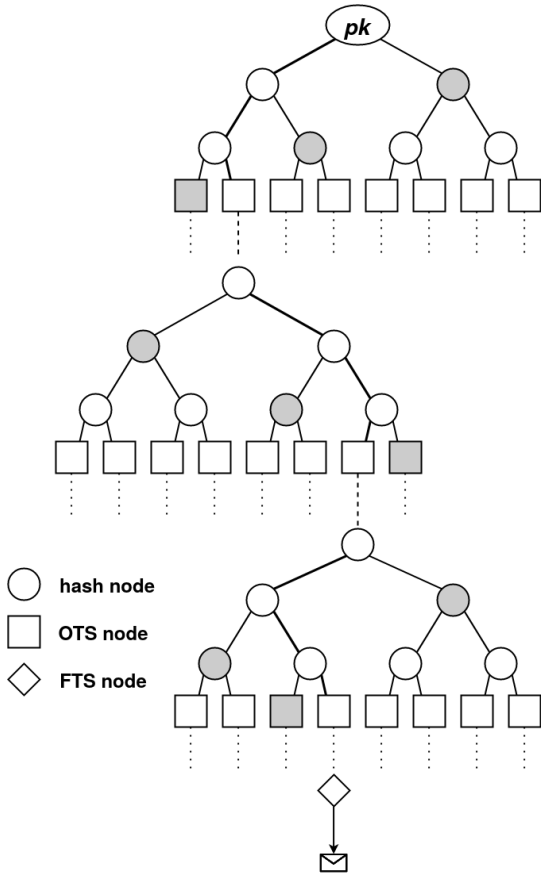

Figure 2. A generic SPHINCS$^+$ tree (source: [Bernstein et al. 2019]).

The lowest layer is formed by FTS nodes that are FORS trees, responsible for signing a message. A message, represented by the envelope, is signed using a combination of leaves picked up from each of the FTS trees. These leaves are addressed by a string formed by a random value and the digest of the signed message. The random value is used to mitigate side-channel attacks if someone stores any information from several signatures. Figure 3 illustrates a FORS tree. It is composed by k FTS trees with t leaves and height $b = \log_2(t)$.

Table 2 presents SPHINCS$^+$ parameters. The parameter n is the size in bytes of the digest generated by the hash function used by the SPHINCS$^+$ signature scheme; the parameter h corresponds to the hypertree height and defines the maximum number of signatures (2^h) that can be performed; the parameter d is the number of layers of the hypertree; the parameter w, Winternitz parameter, is the number of bits to be signed simultaneously in a WOTS$^+$ signature, providing a trade-off between signing time and signature size; the parameter k is the number of FTS trees in a FORS tree and, finally, the parameter b is the FORS tree height.

The parameter *len* is the number of n-byte blocks that form a WOTS$^+$ signature.

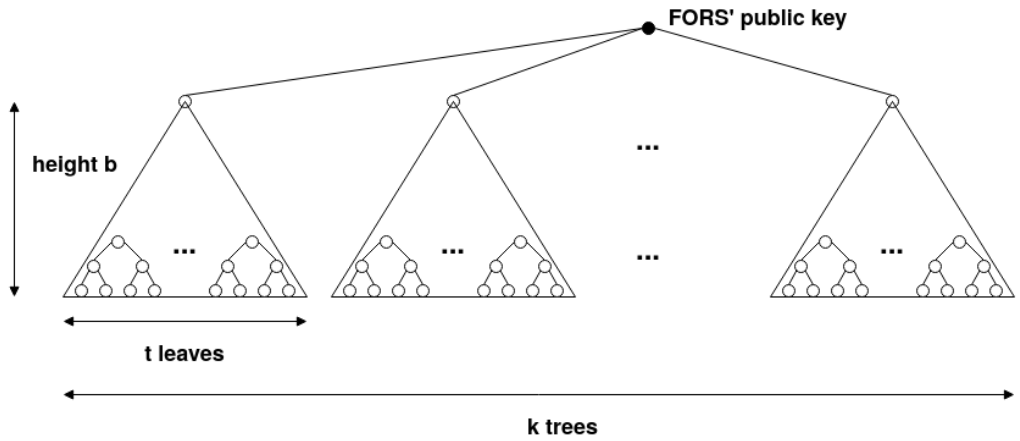

Figure 3. A FORS tree (source: [Bernstein et al. 2017]).

Table 2. SPHINCS⁺ parameters.

parameter	meaning	data range
n	security parameter	32 for SHA-256
h	hypertree height	up to 72
d	number of layers	must be a divisor of h
w	Winternitz parameter	4, 16 or 256
len	number of n-byte blocks in a WOTS⁺ signature	depends on the value used for w
k	number of FTS trees in FORS	1 to 64
b	FORS tree height	3 to 24

It is given by $len = len_1 + len_2$, where $len_1 = \lceil \frac{8n}{\log_2(w)} \rceil$ and $len_2 = \lfloor \frac{\log_2(len_1(w-1))}{\log_2(w)} \rfloor$. The parameter len_1 is the number of integers, between 0 and w - 1, that represents the hash digest of the WOTS⁺ message to be signed. The parameter len_2 is the number of integers, between 0 and w - 1, used to represent the WOTS⁺ checksum, needed to avoid signature forgery [Hülsing 2013]. SPHINCS⁺ parameters influence over its signature size, performance and security. The signature size has $(h + k \cdot (b + 1) + d \cdot len + 1)n$ bytes [Bernstein et al. 2017].

The performance for key generation, signature creation and verification are measured in hash function calls. Table 3 [Bernstein et al. 2017] informs how many times the pseudorandom function **PRF** and the tweakable hash functions **F**, **H** and $\mathbf{T_{len}}$ are called per operation. For more details on these function, we recommend the reading of SPHINCS⁺ submission to the NIST post-quantum project [Bernstein et al. 2017].

Table 3. Hash function calls at SPHINCS⁺ operation.

	key generation	signature creation	signature verification
F	$(2^{h/d}) \cdot w \cdot len$	$k \cdot t + d \cdot (2^{h/d}) \cdot w \cdot len$	$k \cdot t + d \cdot w \cdot len$
H	$2^{h/d} - 1$	$k \cdot (t-1) + d \cdot (2^{h/d} - 1)$	$k \cdot (t-1) + h$
PRF	$(2^{h/d}) \cdot len$	$k \cdot t + d \cdot (2^{h/d}) \cdot len$	-
$\mathbf{T_{len}}$	$2^{h/d}$	$d \cdot (2^{h/d})$	d

The key generation process is influenced by h, d, w, and len parameters. For a fixed value of h, the larger d, the faster is the key generation. A smaller value for the Winternitz parameter results in a faster key generation.

The signing and verification processes are linearly dependent on the parameters k and t. So a FORS tree with a larger number of FTS trees and higher height leads to a slower signing scheme.

The security of SPHINCS$^+$ is measured statistically. Eq. (1) [Bernstein et al. 2017] gives the probability p that a new message digest selects FORS leaves that are covered by positions already revealed in q_s signatures. It represents the probability that a FORS leaf is hit γ times after $q_s = 2^h$ signatures in a SPHINCS$^+$ signature scheme with hypertree height equal to h, k FORS trees and t leaves in each FORS tree.

$$p = \sum_{\gamma} \left(1 - \left(1 - \frac{1}{t}\right)^{\gamma}\right)^{k} \binom{q_s}{\gamma} \left(1 - \frac{1}{2^h}\right)^{q_s - \gamma} \left(\frac{1}{2^h}\right)^{\gamma} \tag{1}$$

The security level s in bits, against quantum computers, for a SPHINCS$^+$ signature scheme is given by eq. (2) [Bernstein et al. 2017].

$$s = -\frac{1}{2}\log_2 \left(\frac{1}{2^{8n}} + \sum_{\gamma} \left(1 - \left(1 - \frac{1}{t}\right)^{\gamma}\right)^{k} \binom{q_s}{\gamma} \left(1 - \frac{1}{2^h}\right)^{q_s - \gamma} \left(\frac{1}{2^h}\right)^{\gamma}\right) \tag{2}$$

As the parameters h, k and b are the ones that influence on the security level, it is necessary to determine, by parameter-space exploration, the sets of parameters that produce the desired security level.

We work on finding sets of parameters that produce smaller SPHINCS$^+$ signatures, ideal for blockchain operation. The initial approach is made by reducing the maximum number of signatures, as we believe a blockchain user will not need the total number of 2^{64} signatures initially designed to be performed by SPHINCS$^+$, and a signature scheme with the capacity of performing at around 2^{20} to 2^{30} signatures is more than enough for a user in a blockchain environment. The next section examines specific sets of parameters that decrease signature size while maintaining at least a 128-bit security level against quantum computers.

5. Results of parameter-space exploration

In this section, we present the results for finding smaller-sized signatures. The results were obtained by using a script provided by SPHINCS$^+$ developers [SPHINCS$^+$ script 2020]. Since the number of FORS trees and their heights have a probabilistic influence on the SPHINCS$^+$ security, we analyze how these two parameters impact on the SPHINCS$^+$ performance. This analysis was performed by fixing the number of FORS trees and varying their height in subsection 5.1 and by fixing the FORS trees height and varying their number in subsection 5.2.

5.1. Influence of the FORS trees height over security, signature size and signature creation time

Next we present the results for security, signature size and signature creation time for a SPHINCS$^+$ signature scheme with security parameter n = 32, hypertree height h = 24 (as 2^{24} possible signatures seems to be enough for blockchains), number of layers d = 4 and Winternitz parameter w = 256. We fixed the number of FORS trees and varied their height.

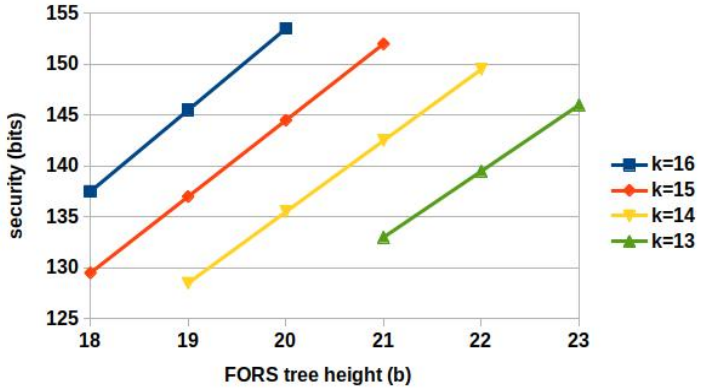

Figure 4. Influence of the FORS tree height over the security for fixed numbers of FORS trees.

Figure 4 shows that the security level is linearly dependent on the FORS trees height and it is possible to achieve the same security level for different numbers of FORS trees.

Figure 5 demonstrates that, for a fixed number of the FORS trees, the signature size increases with the increase of the FORS tree height and, for the same FORS tree height, schemes with more FORS trees will have larger signatures.

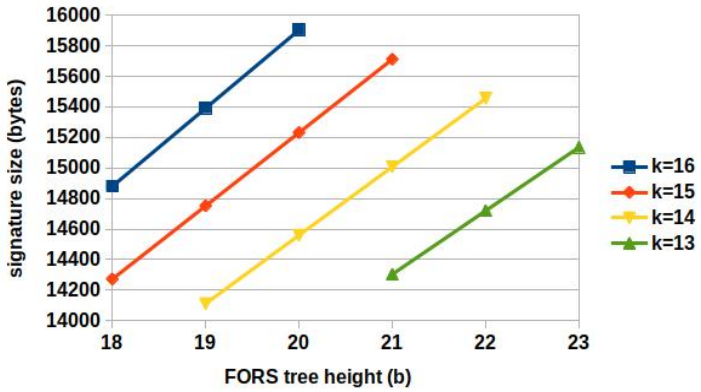

Figure 5. Influence of the FORS tree height over the signature size for fixed numbers of FORS trees.

Figure 6 illustrates the signature creation time, measured in number of hash function calls. It increases exponentially with the increase in the FORS tree height.

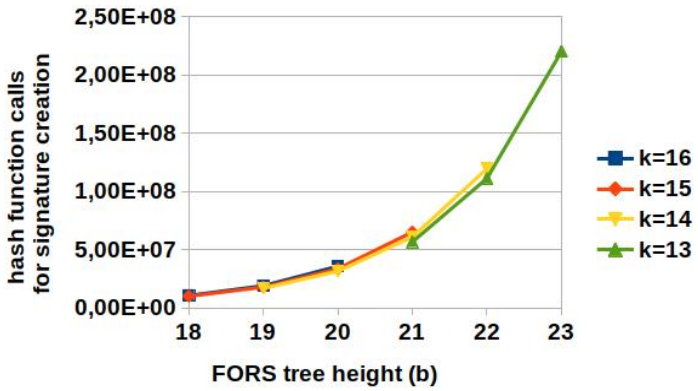

Figure 6. Influence of the FORS tree height over the signature creation time for fixed numbers of FORS trees.

From Figures 4 to 6, we observe that the biggest impact caused by varying the FORS tree height is on the signature creation time. Besides, it is possible to realize that signature schemes with a smaller FORS tree height are faster at generating a signature, despite having a larger number of FORS trees.

5.2. Influence of the number of FORS trees over security, signature size and signature creation time

Next we present the results for security, signature size and signature creation time for a SPHINCS$^+$ signature scheme with security parameter n = 32, hypertree height h = 24, number of layers d = 4 and Winternitz parameter w = 256. We fixed the FORS tree height and varied its number.

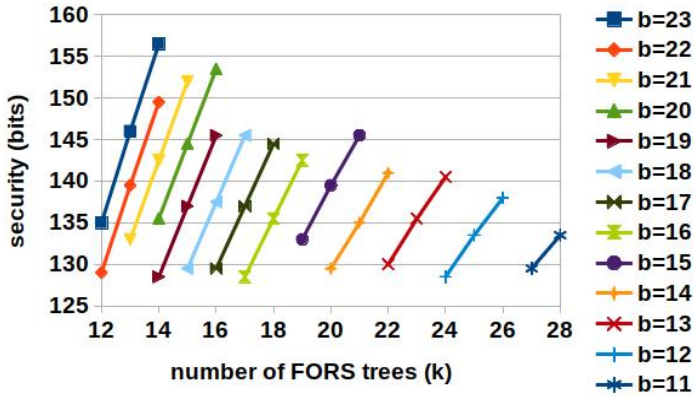

Figure 7. Influence of the number of FORS trees over the security for a fixed value of FORS tree height.

Figure 7 shows that, for the data interval analyzed, the security level increases with the number of FORS trees, for a fixed value of FORS tree height. It is possible to find signature schemes with similar values for security that are built using different numbers of FORS trees.

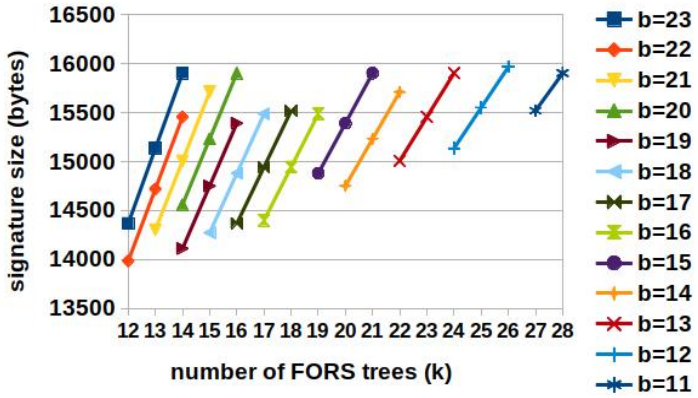

Figure 8. Influence of the number of FORS trees over the signature size for a fixed value of FORS tree height.

Figure 8 illustrates that the signature size, for a fixed value of FORS tree height, is linearly dependent on the number of FORS trees.

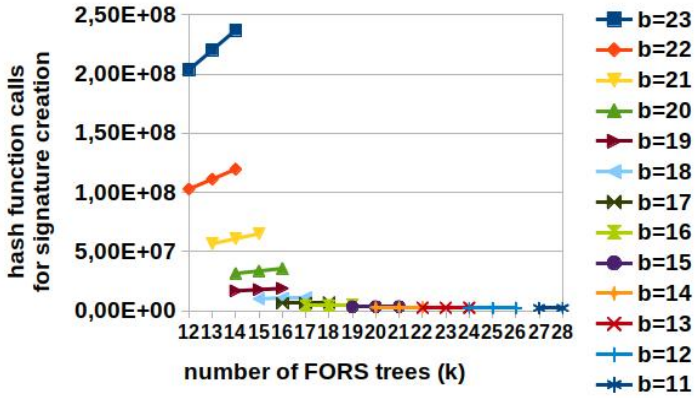

Figure 9. Influence of the number of FORS trees over the signature creation time for a fixed value of FORS tree height.

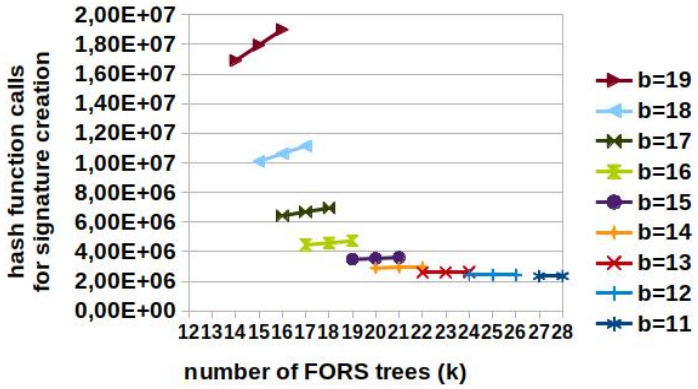

Figure 10. Zoom over Figure 9

Figure 9 demonstrates that the FORS tree height has more influence than the number of FORS trees over the signature creation time, as for the same number of FORS trees, the signature scheme with larger FORS tree height will demand more hash function calls. Figure 10 provides a zoom over the data values that are not readable in Figure 9.

The parameter-space exploration showed that it is possible to halve the signature size of the original SPHINCS$^+$ signature scheme (SPHINCS$^+$-256s), keeping a hypertree capable of performing around 16 million (2^{24}) signatures.

Table 4 compares the obtained signature schemes, sig1, sig2 and sig3, with the scheme that provides the shortest signature for a security level of 128 bits against quantum computers, SPHINCS$^+$-256s, showing that our attempt to adapt SPHINCS$^+$ for blockchain usage was able to halve its signature size. The sets of parameters called sig1, sig2 and sig3 were chosen among several others, as they are the ones with most interesting signature characteristics (small sizes and short times).

Table 4. Comparison for signature size between the original SPHINCS$^+$-256s and the obtained signature schemes.

signature scheme	signature size (bytes)	n	h	d	b	k	w
SPHINCS$^+$-256s	29,792	32	64	8	14	22	16
sig1	14,400	32	24	4	16	17	256
sig2	18,624	32	24	4	16	17	16
sig3	14,112	32	24	4	19	14	256

Table 5 compares the signature schemes as to the signature creation time, measured in hash function calls, showing that, even using a smaller value for hypertree height, the new signature schemes (sig1 and sig3) have bigger values for signature creation time.

Table 5. Comparison for signature creation time between the original SPHINCS$^+$-256s and the obtained signature schemes.

signature scheme	signature creation time	n	h	d	b	k	w
SPHINCS$^+$-256s	2,918,400	32	64	8	14	22	16
sig1	4,456,704	32	24	4	16	17	256
sig2	2,502,912	32	24	4	16	17	16
sig3	16,908,544	32	24	4	19	14	256

As our primary intention was to fit SPHINCS$^+$ signatures to blockchain usage, where signature creation time and size are key points, the signature scheme sig2 is a good candidate to be used as substitute to the original SPHINCS$^+$-256s.

6. Conclusions and Future work

Among the quantum-resistant signing algorithms, SPHINCS$^+$ distinguishes for being stateless and providing a security level of 128 bits against quantum computers. Nonetheless, compared to RSA and ECDSA, its signature size is an obstacle when used in an environment such as a blockchain, where the signature and public key sizes are fundamental. Hence, this work provides a study on adapting SPHINCS$^+$ to blockchain usage.

We showed that it is possible to halve the SPHINCS$^+$ signature size, keeping a security level of 128 bits, at the expense of reducing the SPHINCS$^+$ hypertree height, and the total number of possible signatures. Furthermore, this study revealed that by reducing the hypertree height of the original SPHINCS$^+$, it is possible to obtain a digital signature scheme with a smaller signature size and shorter signature creation time, more suitable to blockchain usage.

In future work, we intend to deepen the analysis of SPHINCS$^+$ to propose a new form to build an FTS tree. In addition, we want to analyze some of the digital signature schemes that passed to the NIST third round Post-Quantum Cryptography Standardization Process.

References

Aumasson, J.-P. and Endignoux, G. (2017). Gravity-SPHINCS. Submission to the NIST Post-Quantum Cryptography Standardization Project (2017).

Bernstein, D. J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.-L., Hülsing, A., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M. M., et al. (2017). Sphincs+– submission to the nist post-quantum project. *Submission to NIST*.

Bernstein, D. J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., and Wilcox-O'Hearn, Z. (2015). SPHINCS: practical stateless hash-based signatures. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 368–397. Springer.

Bernstein, D. J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., and Schwabe, P. (2019). The sphincs+ signature framework. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 2129–2146.

Buchmann, J., Dahmen, E., and Hülsing, A. (2011). XMSS - A practical forward secure signature scheme based on minimal security assumptions. In *International Workshop on Post-Quantum Cryptography*, pages 117–129. Springer.

de Lucena, A. U. and Henriques, M. A. A. (2018). Estudo preliminar da adoção de assinaturas baseadas em hash no blockchain do bitcoin. In *Anais Principais do XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 65–72, Porto Alegre, RS, Brasil. SBC.

Ding, J. and Schmidt, D. (2005). Rainbow, a new multivariable polynomial signature scheme. In *International Conference on Applied Cryptography and Network Security*, pages 164–175. Springer.

Fernández-Caramès, T. M. and Fraga-Lamas, P. (2020). Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks. *IEEE Access*, 8:21091–21116.

GEMSS (2020). GeMSS NIST Call Specification. Accessed: Jul. 12, 2020. [Online]. Available: https://www-polsys.lip6.fr/Links/NIST/GeMSS_specification.pdf.

Gidney, C. and Ekerå, M. (2019). How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *arXiv preprint arXiv:1905.09749*.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219.

Hülsing, A. (2013). W-ots+–shorter signatures for hash-based signature schemes. In *International Conference on Cryptology in Africa*, pages 173–188. Springer.

IOTA (2020). IOTA. Accessed: Jul. 10, 2020. [Online]. Available: https://www.docs.iota.org.

Lamport, L. (1979). Constructing digital signatures from a one-way function. Technical report, Technical Report CSL-98, SRI International.

LUOV (2020). LUOV Official GitHub Repository. Accessed: Jul. 12, 2020. [Online]. Available: https://github.com/WardBeullens/LUOV.

Merkle, R. C. (1979). *Secrecy, authentication, and public key systems.* Stanford University.

Nakamoto, S. (2009). Bitcoin: A peer-to-peer electronic cash system. Accessed: Nov. 2, 2020. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

Nam, Y. and Maslov, D. (2019). Low-cost quantum circuits for classically intractable instances of the hamiltonian dynamics simulation problem. *npj Quantum Information*, 5(1):1–8.

NIST (2020a). NIST Post-Quantum Cryptography Standardization. Accessed: Jul. 12, 2020. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography.

NIST (2020b). PQC Standardization Process: Third Round Candidate Announcement. Accessed: Aug. 5, 2020. [Online]. Available: https://csrc.nist.gov/News/2020/pqc-third-round-candidate-announcement.

PICNIC (2020). PICNIC. Accessed: Jul. 12, 2020. [Online]. Available: https://microsoft.github.io/Picnic/.

Proos, J. and Zalka, C. (2003). Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information & Computation*, 3(4):317–344.

QRL (2020). QRL - The Quantum Resistant Ledger. Accessed: Jul. 12, 2020. [Online]. Available: https://theqrl.org/.

Reyzin, L. and Reyzin, N. (2002). Better than biba: Short one-time signatures with fast signing and verifying. In *Australasian Conference on Information Security and Privacy*, pages 144–153. Springer.

Shor, P. W. (1994). Proceedings of the 35th annual symposium on foundations of computer science. *IEE Computer society press, Santa Fe, NM*.

SPHINCS$^+$ script (2020). SPHINCS+ parameter exploration SAGE script. Accessed: Jul. 17, 2020. [Online]. Available: https://sphincs.org/data/spx_parameter_exploration.sage.