Hash-based Cryptography

Fernando Javier Lopez Cerezo

1 Overview

Traditional digital signature algorithms, such as RSA, DSA, and ECDSA rely on the hardness of number-theoretic problems like integer factorization and discrete logarithms. These foundational assumptions, however, are rendered vulnerable in the face of quantum computing, making such schemes unsuitable for a post-quantum world.

In contrast, hash-based digital signature schemes present a promising alternative. Their security is grounded not in algebraic structures or number-theoretic problems, but in the well-understood properties of cryptographic hash functions (a core primitive also used in most other signature schemes). Because they depend solely on symmetric cryptographic assumptions, hash-based schemes exhibit strong resistance to quantum attacks.

Rompel's seminal result [19] further strengthens the case for hash-based approaches by showing that the existence of one-way functions is both necessary and sufficient for constructing secure digital signature schemes. This minimal assumption underscores the robustness and simplicity of hash-based designs. Moreover, the continual development of secure hash functions inherently leads to new, viable signature constructions, ensuring adaptability and forward security in a rapidly evolving cryptographic landscape.

This paper is structured as follows: Section 2 provides a brief overview of cryptographic hash functions and their security properties. Section 3 introduces the Lamport One-Time Signature scheme and Section 4 presents its improvements, including Winternitz OTS and WOTS+. Section 5 presents Merkle Trees as a means to aggregate one-time signatures into a scalable signature scheme. Section 6 discusses few-time signature schemes such as HORS and HORST. Section 7 describes the SPHINCS and SPHINCS+ schemes, highlighting their stateless design and use of hypertrees and FORS. Section 8 covers the standardization of SPHINCS+ as SLH-DSA by NIST. Finally, Section 9 concludes the paper with a discussion of the strengths and challenges of hash-based digital signatures in a post-quantum context.

2 Reminder of Hash Functions

A cryptographic hash function h is a deterministic algorithm that takes an input (or message) of arbitrary length and produces a fixed-size output, called a hash value or digest. It is designed to be computationally efficient and, unlike general-purpose hash functions, cryptographic hash functions are designed to satisfy specific security properties:

1. Preimage Resistance (One-Wayness): For a given hash value y, it is computationally infeasible to find any input x such that:

$$h(x) = y$$

2. Second Preimage Resistance: For a given input x_1, it is computationally infeasible to find a different input $x_2 \neq x_1$ such that:

$$h(x_1) = h(x_2)$$

3. Collision Resistance: It is computationally infeasible to find any two distinct inputs x_1 and x_2 such that:

$$h(x_1) = h(x_2)$$

Additionally, a cryptographic hash function should exhibit an avalanche effect (a small change in input drastically changes the output). The security of each of these properties is thought to be well understood both in the classical and quantum settings, as is shown in the following table by the best known attacks.

Security Property	**Classical Complexity**	**Quantum Complexity**	
		With qRAM	**Without qRAM**
One-wayness	$O(2^n)$	No speedup	$O(2^{n/2})$ [11]
Preimage Resistance	$O(2^n)$	No speedup	$O(2^{n/2})$ [11]
Collision Resistance	$O(2^{n/2})$[20]	$O(2^{n/3})$[6]	$\widetilde{O}(2^{2n/5})$ [10]

Table 1: Comparison of Classical and Quantum Complexity for Cryptographic Hash Function Security Properties.

It should be noted that it has been proven that Grover's speedup for one-wayness and preimage resistance is optimal [1]. Also, in terms of time-space complexity, the BHT algorithm is less efficient than the classical parallel rho method [2].

For the rest of this paper, let $f\colon \{0,1\}^n \to \{0,1\}^n$ denote a one-way function and $h : \{0,1\}^* \to \{0,1\}^m$ a cryptographic hash function. We will denote private key/values as sk, public key/values as pk and signatures as σ.

3 Lamport OTS

The Lamport One-Time Signature (OTS) scheme [14] was the first digital signature system based solely on the security of one-way functions. The signer generates a private key consisting of $2m$ random bitstrings $\text{sk}_i \in \{0,1\}^n$, where each message bit $b_i \in \{0,1\}$ is associated with a pair $(\text{sk}_{2i}, \text{sk}_{2i+1})$. The public key is defined as $\text{pk}_i = f(\text{sk}_i)$ for $i = 0, \ldots, 2m-1$. To sign a message digest $b_1, \ldots, b_m \in \{0,1\}$, the signature is the sequence $\sigma_i = \text{sk}_{2i+b_i}$ for $i = 0, \ldots, m-1$, disclosing one preimage per bit. Verification consists of computing $f(\sigma_i)$ and checking that it matches the corresponding pk_{2i+b_i}. While conceptually simple and secure under minimal assumptions, the scheme is highly inefficient, requiring $2mn$ bits of private key material to sign each m-bit digest and a fresh keypair for every message.

4 Winternitz OTS

The Winternitz[1] One-Time Signature (W-OTS) scheme [7] enhances the Lamport approach by enabling each private key element to sign multiple message bits using hash chains. The signer generates ℓ private key elements $\text{sk}_i \in \{0,1\}^n$, and computes the public key as $\text{pk}_i = f^{2^w-1}(\text{sk}_i)$, where the parameter w controls the tradeoff between signature size and computational effort. To sign a message M, its hash is divided into $\ell_1 = \lceil n/w \rceil$ chunks of w bits, with each chunk interpreted as an integer $m_i \in [0, 2^w - 1]$. The signature components are computed as $\sigma_i = f^{m_i}(\text{sk}_i)$. To prevent an attacker from modifying any chunk m_i to a larger value and forging signatures, a checksum $C = \sum_{i=1}^{\ell_1}(2^w - m_i)$ is appended to the message digest, ensuring that any increase in m_i must be offset by a decrease elsewhere and making the altered message require shorter hash chains than available in the signature. The checksum is encoded into $\ell_2 = \lceil (\log_2 \ell_1 + w)/w \rceil$ additional chunks, resulting in a total of $\ell = \ell_1 + \ell_2$ signature elements. Verification checks that $\text{pk}_i = f^{2^w-1-m'_i}(\sigma_i)$ for each i, where m'_i includes both message and checksum components.

Decades later, Hülsing independently published an upgrade WOTS+ that shorten the signatures size and increase the security of the WOTS scheme [12]. This variant introduces $r = (r_1, ..., r_{w-1})$ extra bitstrings (or a seed which pseudorandomly generates them) on the public key which act as bitmasks for f. Instead of applying $f(x)$ each time, we now define recursively $c^i(x, r) = f(c^{i-1}(x, r) \oplus r_i)$. The trick is that if f is second preimage resistant then c is collision resistant. Therefore WOTS+ reduces the signature size because you can use a hash function with shorter outputs than in WOTS at the same level of security or longer hash chains.

[1] Suggested by Winternitz in 1979 but never published.

5 Merkle Trees

In 1979 Merkle proposed a way to identify a fixed number of one time verification keys with a single public key using complete binary trees [15]. The signer selects a height $H \geq 2$, generating 2^H OTS key pairs (sk_j, pk_j) for $0 \leq j < 2^H$, where sk_j is the private signing key and pk_j the corresponding public verification key (note that, for example, using using WOTS, sk_j would itself consist of l private key elements). A Merkle tree is built over the public keys: each leaf is the hash $h(pk_j)$, and each internal node is the hash of the concatenation of its two child nodes. The root of this tree serves as the MSS public key, while the collection of all sk_j values forms the MSS private key. To sign a message M, the signer hashes it to obtain a digest $d = h(M)$, then signs d using the s-th one-time signing key sk_s, producing σ_{OTS}. The Merkle signature includes σ_{OTS}, the corresponding verification key pk_s, the index s, and the authentication path $A_s = (a_0, \ldots, a_{H-1})$, which is the set of sibling nodes needed to reconstruct the path from $g(pk_s)$ to the Merkle root. Verification proceeds in two steps: (1) check that σ_{OTS} correctly signs the digest d using pk_s; (2) use the authentication path and $h(pk_s)$ to reconstruct the root of the Merkle tree. If the reconstructed root matches the known public key, the signature is valid.

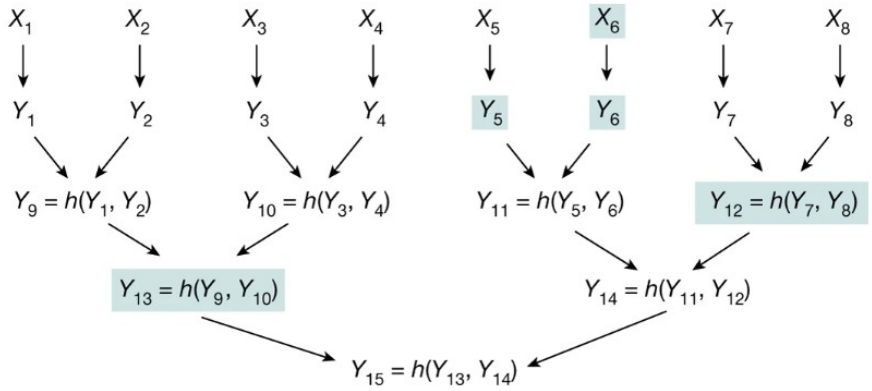

Figure 1: Merkle tree with public key pk_{15} to sign eight messages. Highlighted nodes denote the authentication path for leaf pk_6. Image from [5].

As with WOTS+, if, each time we hash a value we first XOR it with a random bitmask, we don't require collision resistance from the hash function, increasing security. This variant is known as XMSS (eXtended Merkle Signature Scheme) [8].

6 HORS

Notice that so far, all the signature schemes have been stateful, meaning the user has to keep track of the private keys he has used. Stateful signature schemes are problematic because they require the signer to maintain and correctly update internal state information (such as counters or used key indices) across all signing operations. Any state desynchronization (due to crashes, backups, or concurrent signing) can lead to reused keys or invalid signatures, breaking security guarantees and potentially enabling forgery.

The HORS (Hash to Obtain Random Subsets) signature scheme [18] is a few-times stateless signature scheme defined over a one-way hash function and a selection function $H : \{0,1\}^* \to \binom{[t]}{k}$, where $[t] = \{1, 2, \ldots, t\}$, and $\binom{[t]}{k}$ denotes all subsets of $[t]$ of size k. The private key is a list of random values $SK = \{sk_1, sk_2, \ldots, sk_t\}$, each $sk_i \in \{0,1\}^n$, and the public key is $PK = \{pk_1, pk_2, \ldots, pk_t\}$, where $pk_i = f(sk_i)$ for all $i \in [t]$. To sign a message $M \in \{0,1\}^*$, the signer computes $H(M) = \{i_1, i_2, \ldots, i_k\} \subset [t]$ and outputs the signature $\sigma = \{sk_{i_1}, sk_{i_2}, \ldots, sk_{i_k}\}$. Verification involves computing $H(M)$ and checking that $f(sk_{i_j}) = pk_{i_j}$ for all $j = 1, \ldots, k$. The security of HORS relies on the r-subset-resilient property of the function H: given any r messages $m_1, \ldots, m_r$, it is computationally infeasible to find a new message m' such that $H(m') \subseteq \bigcup_{i=1}^{r} H(m_i)$. This ensures that even after observing r signatures, an adversary cannot forge a valid signature on a new message, assuming the hardness of inverting f and the resilience of H.

Alternatively, one can sacrifice runtime to reduce the public key size and the combined size of a signature and a public key by using a Merkle Tree but signing using HORS instead of an OTS. This variant is known as HORST [3].

7 SPHINCS

The Merkle Signature Scheme requires computing a full Merkle tree with 2^H leaves and $2^H - 1$ internal nodes, which becomes computationally expensive as the height H increases. To improve efficiency, a hyper-tree (or multi-tree) structure can be used [9], consisting of $T \geq 2$ layers of Merkle trees. Each tree's leaves are the public keys of OTS schemes and each tree's root serves as a public key: nodes in intermediate layers sign the roots of trees in the layer below and the single top-layer root is the overall public key. To sign a message, a private OTS key from a bottom-layer tree signs the message and the signature includes the corresponding authentication path and tree root. Since this root is not known to the verifier, it is signed by an OTS key from the tree one level above, along with its own authentication path. This process is repeated layer by layer up to the top. Verification begins at the message and proceeds upward, verifying each OTS signature and reconstructing each tree root via its authentication path, until the top-level root is obtained and compared with the

known hyper-tree public key.

The SPHINCS (Stateless Practical Hash-Based Incredibly Nice Cryptographic Signature) signature scheme [3] builds a virtual hyper-tree of total height h, divided into d layers of Merkle trees, each of height h/d. The leaves of each tree are derived from WOTS+ public keys, and each such tree can be used to sign up to $2^{h/d}$ messages. The structure is hierarchical: the top layer $(d-1)$ contains a single Merkle tree whose WOTS+ keys are used to sign the roots of the trees in the layer below. In general, layer i consists of $2^{(d-1-i)(h/d)}$ trees, and the WOTS+ keys from trees on layer $i+1$ are used to sign the roots of those in layer i. At the lowest layer (layer 0), each WOTS+ key signs a HORST public key, which is then used to sign the actual message. The hyper-tree is called "virtual" because none of it is precomputed or stored in full; instead, all keys and nodes are generated on demand from a master seed and deterministic algorithms, allowing for efficient, stateless signing.

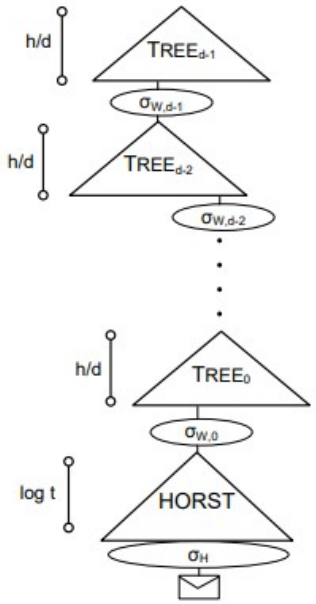

Figure 2: Virtual structure of a SPHINCS signature. Image from [3]

SPHINCS+ [4] retains the high-level structure of SPHINCS but introduces several internal optimizations, most notably the replacement of the few-time signature scheme HORST with FORS (Forest of Random Subsets). Instead of a single large tree as in HORST, a FORS key consists of k binary trees, each of height $\log t$, with a total of kt secret key elements. The leaves are hashes of these secret values, and the public key is computed as a tweakable hash (a

generalization of hashing using bitmasks for improved security analysis) over the concatenation of the k tree roots. The big difference to HORST is that now there is a dedicated set of secret key values per index derived from the message. Although FORS may appear less efficient under identical parameters, its structure allows the use of smaller parameters, ultimately reducing both signature size and computation time. Additionally, SPHINCS+ improves security by eliminating unverifiable index selection. In SPHINCS, the index of the HORST key was pseudorandom and not externally verifiable, enabling multi-target attacks. SPHINCS+ addresses this by deriving both the message digest and the FORS index as $(\text{md}, |, \text{idx}) = H(R, \text{PK}, M)$, where R is a randomized value included in the signature. This binds each message to a unique FORS instance, preventing cross-instance attacks and allowing the index to be omitted from the signature. Moreover, SPHINCS+ incorporates multi-target attack mitigation techniques proposed in [13].

8 SLH-DSA

In 2024, SPHINCS+ was standardized by NIST [17] with some minor adjustments, most notably regarding the accepted parameter sets. Specifically, 12 parameter sets were approved, differing on the hash family used (either SHA-2 or SHAKE), as well as two signature scheme variants: 's' for relatively small signatures ('s') or 'f' to have relatively fast signature generation. The parameters are categorized for security levels 1, 3, and 5. Using this approach, security strength is not described by a single number (e.g., "128 bits of security"). Instead, each parameter set is claimed to offer security at least equivalent to a generic block cipher with a prescribed key size. More precisely, the computational resources needed to break SLH-DSA are stated to be greater than or equal to those required to break the block cipher when computational resources are estimated using any realistic model of computation.

9 Conclusion

Hash-based cryptography stands out as one of the most conservative and well-understood foundations for digital signatures in the post-quantum era. Among its variants, SLH-DSA offers robust, stateless security based purely on the second-preimage resistance of cryptographic hash functions, making it highly reliable even in the face of evolving cryptanalytic threats. The primary tradeoff with SLH-DSA is performance. It is significantly slower and produces much larger signatures compared to lattice-based schemes like ML-DSA or Falcon. However, its flexible parameter sets offer a balance between size and speed, though it remains generally less efficient. This performance cost, however, is justified by the unmatched long-term security confidence it provides, making it particularly valuable for systems that require durable, long-lived signatures or where updating is not feasible.

It is also worth noting that NIST has standardized stateful hash-based schemes like XMSS and LMS [16], which offer better performance than SPHINCS+ but come with the caveat of requiring careful state management. Mismanagement of the signing state can render these schemes insecure, posing challenges for secure deployment in certain environments.

In summary, while hash-based signature schemes may not be the fastest or smallest, they represent the gold standard in conservative security. They are particularly compelling for high-assurance use cases, where resilience against both quantum and classical attacks, as well as the maturity of the cryptographic foundation, are paramount.

References

[1] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of the ACM (JACM)*, 51(4):595–605, 2004.

[2] Daniel J Bernstein. Cost analysis of hash collisions: Will quantum computers make sharcs obsolete. *SHARCS*, 9:105, 2009.

[3] Daniel J Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. Sphincs: practical stateless hash-based signatures. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 368–397. Springer, 2015.

[4] Daniel J Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The sphincs+ signature framework. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2129–2146, 2019.

[5] Daniel J Bernstein and Tanja Lange. Post-quantum cryptography. *Nature*, 549(7671):188–194, 2017.

[6] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. *ACM Sigact News*, 28(2):14–19, 1997.

[7] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. On the security of the winternitz one-time signature scheme. *International Journal of Applied Cryptography*, 3(1):84–96, 2013.

[8] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. Xmss-a practical forward secure signature scheme based on minimal security assumptions. In *Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29–December 2, 2011. Proceedings 4*, pages 117–129. Springer, 2011.

[9] Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. Merkle signatures with virtually unlimited signature capacity. In *Applied Cryptography and Network Security: 5th International Conference, ACNS 2007, Zhuhai, China, June 5-8, 2007. Proceedings 5*, pages 31–45. Springer, 2007.

[10] André Chailloux, María Naya-Plasencia, and André Schrottenloher. An efficient quantum collision search algorithm and implications on symmetric cryptography. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23*, pages 211–240. Springer, 2017.

[11] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC), Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996.

[12] Andreas Hülsing. W-ots+–shorter signatures for hash-based signature schemes. In *Progress in Cryptology–AFRICACRYPT 2013: 6th International Conference on Cryptology in Africa, Cairo, Egypt, June 22-24, 2013. Proceedings 6*, pages 173–188. Springer, 2013.

[13] Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. In *Public-Key Cryptography–PKC 2016: 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I*, pages 387–416. Springer, 2016.

[14] Leslie Lamport. Constructing digital signatures from a one way function. Technical Report CSL-98, October 1979.

[15] Ralph C Merkle. A certified digital signature. In *Conference on the Theory and Application of Cryptology*, pages 218–238. Springer, 1989.

[16] National Institute of Standards and Technology. Recommendation for stateful hash-based signature schemes. Department of Commerce, Washington, D.C., 2020. Available: https://doi.org/10.6028/NIST.SP.800-208.

[17] National Institute of Standards and Technology. Stateless hash-based digital signature standard. Department of Commerce, Washington, D.C., 2024. Available: https://doi.org/10.6028/NIST.FIPS.205.

[18] Leonid Reyzin and Natan Reyzin. Better than biba: Short one-time signatures with fast signing and verifying. In *Australasian Conference on Information Security and Privacy*, pages 144–153. Springer, 2002.

[19] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394, 1990.

[20] Gideon Yuval. How to swindle rabin. *Cryptologia*, 3:187–189, 1979.