MASTER'S THESIS PROPOSAL

Determining SPHINCS+ Readiness for Standardization of SLH-DSA Signature

Student:
Jessica Ancillotti
jna1172@rit.edu

Co-Chairs
Professor Stanisław Radziszowski
Professor T.J. Borrelli

Department of Computer Science
March 31, 2025

Contents

1 Abstract

As quantum computing advances, traditional cryptographic algorithms risk becoming obsolete, necessitating the development and implementation of quantum-resistant alternatives [11]. This research evaluates SPHINCS+, a hash-based signature scheme recently standardized by NIST, to determine its readiness for widespread use. SPHINCS+, standardized under SLH-DSA FIPS 205 [13], is examined within the framework of established and emerging cryptographic standards to evaluate its applicability across diverse ecosystems. This analysis emphasizes SPHINCS+'s robustness, efficiency, and scalability, highlighting its potential as a direct replacement for existing signature schemes. Additionally, the research explores hybrid cryptographic solutions that integrate classical and quantum-resistant methods, ensuring a seamless transition to secure critical systems against evolving threats.

2 Introduction

In recent years, quantum computing has garnered significant research interest due to its potential to tackle complex mathematical problems that are infeasible for classical computers to solve at scale. These advancements, while promising for innovation, pose a serious threat to modern cryptographic systems. Should large-scale, fault-tolerant quantum computers become a reality, foundational cryptographic protocols such as RSA, Diffie-Hellman, and Elliptic Curve Cryptography would be rendered insecure, necessitating the urgent development of quantum-resistant alternatives [10]. In response to this potential threat, NIST has proactively launched an initiative to identify and standardize quantum-resistant public-key cryptographic algorithms [11].

SPHINCS+ is a stateless hash-based signature framework, allowing it to be a drop-in replacement for current signature schemes [12]. SPHINCS+ was submitted to the NIST search in 2017 and has since become standardized. SPHINCS+ has made several contributions, including tweakable hash functions and the development of the few-time signature scheme, Forest of Random Subsets (FORS). NIST also selected CRYSTALS-Kyber (ML-KEM) as the public-key encapsulation mechanism and two other digital signature schemes for standardization: CRYSTALS-Dilithium (ML-DSA) and Falcon (FN-DSA). Among these, SPHINCS+ stands out as the only scheme not based on the computational hardness of problems involving structured lattices [1].

3 Thesis Overview

This thesis, titled "Determining SPHINCS+ Readiness for Standardization," investigates the viability of SPHINCS+ as a post-quantum cryptographic standard in light of its recent endorsement by NIST [15]. The study is motivated by the looming threat posed by quantum computing to classical cryptographic systems, necessitating the development and adoption of quantum-resistant solutions. SPHINCS+ represents a hash-based, stateless signature scheme, designed to provide long-term security and resilience against quantum attacks.

The research is structured around four primary objectives. First, it compares SPHINCS+ against other NIST post-quantum cryptography finalists, analyzing its ease of integration, compatibility with existing systems, and practicality for broad adoption. Second, the thesis explores hybrid cryptographic schemes, which combine classical and quantum-resistant algorithms, as a transitional solution to ensure robust security during the shift to post-quantum systems.

Building on this analysis, the third objective focuses on evaluating SPHINCS+ within blockchain systems, specifically as a potential replacement for traditional signature schemes like ECDSA in Bitcoin. This segment assesses the performance, scalability, and compatibility of SPHINCS+ in critical blockchain protocols, positioning it as a candidate for securing future blockchain infrastructures.

Finally, if time permits, the thesis will address the challenges posed by SPHINCS+ in constrained environments. Using insights from previous research on post-quantum authentication in vehicle-to-vehicle (V2V) communications [16], the thesis examines whether similar limitations affect SPHINCS+ in other critical applications, highlighting areas where alternative solutions may be necessary.

4 Objectives

4.1 Comparative Analysis of SPHINCS+

Compare SPHINCS+ with other NIST post-quantum cryptography finalists, focusing on ease of integration, compatibility with existing systems, and overall practicality for standardization.

4.2 Hybrid Cryptographic Schemes

To assess the potential benefits of hybrid schemes in transitional periods where both classical and post-quantum cryptographic solutions are necessary for secure systems.

4.3 SPHINCS+ in Blockchain Systems

Building upon the assessment of hybrid schemes, this objective narrows in on evaluating SPHINCS+ as a candidate to replace traditional cryptographic algorithms (e.g., ECDSA in Bitcoin) within blockchain systems. As blockchains are increasingly viewed as critical infrastructures, transitioning to quantum-safe algorithms like SPHINCS+ is essential to protect against future quantum threats. This objective will determine whether SPHINCS+ offers a feasible alternative to current standards regarding performance, scalability, and compatibility with blockchain protocols.

4.4 Bandwidth and Latency Constraint

The paper "When Cryptography Needs a Hand: Practical Post-Quantum Authentication for V2V Communications" [16] provides valuable insights into the limitations of SPHINCS+ for use in bandwidth-limited, real-time applications. Specifically, it concludes that SPHINCS+ is unsuitable for Vehicle-to-Vehicle (V2V) communications due to its large signature sizes and longer processing times, incompatible with the stringent latency and spectrum constraints of V2V environments. If time permits, this finding prompts further investigation into whether SPHINCS+ may face similar challenges in other critical areas requiring post-quantum cryptographic (PQC) solutions.

5 Methodology

5.1 Comparative Analysis of SPHINCS+ and NIST PQC Finalists

Assesses SPHINCS+ against other NIST PQC finalists by comparing security properties and key performance metrics. Implement SPHINCS+, Dilithium, and Falcon in a controlled test environment for consistent comparisons. Record and compare the signature size of each algorithm, the time each algorithm takes to sign and verify messages and track CPU and memory usage during signing and verification to gauge each algorithm's computational demands

5.2 Development and Evaluation of SPHINCS+ Hybrid Cryptographic Schemes

Conduct a literature review to identify existing hybrid schemes and their implementation challenges. This review will focus on case studies and theoretical models that have evaluated hybrid systems in practice. From here, develop a SPHINCS+ hybrid cryptographic schemes by pairing it with a classical algorithm. Evaluate this scheme by measuring the signature size, signing time, and verification time and evaluate the security provided by the hybrid scheme, ensuring that it meets security standards for protection against quantum attacks.

5.3 Applicability of SPHINCS+ in Blockchain Systems

Conduct a literature review of the existing research on post-quantum cryptographic algorithms and their applicability in blockchain. A key focus will be on measuring the impact of SPHINCS+ signature sizes, which are significantly larger than those of traditional algorithms. The study will analyze how these larger signatures affect transaction sizes, potentially leading to increased bandwidth requirements for data transmission within blockchain networks.

5.4 Feasibility of SPHINCS+

Examine whether SPHINCS+ is feasible for use within the Medical Internet of Things (MIoT). MIoT often operates on resource-constrained devices where security and efficiency are paramount. By systematically evaluating SPHINCS+ in these areas, I aim to determine if its limitations in V2V environments extend to other sectors or if it can meet their specific requirements.

A comprehensive literature review to establish the operational requirements and constraints for cryptographic algorithms within V2V and MIoT. Similar to the paper mentioned above, I will test SPHINCS+ to evaluate key metrics such as signing time, verification time, and memory consumption. This test will assess SPHINCS+ suitability for secure yet lightweight performance in IoT environments. Thirdly, MIoT brings on the concern of data privacy, I will determine SPHINCS+ for compatibility with data protection standards like HIPAA.

6 Literature Review

6.1 History

6.1.1 SPHINCS

Originally, SPHINCS [4] was designed by Bernstein, Hopwood, Hulsing, Lange, Niederhagen, Papachristodoulou, Schneider, Schwabe, and Wilcox-O'Hearn as a stateless hash-based signature scheme. In the original design of SPHINCS there are some components that SPHINCS+ tries to improve upon. SPHINCS uses HORS/HORTS to sign messages, whereas SPHINCS+ uses FORS. Figure 4 illustrates the difference between HORS and FORS, which will be addressed later in this paper. SPHINCS uses Winternitz One-Time Signature, WOTS, which SPHINCS+ further develops in WOTS+. WOTS+ introduces tweakable hash functions. SPHINCS+ also introduces verifiable index selection[8]. Part of this comes from the change of HORST/HORS to Forest of Random Subsets (FORS). In HORST, the index representing the keypair for signing the message was generated pseudo-randomly. Due to an inability to verify this index, someone, say Oscar, could easily attack by using one hashing computation on the HORST instances and exploit this. The index is now computed using the message digest which is hashed with the public key and the pseudo-randomly generated R value. All of these together are connected directly to the FORS instance and only that instance. This helps enhance the security of the scheme.

6.1.2 Digital Signatures

Digital Signatures provide a way for a receiver to verify that a message being sent to them is from the sender. This is done by digitally signing the message before it is sent to the receiver. Bob would create their private and public keys, sign the message using the private key, and then send the signed message and the signature to Alice. Alice is then able to verify the message using Bob's public key and can confirm if the validity of the signature. Digital Signature Algorithm (DSA) is one popular method that provides message integrity, authentication, and non-repudiation[10]. DSA utilizes a hashing function component to create a message digest that is signed with SHA. There are some important properties needed when using a hash function in a cryptographic scheme.

6.1.3 Properties of Hash-Based Cryptographic Functions

Hash Based Cryptographic Functions have certain requirements they must meet in order to meet the NIST standards. They must be:
1) Collision Resistant: Two different messages will not have the same message digest.
2) Preimage Resistant: For any randomly chosen message digest it is not computationally feasible to find a preimage of the digest.
3) Second Preimage Resistant: It is not computationally possible to find input that produces the same output as another input. [6]

6.2 Key Components of SPHINCS+

The components that make up SPHINCS+ are WOTS+, Extended Merkle Signature Scheme (XMSS), Hypertrees, and Forest of Random Subsets (FORS).

6.2.1 Winternitz One-Time Signature Plus (WOTS+)

Winternitz One-Time Signature Plus, WOTS+, is a one-time signature scheme that has two main parameters, n, w. WOTS+ maintains the core ideas of WOTS, using a certain number of function chains that start from a random input, a secret key, and the end of chains are the public key. [3]

6.2.2 WOTS+ Construction

The first parameter is n, the length of the message in bytes. The second parameter is w, which is the Winternitz parameter and an element of the set $\{4, 16, 256\}$ [2]. The w parameter is limited to the set of $\{4, 16, 256\}$ because it was found that these values yield optimal trade-offs and are easy to implement. However, in the official submission, I would like to note that the parameters only use 4 for w and fail to mention the other elements for the rest of their submission.

The n parameter also includes the length of the private key, public key, and signature element in bytes. It determines the length of the message that WOTS+ can process and sign. The parameters that are recommended for n are shown in Fig 5. These two parameters are then used to compute the len, or number of n-byte string elements in the WOTS+ private key, public key and signature(1,2) [3]

$$len_1 = [8n/log(w)] \tag{1}$$

$$len_2 = [log_2(len_1 * (w-1))/log(w)] + 1 \tag{2}$$

The WOTS+ construction is broken down into the chaining function, public key generation, signature generation, and verification using the signature. Private key generation happens within the chaining functions and public key generation and is not discussed much outside of these.
The chain function takes an $n-byte$ string, X, number of steps, s, and a starting index,i, as input. It also takes in ADRS as input, which is a 32-byte hash function address that can identify the position of the hash function call, aka the value being computed [3]. ADRS is used throughout all of the algorithms that make up SPHINCS+ and is meant as a way to keep track of where you are within all the trees. Lastly, the chain function takes in $PK.seed$, which is a public seed that was added later into the NIST submission as an input to help mitigate multi-key attacks and assist in domain separation between different key pairs [12]. This algorithm for the chaining function will produce the value of F iterated s times on X. F is the tweakable hash function that takes an n-byte message as input and produces an n-byte output. [12] Referring to the example in Fig 1, this is what the chaining function is trying to replicate, by making the hash chains. The chaining function will be used in the other algorithms for WOTS+.

Public Key Generation in WOTS+ will take in $Sk.seed$, $PK.seed$, and $ADRS$ and outputs a compressed version of the WOTS+ public key, pk. The algorithm will also compute the secret value and public value. For the generation of the public value, it will generate a corresponding secret value and use the chaining algorithm from above to compute the end value of the chain of length w[12]. Once these are computed they will be compressed into a single n-byte value [14]. This compression is a bit different as the last nodes of the WOTS+ chains are not compressed using an L-tree but using the single tweakable hash function call. This function call, like the ones before it, receives an address and a public seed to key this call and to generate a bitmask as long as the input [8].

WOTS Signature Generation will take in the message, M, $SK.seed$, $PK.seed$ and $ADRS$ and return the WOTS+ signature. This operates by converting the n-byte message to $base_w$ and computing the checksum over M and appended to the $base_w$ value found. The $base_w$ is essentially integers that belong to the set {0 to $w-1$} The selected nodes are then concatenated to form the signature. Computing the checksum is just a basic checksum evaluation with the addition of w, eq(3,4). The checksum is important to help avoid forgery as stated earlier is an issue with basic Lamport.

$$for(i = 0; i < len_1; i++)\{ \tag{3}$$

$$csum = csum + w - 1 - msg[i]; \} \tag{4}$$

Verifying a WOTS+ signature can be done by computing a public key value from the signature. This is similar to the public key generation steps. The verifier would need to recompute the checksum, derive the chain lengths, and apply the hash function F to complete each chain to its full length[3]. This will then return the public key which can be checked against the known public key. However, since WOTS+ is not just used on its own, the output value will be used in the grand scheme to verify the SPHINCS+ public key.

6.2.3 Merkle Signature Scheme (MSS) and More

Extended Merkle Trees, XMSS, are an important part of SPHINCS+. Merkle Signature Schemes are a useful method of extending a one-time scheme so it can be used with a larger number of signatures without increasing the public key size [9]. The idea behind the MSS is to create a binary tree, aka Merkle Tree, by hashing combinations of various public keys of OTS schemes. The MSS purpose is to authenticate public keys, while WOTS+ is used for creating signatures.

6.2.4 Extended Merkle Trees (XMSS)

XMSS is just an extension of the basic Merkle Signature Scheme. XMSS helps sign a larger number of messages. XMSS contains a few components, WOTS, PRF, which is a pseudorandom function that can generate a randomizer called R, which is then used for the hashing of the message to be signed, and also part of the tweakable hash function. XMSS also contains H, part of the tweakable hash function, and H_{msg} which is used to generate the digest of the message. Each of the public and private key pairs are connected with a perfect binary tree [7].

The XMSS signature consists of the *height + the length ∗n bytes*, WOTS+ signature and the authentication path. The authentication path is an array of nodes from each level that allows the verifier to compute the root of the tree when combined with the WOTS+ public key[12]. The authentication path is similar to that in Fig 2's example using sibling nodes. The sibling nodes will be those that are siblings to the nodes on the path from the WOTS+ key used up to the root. Once the authentication path is created, the n-byte message M is signed with the corresponding WOT+ key[14].

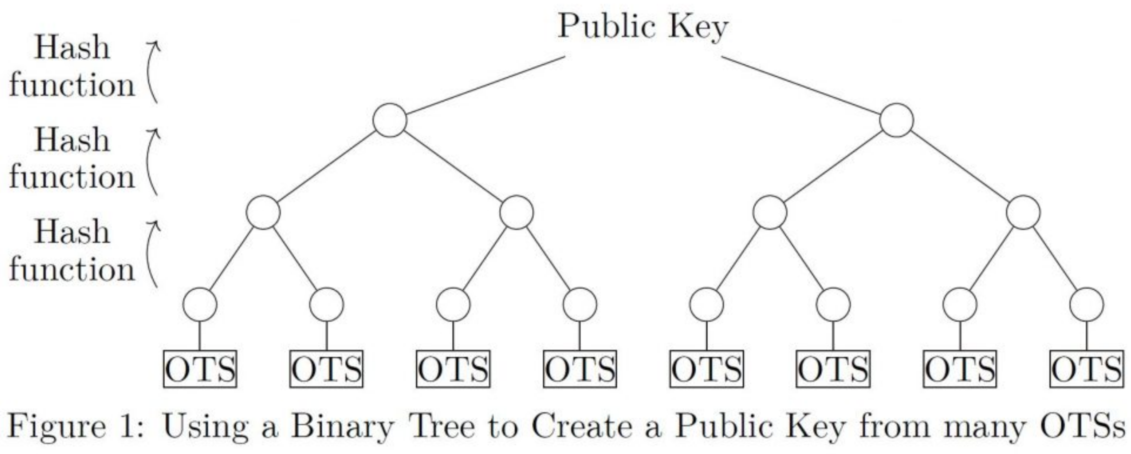

Figure 1: Merkle Tree with OTS [2]

6.2.5 Connection to WOTS

In the example in Fig 2, h is the hash function, that is collision resistant. The hash function for us is part of the tweakable hash functions mentioned. To compute a parent node, you compute the hash of the two children nodes using a version of the tweak. The leaf nodes of XMSS are the WOTS+ public key. We can see it better by referring to Figure 3. Each leaf node is the hash output of the WOTS+ public key[12]. Every other node in the tree is the hash of the two child nodes, to create the parent node. The root node is the public key.

6.2.6 Forest of Random Subsets (FORS)

An improvement to SPHINCS is the addition of a Forest of Random Subsets, FORS. In the original SPHINCS, you would randomly pick one of the HORST trees to sign your message, which created better chances of forgery. However, with SPHINCS+, the mapping of the message is connected to both FORS leaves and the FORS signature. This gives less of a chance for someone, say, Oscar, to forge a message.
FORS is a few-time signature that can sign the digests of the actual message[12]. It's called a few times because it can sign a message a few times, but each time, information is exposed, which reduces the security of the key being used. FORS construction is similar to HORS/HORST, however unlike HORS/HORST where there is just one tree constructed from the message digest, FORS has several, where each tree has a root that is all hashed to create the FORS public key, see Fig 4, [5]

Some key parameters for FORS are k, the number of trees, a, the height of k trees, and t, $2a$. Each k set is formed into a Merkle tree, and as stated earlier, the roots of these trees are hashed together to form the FORS public key. This can be seen in Fig 4 and even in 5, which will be broken down in the next section.

In the generation of the FORS part of the Merkle Hash Tree, the FORS public values and signature will be generated. This is similar to the construction of the hypertree with WOTS nodes, except this time the leaf nodes are the FORS secret values instead of the WOTS+ public keys [12].

The algorithm works by taking in the input as $SK.seed$ $PK.seed$, the target node at index i, the target node height, z, and the $ADRS$. Each node in this tree is the root of

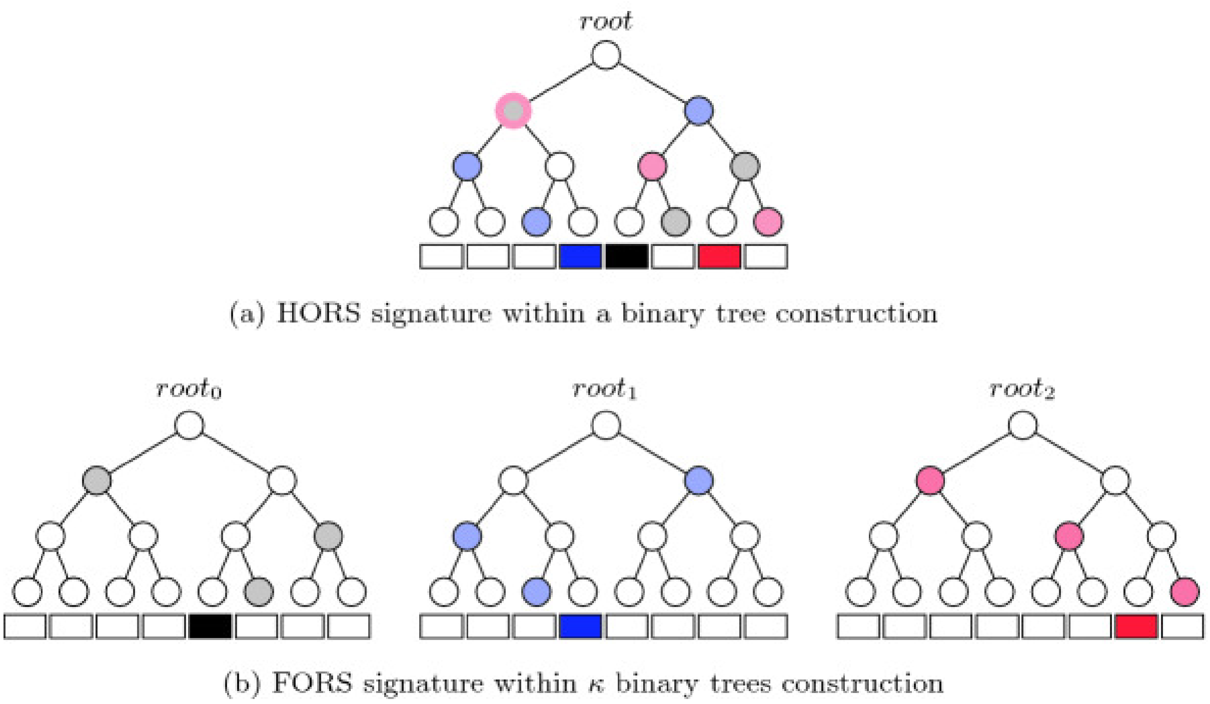
root
(a) HORS signature within a binary tree construction
$root_0$
$root_1$
$root_2$
(b) FORS signature within κ binary trees construction

Figure 2: HORS vs FORS Trees [5]

a subtree, and this follows normal tree creation algorithms by computing right and left nodes and hashing them together to create the parent node. The lead nodes will return the secret key [12]. Again, this looks similar to Fig. 4 FORS tree.

The creation of a FORS signature, again is similar to other signature creations. This time it will sign the message digest, which is generated using the Randomizer function, R. This algorithm will take as input, $SK.seed$ $PK.seed$, $ADRS$, and the message digest. Similarly to Fig 4, the message digest is split into k a bit strings, and similar to the WOTS+ $byte_w$, the integers are in this case between 0 and $t-1$ Each of these integers will then be mapped to a FORS private key, and then the authentication path is computed and the message is signed [3].

To verify a FORS signature, the verifer would need to compute the public key using the message digest and the signature. This is done similarly to WOTS+ verification. The FORS public key is always held within the bottom layer of the tree, layer 0, The verifier would need to compute the roots of each Merkle Tree. The authentication path that was generated in the signing process will be used to do this, along with the hash values of the private keys. Once the verifier has computed the roots, they can hash all the roots together to obtain the FORS public key [12].

7 Time Plan

This will include a weekly meeting with co-chairs as well.

Week	Task
11/3	Proposal Draft
11/10	Proposal Draft
11/17	Proposal Draft
11/24	Proposal Draft
12/1	Proposal Draft
12/8	Proposal Draft
12/15	Proposal Draft
12/22	Obj. One: Comparative Analysis of SPHINCS+ Research
12/29	Obj. One: Comparative Analysis of SPHINCS+ Research
1/5	Obj. One: Comparative Analysis of SPHINCS+ Research
1/12	Obj. One: Comparative Analysis of SPHINCS+ Research
1/19	Obj. Two: Hybrid Cryptographic Schemes Research
1/26	Obj. Two: Hybrid Cryptographic Schemes Research
2/2	Obj. Two: Hybrid Cryptographic Schemes Research
2/9	Obj. Two: Hybrid Cryptographic Schemes Research
2/16	Obj. Three: SPHINCS+ in Blockchain Systems Research
2/23	Obj. Three: SPHINCS+ in Blockchain Systems Research
3/2	Obj. Three: SPHINCS+ in Blockchain Systems Research
3/9	Obj. Three: SPHINCS+ in Blockchain Systems Research
3/16	Obj. Four: Bandwidth and Latency Constraint Research
3/23	Obj. Four: Bandwidth and Latency Constraint Research
3/30	Obj. Four: Bandwidth and Latency Constraint Research
4/6	Obj. Four: Bandwidth and Latency Constraint Testing
4/13	Obj. Four: Bandwidth and Latency Constraint Testing
4/20	Finalization on objectives implementations
4/27	Finalization on presentation and paper
5/5	Defense

References

[1] Gorjan Alagic et al. *Status Report on the First Round of the Additional Digital Signature Schemes for the NIST Post-Quantum Cryptography Standardization Process.* Tech. rep. NIST IR 8528. Gaithersburg, MD: National Institute of Standards and Technology, 2024. DOI: `10.6028/NIST.IR.8528`. URL: `https://doi.org/10.6028/NIST.IR.8528`.

[2] Jean-Philippe Aumasson et al. *SPHINCS+*. Submission to NIST's post-quantum crypto standardization project. 2022. URL: `https://sphincs.org/data/sphincs+-r3.1-specification.pdf`.

[3] D. J. Bernstein et al. "The SPHINCS+ Signature Framework". In: *2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19).* ACM, 2019. DOI: `10.1145/3319535.3363229`.

[4] D.J. Bernstein et al. "SPHINCS: Practical Stateless Hash-Based Signatures". In: *Advances in Cryptology – EUROCRYPT 2015.* Ed. by E. Oswald and M. Fischlin. Vol. 9056. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2015, pp. 368–397. DOI: `10.1007/978-3-662-46800-5_15`.

[5] ISARA Corporation. *Math paths to quantum-safe security: Hash-based cryptography.* 2020. URL: `https://www.isara.com/blog-posts/hash-based-cryptography.html`.

[6] Q. Dang. "Recommendation for Applications Using Approved Hash Algorithms". In: *National Institute of Standards and Technology* (2012).

[7] Andreas Huelsing et al. "RFC 8391: XMSS: Extended Merkle Signature Scheme". In: *IETF Datatracker* (2018). URL: `https://datatracker.ietf.org/doc/html/rfc8391`.

[8] Andreas Huelsing. *SPHINCS+ – The smaller SPHINCS.* 2017. URL: `https://huelsing.net/wordpress/?p=558`.

[9] Mikhail Kudinov et al. *SPHINCS+C: Compressing SPHINCS+ With (Almost) No Cost.* Cryptology ePrint Archive, Paper 2022/778. 2022. URL: `https://eprint.iacr.org/2022/778`.

[10] Christof Paar and Jan Pelzl. *Understanding Cryptography: A Textbook for Students and Practitioners.* Springer, 2010.

[11] National Institute of Standards and Technology. *Call for Proposals for Additional Digital Signature Schemes for the NIST Post-Quantum Cryptography Standardization Process.* `https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf`. Accessed: 2024-12-09. 2022.

[12] National Institute of Standards and Technology. "Stateless Hash-Based Digital Signature Standard". In: *Federal Information Processing Standards Publication (FIPS) NIST FIPS 205 ipd* (2023). DOI: `10.6028/NIST.FIPS.205.ipd`.

[13] National Institute of Standards and Technology. *Stateless Hash-Based Digital Signature Standard.* Tech. rep. Federal Information Processing Standards Publication (FIPS) NIST FIPS 205. Washington, D.C.: Department of Commerce, 2024. DOI: `10.6028/NIST.FIPS.205`. URL: `https://doi.org/10.6028/NIST.FIPS.205`.

[14] D. Stinson and M. Paterson. *Cryptography: Theory and Practice.* CRC Press, 2018. URL: `https://books.google.com/books?id=nHxqDwAAQBAJ`.

[15] The NIST PQC Team. *PQC standardization process: Announcing four candidates to be standardized, plus fourth round candidates.* 2022. URL: `https://www.nist.gov/news-events/news/2022/07/pqc-standardization-process-announcing-four-candidates-be-standardized-plus`.

[16] Geoff Twardokus et al. "When Cryptography Needs a Hand: Practical Post-Quantum Authentication for V2V Communications". In: *Network and Distributed System Security (NDSS) Symposium 2024.* San Diego, CA, USA, 2024. DOI: `10.14722/ndss.2024.24267`.