Professional Certificate in Post-Quantum Cryptography

Hash-Based Signatures

Hash-based signatures form a class of digital signature schemes whose security relies exclusively on the properties of cryptographic hash functions. Unlike RSA or elliptic-curve signatures, they do not depend on number-theoretic assumptions that are vulnerable to quantum algorithms such as Shor's algorithm. Because the underlying hash function is assumed to be quantum-resistant, the entire construction inherits this resistance, making hash-based signatures a central topic in post-quantum cryptography.

The most fundamental building block is the one-time signature (OTS). An OTS key pair can safely sign a single message; reusing the same key pair for multiple messages compromises security. The concept of a one-time signature dates back to Lamport's original proposal, often called the Lamport OTS. In a Lamport scheme, a private key consists of 2 × n random n-bit strings, where n is the output length of the hash function. The public key contains the hash of each of these strings. To sign a bit of the message, the signer reveals one of the two private strings depending on the value of the bit; the verifier checks the hash against the corresponding public component. This construction is simple, provably secure under the assumption that the hash function is pre-image resistant, and it illustrates the essential idea that a signature can be obtained by revealing secret data that can be verified through hashing.

Because a Lamport OTS can only sign a single n-bit value, practical systems need a method to reuse the scheme for many messages. The standard approach is to combine many OTS instances into a larger structure called a Merkle tree. In a Merkle tree, each leaf node holds a public key of an OTS, and internal nodes store the hash of their two children. The root of the tree, a single hash value, serves as the master public key for the whole system. When a signer wishes to sign a message, they select an unused OTS leaf, generate a signature for the message using that OTS, and then provide an authentication path: The sibling hash values from the leaf up to the root. The verifier recomputes the path hashes, arrives at the root, and checks that it matches the master public key. This technique allows a signer to produce up to 2^k signatures if the tree has height k, while each individual OTS remains one-time.

The Merkle-tree construction introduces several important terms. The tree height (or depth) determines the maximum number of signatures; a height of 20 permits about one million signatures. The authentication path (sometimes called the Merkle proof) consists of k hash values that enable the verifier to reconstruct the root. The leaf index identifies which OTS key pair is being used for a particular signature; it must be tracked to avoid reuse. The root hash is the public key of the entire scheme and is typically distributed out-of-band or stored in a certificate.

While the original Lamport OTS is conceptually simple, it is not the most efficient in terms of signature size. Several refinements have been introduced. The Winternitz OTS (WOTS) reduces the size of signatures by encoding the message into a base-w representation, where w is a small integer (commonly 4, 16, or 256). In WOTS, each private key element is a chain of hash values; the length of the chain depends on the corresponding digit of the message. Signing involves revealing the appropriate prefix of each chain, and

verification extends the revealed portion to the end of the chain using the hash function. The parameter w trades off signature size against computation: Larger w yields shorter signatures but requires longer hash chains. The term chain length refers to the number of hash iterations needed for the longest possible chain, which is w – 1 in the standard WOTS construction.

A further improvement is the WOTS+ variant, which adds a randomization step to each hash in the chain. Specifically, each hash operation incorporates a per-chain mask derived from a pseudo-random generator. This mitigates certain structural attacks that exploit the linearity of plain hash chains. The masks are part of the public parameters and do not increase the size of the signature, but they do increase the computational cost modestly.

When a Merkle tree is built on top of WOTS or WOTS+, the resulting scheme is commonly referred to as Merkle signature scheme (MSS) or MSS-WOTS. In practice, the term XMSS (eXtended Merkle Signature Scheme) is used for a standardized version that specifies concrete parameter choices, including hash function, tree height, and Winternitz parameter. XMSS defines a stateful signing process: The signer must keep a counter of used leaf indices and ensure that no index is ever reused. The state must be stored securely, because accidental reuse of a leaf destroys the one-time security guarantee and can lead to private-key recovery.

A stateful design raises operational challenges. Implementations must provide reliable non-volatile storage for the leaf counter, guard against power loss, and handle concurrent signing requests without race conditions. To address these concerns, the HSS (Hierarchical Signature Scheme) construction builds a hierarchy of Merkle trees. In HSS, each node of a higher-level tree signs a lower-level tree’s root, effectively delegating signing authority. This reduces the frequency with which the top-level private key is used, allowing a longer overall signature lifespan without frequent state updates at the top level. The hierarchy introduces new vocabulary: The parent tree, the child tree, and the intermediate public key that links them.

Beyond the deterministic Merkle-tree approach, there exist stateless hash-based schemes that avoid maintaining a counter. The most prominent example is the SPHINCS+ protocol, which combines a few advanced techniques: A few-time signature (FTS) scheme called Forest of Trees (FOT), a randomized OTS, and a tuned Merkle tree. SPHINCS+ replaces the stateful leaf selection with a pseudo-random function (PRF) that deterministically derives a leaf index from the message and a secret seed. The PRF ensures that each message maps to a unique leaf, thereby preventing reuse. However, because the mapping is deterministic, an attacker could attempt to force collisions; SPHINCS+ mitigates this risk by employing a large enough leaf space and by using a robust PRF based on the same hash function family.

Key terms associated with SPHINCS+ include hypertree, FORS (Forest of Random Subsets), and tuned parameters. The hypertree is a multi-level Merkle structure where each level uses a different OTS scheme; the top level provides a root that signs the next level, and so on, until the bottom level signs the actual message. FORS is a short-signature component that aggregates many small OTS signatures into a single compact value, reducing the overall signature size. The term tuned parameters refers to the specific choices of tree height, Winternitz parameter, and number of FOTS instances that balance signature size, verification speed, and key generation cost.

The security of hash-based signatures rests on several cryptographic assumptions. The primary assumption is the pre-image resistance of the underlying hash function: Given a hash output, it should be computationally infeasible to find any input that maps to that output. A related assumption is second-pre-image resistance, which states that, given an input and its hash, it should be hard to find a different input with the same hash. Finally, collision resistance asserts that it is hard to find any two distinct inputs that hash to the same value. In the context of hash-based signatures, pre-image resistance is the most critical, because an attacker would need to invert the hash to forge a signature. Collision resistance is also relevant for certain attacks on the Merkle tree structure, where an adversary might try to construct two different authentication paths that lead to the same root.

When selecting a hash function for a hash-based signature scheme, common choices include SHA-2, SHA-3, and the NIST-approved SHA-KE family (e.G., SHA-KE256). These functions provide security levels measured in bits; for post-quantum security a 256-bit output is generally recommended to achieve a 128-bit security margin against quantum attacks, since Grover's algorithm offers a quadratic speedup. The term quantum security level quantifies this margin: A 256-bit hash yields roughly 128-bit security against quantum adversaries, which aligns with current recommendations for post-quantum cryptographic primitives.

Practical applications of hash-based signatures span several domains. In firmware updates for embedded devices, the small public-key size and straightforward verification process are advantageous. A device can store the root hash in read-only memory, and each update can be signed using a Merkle-tree leaf; the device verifies the signature and the authentication path before applying the update. Similarly, in secure boot chains, the bootloader can verify the integrity of the next stage using a hash-based signature, ensuring that an attacker cannot replace firmware without possessing the appropriate OTS private key material.

Another important use case is in certificate transparency logs. Log servers can sign entries using a hash-based scheme, providing a provably secure audit trail that is resistant to quantum attacks. Because the signatures are short and verification is fast, large volumes of log entries can be processed efficiently. In addition, hash-based signatures are well-suited for blockchain and distributed ledger technologies, where the immutability of data and the need for long-term security are paramount. Some blockchain proposals incorporate XMSS or SPHINCS+ signatures to protect transaction authenticity for decades into the future.

Despite their strengths, hash-based signatures have notable challenges. The most prominent is the requirement for state management in schemes like XMSS, which can be error-prone. If a signer accidentally reuses a leaf, the security of the entire system collapses: An attacker can recover the private key by solving a set of equations derived from the two signatures. This risk necessitates careful implementation, often involving hardware security modules (HSMs) that enforce single-use semantics. Stateless schemes such as SPHINCS+ avoid this problem but at the cost of larger signatures (often several tens of kilobytes) and higher computational overhead during signing and verification.

Signature size is another practical concern. While traditional RSA signatures are a few hundred bytes, and ECDSA signatures are around 70–80 bytes, hash-based signatures can range from a few kilobytes for XMSS with modest tree heights to over 50 KB for high-security SPHINCS+ parameter sets. This increase impacts bandwidth, storage, and latency, especially in constrained environments like IoT devices. Researchers

mitigate this by tuning parameters: Reducing tree height lowers the number of possible signatures but also reduces the authentication path length, thereby shrinking the overall signature. In some deployments, a hybrid approach is used, where a short-lived symmetric key is protected by a hash-based signature, and subsequent messages are authenticated with a MAC, thus amortizing the cost.

Key generation time can be substantial because building a Merkle tree requires hashing all leaf public keys and internal nodes. For large trees, this process can take seconds to minutes on modest hardware. To address this, implementations often pre-compute the tree offline and store the root and authentication paths in flash memory. Some schemes support incremental tree construction, where the tree is built gradually as signatures are issued, spreading the computational load over time. This approach introduces additional terminology such as tree traversal algorithm (e.G., The classic “logarithmic” algorithm, the “fractal” algorithm, or the “stack-based” method) that determines how leaf nodes are generated and how authentication paths are updated without recomputing the entire tree.

The term forward security describes a property where compromise of the private key at a certain time does not endanger signatures generated earlier. In stateful hash-based schemes, forward security is inherent: Once a leaf is used and its private component is revealed, the remaining unused leaves remain secret. However, if the top-level private key is compromised, an attacker can generate new signatures for any future leaf, breaking forward security. Hierarchical constructions like HSS improve forward security by limiting exposure to lower-level keys; compromise of a child key does not affect the parent keys.

Another important notion is post-quantum security margin. While hash-based signatures are believed to be quantum-resistant, the exact security level depends on the hash function’s output size and the algorithmic assumptions. Grover’s algorithm reduces the effective security by a factor of two, so a 256-bit hash offers 128-bit post-quantum security. Some standards recommend using a 384-bit hash for a 192-bit security margin, especially in high-value contexts such as national security communications. The term security level therefore encapsulates both classical and quantum considerations.

Implementation details also introduce specific vocabulary. The seed is a secret value used to derive private keys for OTS instances; it can be generated randomly or derived from a master secret using a key-derivation function (KDF). The PRF key is a secret key for the pseudo-random function that maps messages to leaf indices in stateless schemes. The address structure is a set of fields that uniquely identify nodes in the Merkle tree (e.G., Layer, tree, type, and index); it is used as input to the hash function to ensure domain separation. The domain separator is a constant or prefix added to hash inputs to prevent cross-protocol collisions.

In the context of verification, the term aggregate verification refers to the ability to verify multiple signatures simultaneously, often by combining authentication paths. While not inherent to basic Merkle-tree schemes, certain extensions allow batch verification, reducing the total number of hash computations. This is valuable in settings such as certificate transparency, where a client may need to verify hundreds of signatures quickly.

The key-compromise impersonation (KCI) attack is a scenario where an adversary who learns the private key of a signer attempts to forge signatures on behalf of others. In hash-based signatures, KCI resistance is

generally strong because the private key material for each leaf is independent; knowledge of one leaf does not enable forging signatures for other leaves. However, if the master seed is compromised, KCI becomes possible for all future signatures, emphasizing the importance of protecting the seed.

Another practical consideration is signature size variability. In some schemes, the length of the authentication path may vary depending on the leaf index, especially when using optimized traversal algorithms that prune unused branches. This can affect protocol design, as peers must be prepared to handle signatures of different lengths. To simplify integration, many implementations pad signatures to a fixed length, trading off bandwidth for predictability.

The term compression is sometimes applied to hash-based signatures, where the authentication path is encoded using techniques like Merkle-tree hash compression or truncated hashes. Compression reduces the transmitted size but may weaken security if the truncation length is too short. Standards typically prescribe minimum hash output sizes to avoid such pitfalls.

When integrating hash-based signatures into existing PKI infrastructures, the X.509 Extension can be used to embed the root hash and algorithm identifiers. This allows traditional certificate processing tools to recognize and validate hash-based signatures without extensive modifications. The extension includes fields such as algorithm OID, parameter set, and public-key data (the root hash). Interoperability with legacy systems requires careful mapping of these fields.

The performance metrics most commonly reported for hash-based signatures are signing time, verification time, key generation time, and signature size. Signing time in XMSS is dominated by OTS signing and authentication path generation, usually on the order of a few milliseconds on a modern CPU. Verification time is similar, as it requires recomputing the OTS verification and traversing the path. SPHINCS+ signing can be slower (tens of milliseconds) due to the multiple layers of OTS and the FOTS component, while verification may be faster because the verification algorithm can be parallelized across cores.

A related term is parallelizability. Because each hash operation is independent, many hash-based schemes lend themselves to parallel execution. For instance, building a Merkle tree can be performed using a parallel reduction algorithm where leaf hashes are computed simultaneously, then internal node hashes are computed in successive rounds. This property is advantageous on multi-core processors and GPU architectures, where large batches of signatures can be generated or verified efficiently.

In research, the concept of parameter tuning is used to explore the trade-offs between security level, signature size, and computational cost. Researchers often conduct a parameter sweep, varying the tree height, Winternitz parameter, and hash output length, then measuring the resulting metrics. The outcome is a set of recommended parameter sets for different security targets (e.G., 128-Bit, 192-bit, and 256-bit security). These sets are documented in standards such as the NIST Post-Quantum Cryptography Standardization Process, where XMSS and SPHINCS+ parameter sets are listed.

Another term that appears in advanced discussions is tight security reduction. This refers to a proof that the security of the signature scheme reduces to the hardness of the underlying hash function without a large loss factor. Tight reductions are desirable because they give confidence that the chosen parameters truly

achieve the intended security level. In the case of XMSS, the security proof is tight under the assumption that the hash function is a random oracle; for SPHINCS+, the reduction is less tight because of the multiple layers and the use of a few-time signature component.

The random oracle model (ROM) is an idealized setting where a hash function behaves like a truly random function accessible to all parties. Many security proofs for hash-based signatures are conducted in the ROM, which simplifies analysis but introduces a gap between theory and practice. Real hash functions are deterministic and may have structural properties that deviate from a random oracle. Consequently, designers must ensure that the chosen hash function satisfies additional criteria such as being indifferentiable from a random oracle, a property enjoyed by SHA-3 and SHA-KE families.

In the context of hardware implementations, the term side-channel resistance is critical. Because hash-based signatures involve many hash operations, they can be vulnerable to timing attacks, power analysis, or electromagnetic leakage. Countermeasures include constant-time implementations of the hash function, masking techniques, and random delays. Some hardware security modules provide built-in support for hash-based signatures, offering protected storage for seeds and automatic management of leaf indices.

The revocation mechanism for hash-based signatures differs from traditional PKI. Since each OTS key is used only once, revoking a compromised leaf is straightforward: The signer simply stops using that leaf and marks its index as revoked. However, revoking the master root key requires a new root hash to be distributed, which may involve a re-keying ceremony similar to that used in conventional PKI. In hierarchical schemes, revocation can be performed at different levels, providing flexibility.

One more concept is the forward-secure signature variant, where the private key evolves over time, and past signatures remain secure even after the key is updated. In hash-based settings, forward security can be achieved by using a one-time OTS per time period and discarding old private material. This approach aligns with the natural one-time nature of OTS schemes and is useful in environments where key compromise is a realistic threat.

Finally, the term cryptographic agility describes the ability of a system to switch hash functions or parameters without significant redesign. Because hash-based signatures are built on top of a generic hash primitive, they exhibit high agility: Replacing SHA-256 with SHA-3-256, for example, only requires updating the hash invocation and possibly the mask generation routine. This property simplifies future upgrades when new hash functions are standardized or when security assessments dictate a change.

Collectively, these terms and concepts constitute the essential vocabulary for understanding hash-based signatures in a post-quantum context. Mastery of the definitions, the relationships between one-time signatures, Merkle trees, hierarchical constructions, and stateless designs, as well as awareness of practical considerations such as state management, signature size, and side-channel resistance, equips learners to design, implement, and evaluate secure hash-based signature schemes for a wide range of applications.