

A Black-Box Impossibility Result for Threshold Signing Standard Hash-Based Signatures

Yashvanth Kondi
Silence Laboratories
yash@ykondi.net

Naman Kumar
IRIF*
naman.kumar@irif.fr

September 23, 2026

Abstract

Hash-based signature schemes present an appealing proposition: they are broadly regarded as the most conservative in terms of assumptions, and their efficiency profiles suit many applications. However, little is known about their compatibility with threshold signing, which is common practice today in domains like digital asset security. Especially in the context of an impending migration to post-quantum cryptography, this gap in the literature leaves uncertain what hash-based standards would imply for applications that rely on threshold signing. In this work, we develop a new framework by which to analyze such hash-based schemes, and prove inherent barriers to designing an entire class of efficient (fully distributed) threshold signing protocols. Our framework is simple, and yet powerful—our results encompass standard schemes such as XMSS and SLH-DSA.

Our approach is to connect hash-based signatures to the literature on straight-line extraction in the random oracle model. We then apply and build upon techniques from the recent work of Doerner et al. (Crypto '24) to show that threshold signing protocols for hash based schemes must—beyond a certain number of signers—use the circuit representation of the hash function, i.e. it can not use the hash function as a black box. We then give a more fine-grained analysis for the specific case of the Winternitz signature scheme—a building block of both NIST standards—and show that it can not permit even two-party signing with blackbox use of the hash function.

*IRIF (Institut de Recherche en Informatique Fondamentale) is a CNRS research laboratory of Université Paris Cité. Part of this work was done while the author was at Silence Laboratories.

Contents

1	Introduction	3
2	Technical Overview	5
3	Preliminaries	8
3.1	Winternitz One-Time Signatures (W-OTS ⁺)	9
3.2	Stateless Hash-Based Signatures (SPHINCS)	10
4	Extractability for Signature Schemes	11
4.1	Extractability for One-Time Signatures	11
4.2	Extractability for Multiple-Time Signature Schemes	13
4.2.1	Extractability of XMSS	14
4.2.2	Extractability of SPHINCS and SPHINCS+	16
5	Black-Box Distributed Key Generation is Impossible	19
5.1	Distributed Key Generation in the ROM	19
5.2	Impossibility of Key Generation	19
6	What About 2-Party Signing?	20
6.1	MPC for W-OTS ⁺	21
6.2	MPC for Other Schemes	24

1 Introduction

Threshold cryptography is a concept that dates back to Desmedt [Des88], in which the authority to operate a cryptosystem can be distributed amongst a group of parties. The nature of this distribution is such that a consortium of parties of size greater than the threshold must cooperate in order to use the secret key, for example to sign messages. Today, threshold cryptography is widely used, particularly in the blockchain and digital asset security domain. Threshold signatures for instance present a natural solution to the notorious problem of managing signing keys; compromised signing keys directly result in the theft of assets, and so the decentralization of signing authority greatly mitigates this risk. All-but-one corruption resilience—particularly two-party threshold signing is a cornerstone of modern custody solutions, and is the focus of this work.

Threshold signatures are a special case of Multiparty Computation (MPC), and it is of course possible to use generic tools (including MPC) to decentralize the computation of any signature scheme [GMW87, BGG⁺18]. However, the practical applicability of threshold signing is highly dependent on the structure of the signature itself. While the benefits of threshold signing in the digital asset security domain were clear in theory, its adoption picked up in practice only upon the development of efficient distributed signing for ECDSA in particular [GGN16, Lin17, DKLs18, GG18, LN18], underscoring the importance of compatibility with widely used standards.

With several institutions having set dates in the near future to deprecate quantum-unsafe cryptography, the community is now in the midst of evaluating the costs that different post-quantum signature schemes would induce for different applications. In terms of ease of threshold signing, multivariate cryptography is known to permit efficient decentralization [CEN25], as are isogenies in certain settings [CS20, ABCP23]. There has been remarkable progress in lattice based distributed signing as well [DOTT22, DKM⁺24, PN25], including for the NIST standard ML-DSA [BCdP⁺25, BdCE⁺25]. However, barring limited trusted dealer schemes [KLL25, BBD⁺26], there is very little literature on the efficiency of *standards-compatible* threshold signing for an entire class of post-quantum signatures: those based on symmetric-key primitives alone. Given the increasing interest in deploying such signatures—particularly standards such as XMSS and SLH-DSA—this gap is important to address, and is the focus of our work.

Symmetric-key primitives like block ciphers are typically far less structured than public-key ones, and are therefore believed to be the safest option for post-quantum security where possible. Digital signatures are known to be implied by one-way functions alone [Lam79, NY89, Rom90], which presents a strong theoretical foundation for symmetric-key based signatures. As for practical constructions, we find it useful to group them into two paradigms:

- **NIZK based.** In this approach, the public key is a one-way function of the signing key, i.e. $\text{pk} = \text{OWF}(\text{sk})$, and a signature is essentially a non-interactive zero-knowledge proof (NIZK) that proves knowledge of sk , where the parameters for the NIZK incorporate the message being signed. The NIZK itself is instantiated with a hash-based construction, yielding a signature that only makes use of symmetric-key primitives end to end. Schemes that take this approach include the NIST second round “additional digital signature” candidate FAEST [BBD⁺23], MPC-in-the-head schemes such as Picnic [ZCD⁺20], and other esoteric constructions based on proof systems like STARKs [AdSGK24].
- **OTS based.** Lamport’s One-time Signature (OTS) makes blackbox use of a hash function H ; the public key pk represents a collection of evaluations of H , and a signature consists of simply revealing a subset of the pre-images. Generalizing this principle and using sophisticated relations between the hash evaluations and pk yields more performant schemes such as HORS [RR02], the Winternitz OTS and its follow-ups [BDE⁺11, Hül13], and the Merkle signature scheme and its NIST-standardized version XMSS [BDH11]. While all of these constructions are stateful and entail an explicit bound on the number of signatures per public key, Goldreich [Gol04] showed how one can compose OTS constructions to achieve a stateless scheme that supports a practically unlimited number of signatures. This template is instantiated with practical efficiency in SPHINCS [BHH⁺15a] and the

NIST standard SLH-DSA based on SPHINCS+ [BHK⁺19]. These schemes are relatively simple to understand and implement, and their standardized versions (and variants thereof) are already under consideration for wide deployment.

Interestingly, *none* of the above schemes are known to support meaningfully tailored efficient threshold signing protocols, i.e. the only known method by which to fully decentralize their computation is to use generic MPC. This would entail evaluating several million gates in MPC for each signature, which, while not unprecedented, is much more complex than threshold signing protocols commonly deployed in the industry today.

The recent work of Doerner et al. [DKR24] contains perhaps the first evidence of an inherent difficulty in designing tailored threshold signing protocols, particularly in the all-but-one corruption setting, which is also the focus of this work. In particular, Doerner et al. showed (with certain caveats) that any secure distributed protocol to compute symmetric-key NIZK-based signatures must necessarily use the circuit representation of the symmetric-key primitive—one can not hope to design a custom threshold signing protocol that treats such primitives as a black box. For standard symmetric key functions like AES or SHA, the only interpretation (of which we are aware) of such a result is that we must resort to generic secure computation machinery.

There are several questions that Doerner et al. leave open in their work. For one, their bounds are generic, but not tight. Their theorem rules out blackbox threshold signing amongst more than a certain number of signing nodes as determined by the NIZK underlying the signature, but this number can be quite high. There are no direct implications for say, the two-party case, which pertains to a large class of real use cases. However, they do remark informally that inspecting each NIZK on a case by case basis can yield much tighter bounds.

What About OTS Based Schemes? The technique that Doerner et al. developed to reason about NIZK-based signatures depends very strongly on the NIZK formulation. At a high level, they show that any distributed protocol to compute such NIZK-based signatures that treats symmetric-key primitives as a black box, will be vulnerable to an attack wherein an adversary that simply runs the “straight-line extractor” of the NIZK on the views of enough parties can obtain the secret key. Such an extractor $\mathcal{E}$ is an algorithm whose existence satisfies the “proof of knowledge” property of a zero-knowledge proof in the random oracle model— $\mathcal{E}$ in this context is able to obtain the witness for the proven statement, given the proof string and the random oracle queries that were made during its generation. Doerner et al. argue that essentially all known constructions in the NIZK-based paradigm of signatures permit such an algorithm, and are therefore subject to their bound. However, OTS schemes do not satisfy any meaningful notion of proof of knowledge, or indeed any of the standard notions associated with zero-knowledge proofs. Reasoning about distributed signing for an OTS in a similar fashion seems require a fundamentally different approach—once the symmetric-key primitive is treated as a black box (i.e. modeled as a random oracle) there is no NP statement that characterizes the relationship between the public and signing keys and the signatures.

This Work. We study OTS based signatures from a blackbox threshold signing lens, particularly those that underlie standards such as XMSS and Sphincs, and are under consideration for wide deployment. We increase the scope and depth of the line of study that Doerner et al. initiated, in particular:

- **Extractability Framework for Signatures.** We formulate a definition of extractability that can be applied to characterize any symmetric-key based signature scheme. Roughly, our definition mandates the existence of an algorithm $\mathcal{E}$ that extracts a *signature forgery* rather than an NP witness.
- **General Bound.** With our extractability definition in place, we extend the technique of Doerner et al. to derive an upper bound on the number of signers amongst whom threshold signing can be distributed in a blackbox fashion, for any given (extractable) signature scheme.

- **Minimal Extractable Set Method.** Towards proving tighter bounds on the number of signers that may run a blackbox protocol, we introduce the notion of a “minimal extractable set”, which characterizes the size of the smallest set that an extractor needs in order to succeed. While we do not give a general method to derive this number, we show by an analysis of the W-OTS⁺ signature scheme that one can derive substantially better bounds than the generic result with this technique.
- **Tight Bound for Standards.** We give a more fine-grained analysis of the W-OTS⁺ signature scheme, to show that it can not permit even two-party signing without suffering an unacceptable concrete security loss. While we draw on tools introduced as part of our minimal extractable set analysis, we do not immediately obtain a more generalizable methodology. As W-OTS⁺ as a building block underlies both hash-based NIST standards, XMSS as well as SLH-DSA, this result has wide implications.

Our results therefore establish comprehensively that we must resort to non-blackbox methods if we wish to fully decentralize the computation of standardized symmetric-key based signature schemes. We stress that our results must not be interpreted as a deterrent to attempt threshold signing for symmetric-key based signatures. In fact, we leave several interesting lines of inquiry open:

- How much nonblackbox use of the symmetric-key primitive is sufficient for threshold signing? Put differently, if one is given an MPC oracle to compute the symmetric key primitive securely, how many times would a threshold signing protocol need to invoke such an oracle?
- Is it possible to design a symmetric-key based signature that resists extraction per our definition?
- Can we design and optimize custom MPC gadgets to improve threshold signing in this context?

We believe that the results and analytical methods we introduce in this paper can serve as a useful reference when investigating the above questions.

AI Disclosure. All the ideas in this paper were the original work of the authors, and all text is human-written. GPT-5.6 Sol Ultra was used to identify bugs (which were later fixed by hand) and craft figures. A previous version of this paper had a weaker argument specifically for the probability analysis of Theorem 6.1, which was fed into GPT. Using this, GPT was able to come up with a better concrete calculation for the success probability of the DKG adversary. The presented step-polynomial technique was unknown to the authors and was later thoroughly verified. It can be found in the standard reference [Wil90].

Acknowledgements. The authors would like to acknowledge Geoffroy Couteau and Yuval Ishai for helpful discussions and insightful observations.

2 Technical Overview

We will provide here an overview of our main techniques.

Extractable Signature Schemes. We start by recalling the informal Lamport Signature scheme, which is a one-time signature scheme that signs 1-bit messages. The key generation procedure of Lamport samples the secret key as two λ -bit strings $\mathbf{sk} = (s_0, s_1)$ and the public key as their hashes $\mathbf{pk} = (\mathbf{H}(s_0), \mathbf{H}(s_1))$. To sign the message b the signer merely reveals the string s_b and verification is as simple as checking whether $\mathbf{pk}_b = \mathbf{H}(s_b)$. Intuitively, security of the scheme follows from the one-wayness of $\mathbf{H}$: an adversary who wishes to forge a signature for message $1 - b$ must find a valid s_{1-b} that hashes to $\mathbf{pk}_{1-b}$ —this is provably hard when $\mathbf{H}$ is modelled as a Random Oracle, and conjectured to be hard for concrete instantiations with standard hash functions.

Indeed, Lamport forms an archetypical example of the basic paradigm behind *all* known hash-based schemes: such schemes inevitably rely on hiding certain preimages from the verifier during key generation.

Note that in the case of Lamport, if *both* signing preimages (s_0, s_1) are revealed to the verifier, then the entire secret key is leaked. This property—the insecurity that follows revealing hash preimages—is not held by all schemes that use hash functions. For example, every oracle query made during an honest execution of the Schnorr generator and verifier is public: the only such query is made to derive the Fiat-Shamir challenge $H(g^r \| m)$, where r is the nonce and m is the message. We term this property of signatures *extractability*.

Definition 2.1 (Informal). *A signature scheme Π is called extractable if every valid key generator and signer algorithm that produces valid signatures for Π necessarily makes some random oracle queries which, if revealed, allow for the forgery of signatures.*

Our formalization of this definition involves the notion of an extractor; i.e. a probabilistic polynomial time algorithm $\mathcal{E}$ which when given the list of random oracle queries made by the key generator, is able to forge a signature on some message. Our extractability definition draws heavily on *straightline extractability in the random oracle model* [Pas03] and can be thought of as a natural analog in the signature setting, where the goal of the extractor is to forge a signature rather than output a witness for an NP statement.

Threshold Key Generation for Extractable Signatures. Our main result is therefore simple in context: consider any multiparty protocol that realizes a distributed key generation procedure for an extractable signature scheme Π . Furthermore, assume that this protocol is restricted to make only *black-box* use of the hash function, and thus every oracle call made by the protocol must explicitly be made by at least one of the parties. Then, provided a high-enough corruption budget, an adversary learns a large number of the ‘secret’ oracle queries by simple observation. Once it has done so, it can feed the oracle queries that it has learnt to the extractor $\mathcal{E}$ which, as we show, will be able to forge a signature with high probability.

The above high-level sketch forms the core component of our main result, Theorem 5.1. Given an extractable signature scheme, our result provides a bound on the largest possible signing set for a blackbox protocol. This bound, being generic, can be quite loose—it is likely that one can derive tighter bounds for a signature scheme on a case by case basis. As an aid for such closer inspection, we devise *extraction size* as a helper tool.

Definition 2.2 (Informal). *An extractable signature scheme Π is t -extractable if, among the random oracle queries made during key and signature generation, there is almost guaranteed to be a subset of size t that permits extraction of a forgery.*

For W-OTS^+ , for example, the extraction size is as low as 67, implying that no secure black-box protocol for threshold signing W-OTS^+ can exist for $n > 67$ parties. For XMSS , depending on the precise parameters, the number can be as low as $n > 87$.

Extractability of W-OTS^+ . While we are unable to show that any purely hash based scheme is necessarily extractable (and therefore subject to our impossibility) we show that our results are applicable to essentially all hash-based schemes used in practice. In particular, we show that the W-OTS^+ signature scheme is extractable, following which we provide complete proofs for XMSS and $\text{SPHINCS}/\text{SPHINCS}^+$ when initialized with W-OTS^+ as well. We now lay out a quick sketch of the extractability of W-OTS^+ ; a full proof can be found in Theorem 4.1.

Recall that the W-OTS^+ signature scheme consists of two components: the *signing chains* and the *checksum chains*. The key generation algorithm of W-OTS^+ generates ℓ_1 strings (labelled as x_i) that correspond to the signing chains and ℓ_2 strings (labelled as y_i) that correspond to the checksum chains, which are the secret keys. The public key can be output as the $(w - 1)^{\text{th}}$ hashes of the secret keys (where w is the ‘Winternitz Parameter’). Signing a message m is then performed by splitting the message into ℓ_1 words m_i of bitlength $\log w$ each, and hashing the i^{th} signing chain m_i times. This is repeated for each one of the ℓ_1 signing chains.

A visualization can be found in Fig. 1.

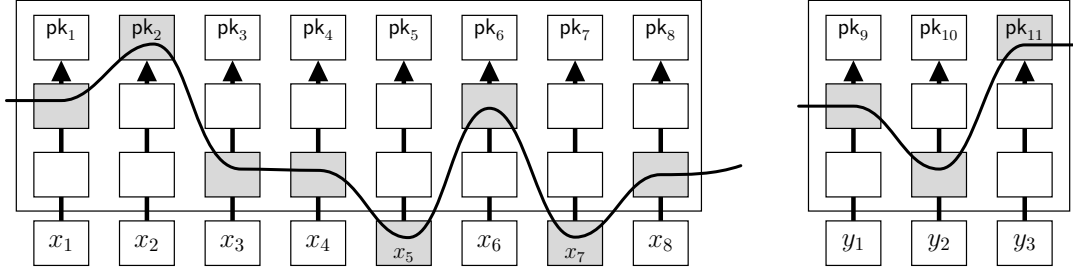


Figure 1: Example of W-OTS^+ signing. The signing chains are represented on the left and the checksum chains are represented on the right. The illustration and the interpretation of the signature as cuts over the signing box is taken from [Bon26].

As illustrated in the figure, each signature consists of the grey boxes, i.e. the m_i^{th} hash of the signing chains and can be viewed as a ‘cut’ through the signing box. Verification is then as simple as hashing each component of the resulting signature $w - 1 - m_i$ times and checking whether the resulting hash is equal to pk_i . Note, however, that this scheme admits a fundamental insecurity: when provided with a signature, any adversary can forge a signature for the message with i^{th} component $m_i + 1$ instead by moving one step up the signing box by hashing! To prevent this, W-OTS^+ also contains the checksum box, which splits $\sum m_i$ into ℓ_2 components m'_j and signs the resultant $w - 1 - m'_j$ using the same technique. Ultimately, any adversary attempting to forge a valid signature by going up the signing chain must also go down a step on the corresponding checksum chain, which requires the adversary to find a preimage of the checksum chain. Put differently, given a signature on m , an adversary can forge a signature on m' only if m' induces a cut that strictly ‘supersedes’ that of m , and the constraints as laid out above make this task computationally infeasible.

Our key observation is that *every* cut through the signing box is a candidate for a signature, and hence key generation must enable signing to handle any cut determined by a message. Observe that for each chain in the signing box, a noticeable fraction (roughly $1/w$) of messages will require the base x_i value—and hence the entire chain—to be revealed when signing. A similar argument applies to the base y_i values in the checksum chains as well. We are therefore able to guarantee that any key generation procedure that produces a valid W-OTS^+ key pair (i.e. enables the signing of any message m) must necessarily query well-defined ‘base’ x_i and y_i elements for each chain, to the random oracle. Hence, one can construct an extractor for W-OTS^+ that simply reads the x_i and y_i values, reconstructs all signing and checksum chains, and is therefore able to sign any arbitrary message. Given that there are a total of $\ell_1 + \ell_2$ such x_i and y_i queries, this determines a relatively small ‘secret query’ set that suffices for extraction.

Extractability of XMSS and SPHINCS. Both XMSS and SPHINCS/SPHINCS+ use W-OTS^+ as a building block, and hence it follows naturally (though not easily!) that a single compromised W-OTS^+ invocation on the tree during the signing procedure can be used to completely compromise the resulting XMSS and SPHINCS/SPHINCS+ instances. Complete details can be found in Section 4.2.

Beyond Extraction Size: Two-party Signing. The above results together show that the main hash-based signing schemes used in practice are subject to a black-box threshold limitation. As we have seen, this result kicks in for a high-enough number of parties (67 for W-OTS^+).

Tightening the bound even further seems to require a more sophisticated strategy than relying on extraction alone. Rather than obtaining straightforward forgeries by simply reading random oracle queries, we show with a more fine-grained analysis of W-OTS^+ (and hence for XMSS and SPHINCS) that any *two-party* blackbox signing protocol must inherently suffer a substantial concrete security loss—enough to rule out real-world use.

Our technique is essentially an application of the pigeonhole principle: any 2-party distributed key-gen/signing procedure for W-OTS^+ must necessarily induce a partition of the hash-queries, with at least one of the parties learning a majority of the base x_i values. An adversary that corrupts this party can then work to find a pair of messages m_0, m_1 such that m_1 's cut along the missing chains supersedes m_0 's cut—note that this is substantially easier than the general case, as the adversary only need find a superseding cut across less than half the chains. This significantly speeds up a brute-force guessing attack and effectively slashes the concrete security for current parameters by enabling more than a doubly-quadratic (factor $1/4$) security loss. Note that this is merely a simple brute-force guessing attack: we do not consider more sophisticated strategies to find message pairs with superseding cuts. Full details can be found in Section 6.

3 Preliminaries

We will be dealing with stateful schemes. Note that our definitions are slightly relaxed; we do not require our signature scheme to have perfect completeness, but merely that correctness holds with some noticeable inverse polynomial probability.

In the following definition we define a signature scheme in the random oracle model which is secure despite the presence of *auxiliary information*. This information aux models key leakage scenarios. By default, we will assume that aux is $\emptyset$ unless stated otherwise, and similarly $\text{st}_0 = \emptyset$ unless stated otherwise (for stateless schemes). We define k -existential unforgeability to denote that the adversary may query the signing oracle k times for the same public key.

Definition 3.1 (Signature Scheme in the ROM (With Auxiliary Information)). *A signature scheme Π in the random oracle model consists of three polynomial-time algorithms that have access to the random oracle $\mathbf{H}$:*

1. $\text{KeyGen}^{\mathbf{H}}(1^\lambda)$ takes as input the security parameter and outputs a pair of secret and verification/public keys $(\text{sk}, \text{pk}, \text{st}_0, \text{aux})$.
2. $\text{Sign}^{\mathbf{H}}(\text{sk}, m, \text{st})$ takes as input the secret key sk , a message m from a message space $\Pi.\text{MsgSpace}_\lambda$ and a state st and outputs a signature σ along with an updated state st' .
3. $\text{Vrfy}^{\mathbf{H}}(\text{pk}, m, \sigma)$ is a deterministic algorithm that takes as input the verification key pk , along with a message m and a signature σ and outputs a bit b .

The scheme has the following properties.

- **Completeness:** for all $\lambda \in \mathbb{N}$,

$$\Pr \left[\text{Vrfy}^{\mathbf{H}}(\text{pk}, m, \sigma) = 1 : \begin{array}{l} (\text{sk}, \text{pk}, \text{st}_0, \text{aux}) \leftarrow \text{KeyGen}^{\mathbf{H}}(1^\lambda) \\ (\sigma, \text{st}') \leftarrow \text{Sign}^{\mathbf{H}}(\text{sk}, m, \text{st}) \end{array} \right] \geq \frac{1}{\text{poly}(\lambda)}.$$

- **k -Existential Unforgeability:** Let the k -existential unforgeability advantage of $\mathcal{A}$ against Π $\text{AdvEU}_{\mathcal{A}}(\lambda, k)$ with auxiliary information aux be the probability that $\text{Vrfy}^{\mathbf{H}}(\text{pk}, m^*, \sigma^*) = 1$ in the k -existential unforgeability game of Figure 2. We say that Π is k -existentially unforgeable with auxiliary information aux if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists some negligible function negl such that for all $\lambda \in \mathbb{N}$ we have

$$\text{AdvEU}_{\mathcal{A}}(\lambda, k) \leq \text{negl}(\lambda).$$

Unless specified, we assume that $\text{aux} = \emptyset$. In such a setting, we state that a scheme is merely k -existentially unforgeable.

Henceforth we will omit the superscript $\mathbf{H}$, and only consider algorithms with access to $\mathbf{H}$.

Stage 1: The adversary queries a *key generation oracle*. The key generation oracle computes $(\text{pk}, \text{sk}, \text{aux}) \leftarrow \text{KeyGen}^H(1^\lambda)$ and responds with (pk, aux) .

Stage 2: The adversary makes a sequence of (upto) k calls to a *signing oracle*, which is initialized with $\text{st} = \emptyset$. For each signing oracle query, the adversary submits a message $m \in \Pi.\text{MsgSpace}_\lambda$ and receives σ as a response, where $(\sigma, \text{st}') \leftarrow \text{Sign}^H(\text{sk}, m, \text{st})$. After each query, the signing oracle privately updates the state to st' .

Stage 3: The adversary outputs a message-signature pair (m^*, σ^*) subject to the restriction that m^* was never queried to the signing oracle in stage 2.

The adversary wins the game if $\text{Vrfy}^H(\text{pk}, m^*, \sigma^*) = 1$.

Figure 2: k -Existential Unforgeability Game.

3.1 Winternitz One-Time Signatures (W-OTS⁺)

In this section we recall the one-time signature scheme of Winternitz, which forms an important cornerstone of our work. We make use of a *hashed* variant of W-OTS⁺, which signs a hashed message instead of a plaintext message.

Parameters. Let λ be the security parameter. We let the message length be $n := n(\lambda)$ and the message space be $\{0, 1\}^n$, with $H_\lambda : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda$ and $H_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ as random oracles. Let w be the Winternitz parameter, and set $\ell = \ell_1 + \ell_2$, where

$$\ell_1 = \left\lceil \frac{n}{\log w} \right\rceil, \quad \ell_2 = \left\lceil \frac{\log(\ell_1 \cdot (w - 1))}{\log w} + 1 \right\rceil.$$

Chain Function. Given an input $x \in \{0, 1\}^\lambda$, a random bitmask $\mathbf{r} = (r_j)_{j \geq i, r_j \in \{0, 1\}^\lambda}$ and any iterative parameter i , we define the chain function c of the W-OTS⁺ as

$$c^i(x, \mathbf{r}) = \begin{cases} \text{if } i = 0 & c^i(x, \mathbf{r}) = x, \\ \text{otherwise} & c^i(x, \mathbf{r}) = H_\lambda(c^{i-1}(x, \mathbf{r}) \oplus r_i). \end{cases}$$

Signature Scheme. The W-OTS⁺ signature scheme is defined in terms of three algorithms, which are presented as follows.

- **W-OTS⁺.KeyGen(1^λ):** The key generation algorithm samples a secret key $\text{sk} := (\text{sk}_1, \dots, \text{sk}_\ell) \in \{0, 1\}^{\ell\lambda}$ and a bitmask $\mathbf{r} := (r_1, \dots, r_{w-1}) \in \{0, 1\}^{(w-1)\lambda}$, and outputs (sk, pk) , where

$$\text{pk} = (\text{pk}_0, \dots, \text{pk}_\ell) := (\mathbf{r}, c^{w-1}(\text{sk}_1, \mathbf{r}), \dots, c^{w-1}(\text{sk}_\ell, \mathbf{r})).$$

- **W-OTS⁺.Sign($m', \text{sk}, \mathbf{r}$):** The signing algorithm takes as input a message $m' \in \{0, 1\}^*$ and converts it to a hashed message of length n as $m := H_n(m')$. It computes a base w representation of m as $(m_1, \dots, m_{\ell_1})$, following which it computes the checksum

$$\text{C} = \sum_{i=1}^{\ell_1} (w - 1 - m_i).$$

Similarly, set the base- w representation of the checksum as $\text{C} = (\text{C}_1, \dots, \text{C}_{\ell_2})$, set $b = (b_1, \dots, b_\ell) = m \parallel \text{C}$ and compute the signature as

$$\sigma = (\sigma_1, \dots, \sigma_\ell) := (c^{b_1}(\text{sk}_1, \mathbf{r}), \dots, c^{b_\ell}(\text{sk}_\ell, \mathbf{r})).$$

- $\text{W-OTS}^+.\text{Vrfy}(m', \sigma, \text{pk})$: The verifier computes $b = H_n(m') \parallel C$ and checks whether

$$\text{pk} \stackrel{?}{=} (c^{w-1-b_1}(\sigma, \mathbf{r}[b_1 + 1..w - 1]), \dots, c^{w-1-b_\ell}(\sigma, \mathbf{r}[b_\ell + 1..w - 1])).$$

It outputs 1 if true and 0 otherwise.

3.2 Stateless Hash-Based Signatures (SPHINCS)

We describe certain preliminaries associated with SPHINCS and SPHINCS+. A full description is omitted and can be found in [BHH⁺15b].

Binary Hash Trees We describe the full binary hash tree used in the SPHINCS specification. A binary hash tree of height h has 2^h leaves and consists of nodes labelled $\text{Node}_{i,j}$ where $0 \leq i < h$ and $0 \leq j < 2^i$ which consist of λ -bit strings. A $(h-1)$ -size set of bitmasks $\mathbf{r}$ where $\mathbf{r}_i \in \{0, 1\}^{2^\lambda}$ is used to construct the tree. Once the leaf values have been set, the nodes are computed as

$$\text{Node}_{i,j} = H((\text{Node}_{i-1,2j} \parallel \text{Node}_{i-1,2j+1}) \oplus \mathbf{r}_i).$$

We label $\text{Node}_{h,0}$ as **root**. Define the authentication path $(\text{auth}_0, \dots, \text{auth}_{h-1})$ of any leaf to be the ordered set of siblings of the nodes on the path from the leaf to **root**. Given the authentication path **auth** and a leaf index i one can compute **root** using the following algorithm.

The algorithm takes as input a leaf index i , a leaf $\text{Node}_{0,i}$ and an authentication path **auth** := $(\text{auth}_j)_{j \in [h]}$ and outputs **root**.

1. Set $P_0 \leftarrow \text{Node}_{0,i}$.
2. From $j := 1$ to h , compute

$$P_j = \begin{cases} H((P_{j-1} \parallel \text{auth}_{j-1}) \oplus \mathbf{r}_j) & \text{if } \lfloor i/2^{j-1} \rfloor \equiv 0 \pmod{2} \\ H((\text{auth}_{j-1} \parallel P_{j-1}) \oplus \mathbf{r}_j) & \text{if } \lfloor i/2^{j-1} \rfloor \equiv 1 \pmod{2} \end{cases}$$

and return P_h .

Figure 3: Root-Finding Algorithm.

Authentication Paths As described above, the authentication path of any leaf i in a BHT is the ordered set of siblings of the nodes on the path from i to **root**. Assume that we start with knowing only the leaf nodes L_i . We show the following lemma, which shows the exact set of queries requires to construct the associated BHT and the authentication path.

Lemma 3.1. *Consider a complete Binary Hash Tree of height h with distinct λ -bit labels in the Random Oracle Model, and assume we are given the labels of each of the leaf nodes L_i . Then computing a valid authentication path $\text{auth}^i = (\text{auth}_0^i, \dots, \text{auth}_{h-1}^i)$ with non-negligible success probability requires a total of (at least) $2^h - 1$ queries to the random oracle. Furthermore, each query is of the form $q_{i,j} = H((\text{Node}_{i-1,2j} \parallel \text{Node}_{i-1,2j+1}) \oplus \mathbf{r}_i)$ for some $i \in [h]$ and $j \in [2^i]$.*

Proof. It is easy to see that the above set of queries suffices to determine the authentication path. We show that the above set of queries is also necessary. Suppose that there is some query $q_{i,j}$ which was not made. Then since H is a random oracle, the value of $\text{Node}_{i,j}$ cannot be computed with better than $1/2^\lambda$ probability. Hence the authentication path of any leaf corresponding to the subtree rooted by the sibling of $\text{Node}_{i,j}$ cannot be computed with validity greater than $1/2^\lambda$. Hence the query $q_{i,j}$ must have been made for each $i \in [1, h]$ and $j \in [2^i]$. $\square$

4 Extractability for Signature Schemes

We start by defining what it means for a signature scheme to be extractable. We seek to capture the idea that a scheme Π is extractable if any valid key generation algorithm for Π hashes delicate information about the secret key; intuition suggests that making such a hash query implies knowledge of the secret key.

4.1 Extractability for One-Time Signatures

Since we would like our definition of extractability to apply to all possible algorithms which are able to sign valid messages, we begin by defining the notion of a valid generator-signer pair for a signature scheme Π .

Definition 4.1 (Valid Generator and Signer). *Let $\Pi = (\Pi.\text{KeyGen}, \Pi.\text{Sign}, \Pi.\text{Vrfy})$ be a one-time signature scheme in the Random Oracle Model. Let (K, S) be a pair of probabilistic polynomial-time algorithms that have access to the random oracle H such that*

1. $K^H(1^\lambda)$ takes as input the security parameter and outputs a pair (ss, pk) where ss is called a signing state and pk is a valid verification key for Π (with a possibly malformed secret key).
2. $S^H(ss, m)$ takes as input a signing state ss output by K along with a message m and produces a signature σ .

We say that (K, S) are valid generator and signer algorithms for Π if for all $\lambda \in \mathbb{N}$ and for all but a negligible portion of messages m ,

$$\Pr \left[\Pi.\text{Vrfy}^H(pk, m, \sigma) = 1 : \begin{array}{l} (ss, pk) \leftarrow K^H(1^\lambda) \\ \sigma \leftarrow S^H(ss, m) \end{array} \right] \geq \frac{1}{\text{poly}(\lambda)}.$$

We now introduce our definition of extractability, which iterates over all such generators.

Definition 4.2 (Extractability of Signature Schemes). *Let $\Pi = (\Pi.\text{KeyGen}, \Pi.\text{Sign}, \Pi.\text{Vrfy})$ be a one-time signature scheme in the Random Oracle Model and let $K(1^\lambda)$ be any valid Generator for Π that outputs a well-formed (ss, pk) pair for Π with non-negligible probability (for example, $\Pi.\text{KeyGen}$ is such an algorithm). Let $\mathbf{Q}_K := \{(q_i, H(q_i))\}_{i \leq \text{poly}(\lambda)}$ be the set of oracle query-answer pairs made during an execution of K . Then we say that Π is an extractable signature scheme if for all such K , there exists some adversary $\mathcal{A}_K$ which, provided with some (ordered) $\mathbf{Q} \subseteq \mathbf{Q}_K$, can forge a valid signature for Π on some message of its choice with non-negligible probability. More precisely, we say that Π is extractable if*

$$\Pr \left[\Pi.\text{Vrfy}(m, pk, \sigma) = 1 : \begin{array}{l} (ss, pk) \leftarrow K(1^\lambda) \\ (m, \sigma) \leftarrow \mathcal{A}_K(1^\lambda, \mathbf{Q}, pk) \end{array} \right] \geq \frac{1}{\text{poly}(\lambda)}.$$

We will further require that given a generator K and the set of queries $\mathbf{Q}_K$ made by it, the extraction set $\mathbf{Q}$ be efficiently constructable. Note that this is a natural property of all hash-based signature schemes in practice; we will show that it holds for schemes of interest.

Definition 4.3 (Extractable Schemes with Constructable Extraction Set). *Let Π be an extractable signature scheme in the ROM. We say that Π has a constructable extraction set if for every generator-signer pair (K, S) , there exist a PPT algorithm $\text{Construct}_K : (\mathbf{Q}', pk) \mapsto \mathbf{Q}$ that takes as input a subset $\mathbf{Q}' \subseteq \mathbf{Q}_K$ of queries made by K that resulted in the production of pk , and with overwhelming probability outputs either (1) a set $\mathbf{Q}$ such that $\mathcal{A}_K(1^\lambda, \mathbf{Q}, pk)$ outputs a valid message-signature pair with noticeable probability (if such a subset exists), or (2) $\perp$ if no such subset exists.*

We now define the minimum extractable set, which (roughly) refers to the size of the lowest number of oracle queries required to forge a signature.

Definition 4.4 (Minimum Extraction Size). *Let Π be an extractable one-time signature scheme which supports constructable extraction, and let K be any valid generator for Π . Let $\mathbf{Q}_K$ refer to the set of query-answer pairs made by K . We define the minimum extraction size of Π to be*

$$\max_{K \in \{0,1\}^*} \left\{ \min \left\{ |Q| : Q \subseteq \mathbf{Q}_K, \Pr \left[\Pi.\text{Vrfy}(m, \text{pk}, \sigma) = 1 : \begin{array}{l} (\text{ss}, \text{pk}) \leftarrow K(1^\lambda) \\ (m, \sigma) \leftarrow \mathcal{A}_K(1^\lambda, Q, \text{pk}) \end{array} \right] \geq \frac{1}{\text{poly}(\lambda)} \right\} \right\}$$

where the maxima is taken over all valid generator algorithms for Π , and the minima is taken over constructable subsets.

We now formalize a proposition that is almost immediate with the above definitions.

Theorem 4.1. W-OTS^+ is extractable.

Proof. Henceforth we let Π refer to the W-OTS^+ signature scheme, with λ being the security parameter, n being the message length, and w being the Winternitz parameter. We let ℓ (and ℓ_1 and ℓ_2) denote the derived length values. Furthermore, we assume that $w = O(\lambda)$.

Let K be any valid generator for Π , which produces a well-formed (ss, pk) pair. We claim that if $(\text{ss}, \text{pk}) \leftarrow K(1^\lambda)$, then interpreting $\text{pk} = (\text{pk}_0, \dots, \text{pk}_\ell)$, K must query on some size- ℓ subset $\{q_i\}_{i \in [\ell]}$ where for each i , $\text{pk}_i = c^{w-1}(q_i, \text{pk}_0)$. If this is true, then our adversary $\mathcal{A}_K$ can forge a signature as follows: $\mathcal{A}_K$ chooses any message m of its choice and runs the algorithm $\text{W-OTS}^+.\text{Sign}(m, \{q_i\}_{i \in [\ell]}, \text{pk}_0)$, which outputs a valid signature with probability 1; the fact that verification succeeds follows simply from the correctness of the W-OTS^+ one-time signature scheme.

We now show that K must make the above queries. Suppose to the contrary that K does not query H on any q_1 such that $c^{w-2}(q_1, \text{pk}_0) = \text{pk}_1$. Consider now the signing algorithm S , which given ss should be able to sign all but a negligible number of messages. We consider the message space to be $\{0,1\}^n$. We are interested in messages m such that $H(m) = 0^{\log w} \| m'$ where $m' \in \{0,1\}^{n-\log w}$ is arbitrary. Begin by noting that the probability (taken over the random oracle) that $H(m) = 0^{\log w} \| m'$ for any individual m is $1/w = O(1/\lambda)$. We let X_i denote the random variable that decides whether the string $x \in \{0,1\}^n$ hashes to some output string with that starts with $0^{\log w}$. Then we have that

$$\mathbb{E}[\#m \mid H(m) = 0^{\log w} \| m'] = \mathbb{E}\left[\sum_{i=0}^{2^n-1} X_i\right] = \sum_{i=0}^{2^n-1} \mathbb{E}[X_i] = \sum_{i=0}^{2^n-1} \frac{1}{w} = \frac{2^n}{w}.$$

Note that this is non-negligible in the space of all messages. For convenience, we can let X denote the RV that counts the number of such ‘good’ messages. Our goal is to show that the number of ‘good’ messages is non-negligible with overwhelming probability over H . If so, it follows that there exists some message m^* for which both (a) S produces a valid signature, and (b) $H(m^*)$ begins with a string of $\log w$ many 0s. To do this, we use the Chernoff Bound to bound the probability of getting $\frac{2^n}{2w}$ -many good messages as

$$\Pr\left[X \leq \frac{(1 - \frac{1}{2})2^n}{w}\right] \leq e^{-\frac{(\frac{1}{2})^2 2^n}{2w}} = e^{-\frac{2^n}{8w}}.$$

which is negligible over the choice of the random oracle.

Based on the above discussion, there exists with overwhelming probability over H some message m^* such that the first $\log w$ bits of $H(m^*)$ are $0^{\log w}$ and for which S can produce a valid signature that verifies with at least inverse polynomial probability. Let $\sigma^* = (\sigma_1^*, \dots, \sigma_\ell^*)$ be such a signature. Then $\Pi.\text{Vrfy}(\text{pk}, \sigma^*, m^*) = 1$ only if $c^{w-1}(\sigma_1^*, \text{pk}_0) = \text{pk}_1$. We recall our assumption that K has not queried H on any q_1 such that $c^{w-1}(q_1, \text{pk}_0) = \text{pk}_1$. Now consider the action of $S(\text{ss}, \text{pk})$.

- If S doesn’t make any such query, then until the verification algorithm checks that $c^{w-1}(\sigma_1^*, \text{pk}_0) = \text{pk}_1$, no such query has been made. Consider that $c^{w-1}(\sigma_1^*, \text{pk}_0) = c^{w-2}(H_\lambda(\sigma_1^* \oplus \text{pk}_0^1), \text{pk}_0[2..w])$; hence the probability that verification succeeds is as most as much as the probability that the latter term evaluates to pk_1 . Since the random oracle maps each input to a uniform output, we can

restrict ourselves to the case in which some query q was made such that $\mathbf{pk}_1 = c^{w-2}(q, \mathbf{pk}_0[2..w])$ (if no such query was made, then we can repeat the previous argument). Since S only has black-box access to the random oracle, the probability that $H_\lambda(\sigma_1^* \oplus \mathbf{pk}_0[1]) = \mathbf{pk}_1$ is at most $\frac{\text{poly}(\lambda)}{2^\lambda}$, since S is polynomial-time. This term is negligible, and it follows that verification only succeeds with negligible probability.

- By the above consideration, it immediately follows that S can only make such a query with probability at most $\frac{\text{poly}(\lambda)}{2^\lambda}$ since it has black-box access to the random oracle and hence cannot do better than a brute-force search.

In either case, we have that verification only succeeds with a negligible probability. Hence, with overwhelming probability over H there exists at least a $\frac{2^n}{2^w}$ -size set of messages which S cannot produce valid signatures for, contradicting the fact that (K, S) are a valid generator and signer pair.

We have shown that K must make such a query q_1 , the chain function of which evaluates to $\mathbf{pk}_1$. Similarly, we can show that K makes such a query for all $\{\mathbf{pk}_i\}_{i \in [\ell_1]}$. What remains is to show that the same is true for $\{\mathbf{pk}_i\}_{i \in [\ell_1+1..\ell_1+\ell_2]}$. To do this, we simply repeat the argument above and show that unless those queries were made by K there exists some non-negligible ‘unsignable’ set of messages such that the checksum of their hashes starts with $0^{\log w}$, and the proof follows. $\square$

Remark 4.1. *If the W-OTS^+ message is signed without hashing first, the above argument is simpler; we can simply show that there has to be an ‘unsignable’ set of messages that starts with $0^{\log w}$, instead of an unsignable set of messages that hashes to some nonce which starts with $0^{\log w}$.*

From the above proof, we can immediately draw the following corollaries.

Corollary 4.1. *W-OTS is extractable.*

Corollary 4.2. *The minimum extraction size of the W-OTS^+ signature scheme is upper bounded by ℓ .*

Proof of Corollary 4.2. We will show that the above subset is constructable. Given any pair (K, S) , our algorithm Construct_K works simply as follows: after initializing $\mathbf{Q} := \emptyset$, for every $q \in \mathbf{Q}'$, q simply checks if there exists some $\mathbf{pk}_j$ for which $\mathbf{pk}_j = c^{w-1}(q, \mathbf{pk}_0)$. If so, it adds (j, q) to $\mathbf{Q}$. If, at the end, $|\mathbf{Q}| = \ell$ and there is such a q for every $\mathbf{pk}_i, i \in \ell$ we are done and the algorithm outputs $\mathbf{Q}$. Otherwise it outputs $\perp$. In any case, we have shown that for every K there is a (constructable) extractable set of size at most λ . Hence the minimum extraction size is at most ℓ . $\square$

Finally, we make the following useful observation that shows that W-OTS^+ in fact satisfies a slightly stronger property than the extractability definition of Definition 4.2: given the query set, it can produce a signature for any message of its choice.

Lemma 4.1. *For every valid generator K of W-OTS^+ (resp. W-OTS), there exists a universal extracting adversary $\mathcal{A}_{\text{W-OTS}^+}$ (resp. $\mathcal{A}_{\text{W-OTS}}$) that takes as input (1) an ℓ -size set of queries $\mathbf{Q}_K$, (2) an honestly generated (by K) public key $\mathbf{pk}_{\text{W-OTS}^+}$, and (3) a message m , and outputs a signature σ on m that verifies with probability exactly 1.*

Proof. The adversary $\mathcal{A}_K$ exhibited in the proof of Theorem 4.1 is exactly such an adversary. $\square$

4.2 Extractability for Multiple-Time Signature Schemes

In the previous section we considered extractability for one-time signature schemes. In this section we extend this notion to multiple-time *stateful* signature schemes.

Definition 4.5 (Valid Generator and Signer, General Setting). *Let $\Pi = (\Pi.\text{KeyGen}, \Pi.\text{Sign}, \Pi.\text{Vrfy})$ be a stateful k -time signature scheme in the Random Oracle Model and $t \leq k$. Let (K, S) be a pair of probabilistic polynomial-time algorithms that have access to the random oracle H such that*

1. $K^H(1^\lambda)$ takes as input the security parameter and outputs a pair (ss, pk, st_0) where ss is called a signing state, st_0 is the start state for the signature scheme, and pk is a valid verification key for Π (with a possibly malformed secret key).
2. $S^H(ss, m, st_i)$ takes as input a signing state ss output by K along with a message m and a state st_i and produces a signature σ along with a state st_{i+1} .

We say that (K, S) are valid generator and t -time signer algorithms for Π if for all $\lambda \in \mathbb{N}$ and for all but a negligible portion of messages m ,

$$\Pr \left[\Pi.\text{Vrfy}^H(pk, m, \sigma) = 1 : \begin{array}{l} (ss, pk, st_0) \leftarrow K^H(1^\lambda) \\ (\sigma, st_{i+1}) \leftarrow S^H(ss, m, st_i) \end{array} \right] \geq \frac{1}{\text{poly}(\lambda)}$$

for all $i < t$.

We now define the corresponding notion of extractability.

Definition 4.6 (Extractability of Signature Schemes, General). *Let $\Pi = (\Pi.\text{KeyGen}, \Pi.\text{Sign}, \Pi.\text{Vrfy})$ be a k -time signature scheme in the Random Oracle Model and let $K(1^\lambda)$ be any valid Generator for Π that outputs a well-formed (ss, pk, st_0) tuple for Π with non-negligible probability (for example, $\Pi.\text{KeyGen}$ is such an algorithm). Let $\mathbf{Q}_K := \{(q_i, H(q_i))\}_{i \leq \text{poly}(\lambda)}$ be the set of oracle query-answer pairs made during an execution of K . For $0 < t \leq k$, we say that Π is a t -extractable signature scheme if for all such K , there exists some adversary $\mathcal{A}_K$ which, provided with some (ordered) $\mathbf{Q} \subseteq \mathbf{Q}_K$, can forge t valid signatures for Π on some message of its choice with non-negligible probability. More precisely, we say that Π is extractable if for all $0 \leq i < t$, we have that the following probability is noticeable:*

$$\Pr \left[\Pi.\text{Vrfy}(m_i, pk, \sigma_i) = 1 : \begin{array}{l} (ss, pk, st_0) \leftarrow K(1^\lambda) \\ (\{m_i\}_{i \in [t]}, \{\sigma_i\}_{i \in [t]}, st_t) \leftarrow \mathcal{A}_K(1^\lambda, \mathbf{Q}, pk, st_0) \end{array} \right]$$

where $\sigma_i \neq \sigma_j$ for all $i \neq j \in [t]$.

We can similarly define the minimum t -extraction size for a signature scheme, which informally refers to the minimum number of queries required for an adversary to be able to forge t signatures of its choice. This is particularly useful for stateful schemes, which maintain a memory of all ‘partial’ signatures that have already previously been utilized. Our definition of constructable extraction is already encapsulated by that of Definition 4.3, and hence we do not repeat it here.

Definition 4.7 (Minimum Extraction Size, General). *Let Π be an extractable signature scheme, and let K be any valid generator for Π . Let $\mathbf{Q}_K$ refer to the set of query-answer pairs made by K . We define the minimum t -extraction size of Π to be*

$$\max_{K \in \{0,1\}^*} \left\{ \min \left\{ |Q| : Q \subseteq \mathbf{Q}_K, \Pr \left[\Pi.\text{Vrfy}(m_i, pk, \sigma_i) = 1 : \begin{array}{l} (ss, pk, st_0) \leftarrow K(1^\lambda) \\ (\{m_i\}_{i \in [t]}, \{\sigma_i\}_{i \in [t]}, st_t) \leftarrow \mathcal{A}_K(1^\lambda, Q, pk, st_0) \end{array} \right] \geq \frac{1}{\text{poly}(\lambda)} \right\} \right\}$$

where the maxima is taken over all valid generator algorithms for Π , and where $\sigma_i \neq \sigma_j$, and the minima is taken over constructable subsets.

4.2.1 Extractability of XMSS

We now show an immediate application of our results. We will work with the eXtended Merkle Signature Scheme (XMSS) as defined in [BDH11] when instantiated with W-OTS. We note that the below proof can be easily modified to support all newer versions of XMSS, including the draft approved by NIST [CAD⁺20] as well as the IETF RFC [HBG⁺18], or any variant instantiated with W-OTS⁺.

Theorem 4.2. *XMSS is 1-extractable with minimum 1-extraction size upper bounded by $\ell + h$.*

Proof. We assume that XMSS is instantiated with parameters λ and h , and the underlying W-OTS parameters. Let K be any valid generator for XMSS, which produces a well-formed (ss, pk, st_0) tuple. Recall the structure of the XMSS public key, which consists of $(root, bitmasks)$, where $root$ is the value of the root node of the authentication path of the XMSS tree and $bitmasks$ is the collection of required bitmasks. We further recall that each leaf of the XMSS tree is associated with a W-OTS key pair. Since W-OTS is an extractable signature scheme, there exists some adversary $\mathcal{A}_{W-OTS}$ which takes as input a W-OTS public key along with a set of queries Q_{W-OTS} where $|Q_{W-OTS}| = \ell$ and outputs a valid message-signature pair (m, σ) that verifies with (at least) inverse polynomial probability.

We now show an adversary $\mathcal{A}_K$ which takes as input a set of $\ell + h$ queries Q_K made by K that satisfies the property of Definition 4.6. Our adversary proceeds as follows: it parses the set $Q_K := (Q_{W-OTS}, Q_K)$ and interprets Q_{W-OTS} as a set of ℓ queries corresponding to the W-OTS key pair associated with $Node_{0,h}$ of the XMSS tree (ie. the first leaf node). The adversary then considers the set Q_K , which consists of a set of h queries of the form $\{((Node_{0,j} \oplus b_{0,j}) \parallel (Node_{1,j} \oplus b_{1,j}))\}_{j \in [h]}$. It sets $Node_{0,h} := pk_{W-OTS}$ and runs $(m_{W-OTS}, \sigma_{W-OTS}) \leftarrow \mathcal{A}_{W-OTS}(1^\lambda, Q_{W-OTS}, pk_{W-OTS})$, before finally outputting

$$(m, \sigma, st') := (m, (0, \sigma, \{((Node_{0,j} \oplus b_{0,j}) \parallel (Node_{1,j} \oplus b_{1,j}))\}_{j \in [h]}), 1).$$

To show that this adversary succeeds with noticeable probability, we need to show that there exists such a set Q_K consisting of queries made by the XMSS key generation algorithm such that the signature so output by $\mathcal{A}_K$ is accepting. Let S be an algorithm such that (K, S) is a valid generator-signer pair for XMSS. Then given a signing state ss , the state 0 , and a message m , S produces a valid signature for all but negligibly many m with noticeable probability. This signature is of the form $(0, \sigma_{W-OTS}, auth)$ where σ_{W-OTS} is the W-OTS signature on message m . Now suppose that no such set Q_K consisting of the queries $\{((Node_{0,j} \oplus b_{0,j}) \parallel (Node_{1,j} \oplus b_{1,j}))\}_{j \in [h]}$ was made by K during key generation.

Consider the action of $XMSS.Vrfy$ on the input $(0, \sigma_{W-OTS}, auth)$. $XMSS.Vrfy$ first parses $auth = (auth_1, \dots, auth_h)$, sets $p_h = pk_{W-OTS+}$ and computes $p_{i-1} := H((p_i \oplus b_{0,i}) \parallel (auth_i \oplus b_{1,i})) = auth_{i-1}$, until a final check of whether $H((p_1 \oplus b_{0,1}) \parallel (auth_1 \oplus b_{1,1})) = root$. Verification succeeds if the last check succeeds. Note that if any one of these queries were not made by K , then the probability of successful verification would be at most $1/2^\lambda$, which contradicts the fact that (K, S) is a valid generator-signer pair. Hence, every such query must have been made by K , and the set so constructed can be taken to be Q_K . In fact, by Lemma 3.1, the previous argument works for every such leaf node.

Building on the previous argument, we also obtain that verification only succeeds if $H((Node_{0,h} \oplus b_{0,h}) \parallel (Node_{1,h} \oplus b_{1,h})) = auth_{h-1}$, where $Node_{0,h}$ is the W-OTS public key corresponding to the first signing leaf. Hence, K necessarily makes a query that contains the value of pk corresponding to that node. To complete the proof, we prove the following lemma.

Lemma 4.2. *There exists a size- ℓ set of queries Q_{W-OTS+} made by K such that the adversary $\mathcal{A}_K$ succeeds with probability 1.*

Proof. We consider the following generator-signer pair for W-OTS:

1. K_{W-OTS} runs the generation algorithm K of XMSS and determines $ss = ss_{XMSS}$ and pk_{XMSS} . Each time K requires a query to the oracle, K_{W-OTS} makes the query and returns it to K .
2. K_{W-OTS} considers the set of queries made by K and uses it to obtain pk_{W-OTS} , which is the public key corresponding to the first leaf of the XMSS tree.
3. K_{W-OTS} then outputs ss_{XMSS} and pk_{W-OTS} .
4. S_{W-OTS} takes as input ss_{XMSS} and a message m along with the state st_0 and runs S to obtain $(0, \sigma_{W-OTS}, auth)$. It outputs the signature σ_{W-OTS} .

It is easy to see that this is a valid generator-signer pair for W-OTS, and hence by Theorem 4.1 it follows that there is a size- ℓ set of queries Q_{W-OTS} made by K_{W-OTS} using which the universal W-OTS adversary

$\mathcal{A}_{\text{W-OTS}}$ can forge a signature on any message of its choice associated with the key pair on $\text{Node}_{0,h}$. However, note that $\text{K}_{\text{W-OTS}}$ makes no oracle queries on its own, and in fact all such queries are made by the subroutine K during step 1. Therefore it follows that K also makes such a set of queries $Q_{\text{W-OTS}}$. $\square$

In the above consideration K is any valid generator for XMSS , and we can conclude that any such K makes the aforementioned set of queries Q_{K} . With this in mind, we can calculate the success probability of $\mathcal{A}_{\text{K}}$ as the probability that $\sigma_{\text{W-OTS}}$ verifies on $m_{\text{W-OTS}}$ times the fact that the authentication check succeeds. Both probabilities are 1: the latter follows from the fact that the random oracle is static, while the former follows from Lemma 4.1. The 1-extractability of XMSS follows immediately.

Finally, we show that the minimum extraction size of XMSS is $\ell + h$, which follows if the extraction set is constructable. We show that the above set can be detected as follows: the algorithm $\text{Construct}_{\text{K}}$ initializes $Q := \emptyset$, and then for every $q \in Q'$ first determines the W-OTS inverses, labels them as (i, q_i) and continues (if there are no such inverses, it outputs $\perp$). Then, it checks for each remaining q whether $H(q) = \text{root}$. If there is no such q it outputs $\perp$. Otherwise, it adds $(\ell + 1, q)$ to Q and checks whether there exists q_1 and q_2 such that $H(q_1) = q[0..\lambda]$ and $H(q_2) = q[\lambda + 1..2\lambda]$ and so on, building the entire authentication path for the relevant W-OTS leaf. It is easy to see that this algorithm is polynomial-time, and that the extraction set is constructable. $\square$

In fact we can repeat the above argument any number of times to show the following corollary, provided that each leaf of the (polynomial-sized) XMSS tree contains a non-degenerate W-OTS key pair. Informally, the proof follows from the consideration that for the authentication path to verify each leaf must consist of a valid, fresh W-OTS public key that is precomputed during key generation.

Corollary 4.3. *For all (constant) $k \leq 2^h$, XMSS is k -extractable with minimum k -extraction size upper bounded by $k(\ell + h)$.*

We now turn to the second major result of this section, which establishes the same result for SPHINCS . SPHINCS is a stateless signature scheme, which means that Definition 4.5 applies with $\text{st}_i = 0$ for all $i \in \mathbb{N}$. We will be using the specification of SPHINCS as in [BHH⁺15b].

4.2.2 Extractability of SPHINCS and SPHINCS^+

Theorem 4.3. *SPHINCS is 1-extractable with minimum extraction size upper bounded by $\ell + h/d$.*

Proof. We recall the structure of the SPHINCS signing tree, which consists of a hypertree of height h that consists of d layers of trees of height h/d . We label the trees as $\text{BHT}_{i,j}$ which refers to the i^{th} tree on the j^{th} layer, where the layers are labelled $0, \dots, d-1$ from bottom to top. The leaves of each tree consist of a W-OTS^+ key pair that is used to authenticate the next tree, except for each leaf node of the 0^{th} tree which is used to authenticate a HORST public key (which is used to sign the message). In the following section we will consider only generator-signer pairs (K, S) in which K does not perform any computation beyond $\text{BST}_{0,d-1}$. We will see how to remove this restriction (*) later.

Like in the case of XMSS , we begin by recalling the structure of the SPHINCS public key, which is of the form $(\text{root}, \text{bitmasks})$ where root denotes the label of the root node of $\text{Tree}_{0,d-1}$ (there is only a single tree at this level), while bitmasks consists of a series of p different λ -bit values that are used as bitmasks for (1) each W-OTS^+ instance, (2) the final HORST instance, and (3) the SPHINCS trees. Now let K be any valid generator for SPHINCS which produces a well-formed (ss, pk) pair and let $\mathcal{A}_{\text{W-OTS}^+}$ be the universal W-OTS^+ adversary which takes as input an ℓ -sized set $Q_{\text{W-OTS}^+}$ along with a message m and produces a signature σ that verifies with probability 1, guaranteed by Lemma 4.1. With this in mind, we claim that the following adversary $\mathcal{A}_{\text{K}}$ of Fig. 4 produces a valid SPHINCS signature with noticeable probability.

Our ultimate goal is to show that there exists such a set of queries Q_{K} for which the above extractor succeeds. We will proceed through a number of lemmas.

$\mathcal{A}_K$ takes in as input the produced $\mathbf{pk}$ along with query set $\mathbf{Q}_K := (Q_{\text{W-OTS}^+}, Q_K)$ and runs the following algorithm.

1. $\mathcal{A}_K$ picks a random string $R := (R_1, R_2) \in \{0, 1\}^{2\lambda}$ and a message m uniformly at random and computes the message digest $D = H(R_1, m)$.
2. $\mathcal{A}_K$ uses the first h bits of R_2 to determine an index i . It chooses a HORST address

$$A_{\text{HORST}} = (d \| i[0, (d-1)h/d] \| i[(d-1)h/d, h/d]),$$

ie. uses the first $(d-1)h/d$ bits to determine the index of a (bottom-layer) tree and the remaining h/d bits to determine a leaf index of that tree.

3. $\mathcal{A}_K$ randomly samples a HORST seed S_{HORST} and computes an internal secret key $\mathbf{sk}_{\text{HORST}}$ using which it signs D and outputs $(\sigma_{\text{HORST}}, \mathbf{pk}_{\text{HORST}})$.
4. $\mathcal{A}_K$ takes A_{HORST} and sets the first $\lceil \log(d+1) \rceil$ bits to zero and labels the resulting string A_0 . It then samples a W-OTS⁺ key pair $(\mathbf{sk}_0, \mathbf{pk}_0)$ and uses it to produce a signature on $\mathbf{pk}_{\text{HORST}}$ called $\sigma_{\text{W-OTS}^+, 0}$, following which it computes an authentication path $\mathbf{auth}_0$ for the produced W-OTS⁺ key pair at A_0 . It compresses $\mathbf{pk}_0$ using an L-tree and then applies the algorithm of Fig. 3 to compute $\mathbf{root}_0$.
5. $\mathcal{A}_K$ repeats the above procedure for layers 1 to $d-2$.
6. On reaching layer $d-1$ (the top layer), $\mathcal{A}_K$ sets $\mathbf{auth}_{d-1} = Q_K$. It runs $\sigma_{\text{W-OTS}^+, d-1} \leftarrow \mathcal{A}_{\text{W-OTS}^+}(Q_{\text{W-OTS}^+}, \mathbf{root}_{d-2})$ to produce a signature and an associated public key $\mathbf{pk}_{d-1}$, and follows the rest of the previous steps without change.
7. $\mathcal{A}_K$ outputs $\sigma_{\text{SPHINCS}} = (i, R_1, \sigma_{\text{HORST}}, \sigma_{\text{W-OTS}^+, 0}, \mathbf{auth}_0, \dots, \sigma_{\text{W-OTS}^+, d-1}, \mathbf{auth}_{d-1})$.

Figure 4: SPHINCS Extractor.

Lemma 4.3. *There exists such a size- ℓ set of queries $Q_{\text{W-OTS}^+}$ such that the signature $\sigma_{\text{W-OTS}^+}$ so produced in Step 6 is a valid signature for $\mathbf{root}_{d-2}$ with probability 1.*

Proof. We will show this using a similar technique as in the proof of Theorem 4.2, ie. using any key generator K for SPHINCS, we will construct a valid generator-signer pair for W-OTS⁺ termed $(K_{\text{W-OTS}^+}, S_{\text{W-OTS}^+})$.

1. $K_{\text{W-OTS}^+}$ runs the generation algorithm K and determines $\mathbf{ss} = \mathbf{ss}_{\text{SPHINCS}}$ and $\mathbf{pk}_{\text{SPHINCS}}$. Each time K requires a query to the oracle, $K_{\text{W-OTS}^+}$ makes the query and returns it to K .
2. Recall that $\mathbf{pk}_{\text{SPHINCS}}$ consists of $(\mathbf{root}, \mathbf{bitmasks})$ where $\mathbf{root}$ is the top node of the BHT generated by W-OTS⁺ public keys corresponding to each leaf of the $d-1^{\text{th}}$ tree. By Lemma 3.1, the query $H((\mathbf{pk}_{0,i} \| \mathbf{pk}_{0,i+1}) \oplus \mathbf{bitmasks}[\mathbf{BHT}_{0,d-1}]_0)$ for each $i \equiv 0 \pmod{2}$ was made to correctly determine $\mathbf{root}$ (otherwise the correctness of K would be negligible), where $\mathbf{pk}_{0,i}$ consists of the W-OTS⁺ public key corresponding to the i^{th} leaf of the top BHT. $K_{\text{W-OTS}^+}$ unhashes the corresponding $\mathbf{bitmasks}$ and learns $(\mathbf{ss}_{0,j}, \mathbf{pk}_{0,j})$ for some j , where $\mathbf{ss}_{0,j}$ is a valid signing state for some W-OTS⁺ signer S' .
3. $K_{\text{W-OTS}^+}$ outputs $\mathbf{ss}_{0,j}$ and $\mathbf{pk}_{0,j}$.
4. $S_{\text{W-OTS}^+}$ outputs $\sigma_{\text{W-OTS}^+} \leftarrow S'(\mathbf{ss}_{0,j}, m)$.

It is clear that the above is a valid generator-signer pair for W-OTS⁺ on the condition that *there exists some valid $(j, \mathbf{ss}_{0,j})$ extractable from K for which there is a signer S' for W-OTS⁺ that can sign with $\mathbf{ss}_{0,j}$* . Suppose that there is no such j with this property. Then taking any message m at random, the signer S associated with K will produce a valid SPHINCS signature on m with non-negligible probability. This signature will contain a subsignature $\sigma_{m,d-1}$ which verifies with some public key $\mathbf{pk}_{0,j}$ by definition. Here we will use restriction $(*)$. Since K cannot perform any processing beyond $\mathbf{BST}_{0,d-1}$, it does not know in

advance the value of root_{d-2} to be signed by S ; any guess will succeed with probability $1/2^\lambda$ at best since root_{d-2} is determined as the output of a series of oracle queries. Therefore S must run a subalgorithm which produces a valid W-OTS^+ signature on any message $m \in \{0, 1\}^\lambda$. The state taken as input by this subalgorithm is generated by K and is set as $\text{ss}_{0,j}$.

We now have shown a valid W-OTS^+ generator-signer pair. Hence the universal W-OTS^+ extractor $\mathcal{A}_{\text{W-OTS}^+}$ applies, and can extract a signature using some set of queries $Q_{\text{W-OTS}^+}$ on any message of its choice with probability 1. $\square$

Lemma 4.4. *There exists some set Q_K of queries made by K which contains the authentication path of i for every leaf i in $\text{BST}_{0,d-1}$.*

Proof. Recall that K outputs root as part of the public key. Then the required set is guaranteed by Lemma 3.1. $\square$

Lemma 4.5. *The signature σ_{SPHINCS} produced by adversary $\mathcal{A}_K$ of Fig. 4 verifies with probability at least $1/2^{h/d}$.*

Note that $2^{h/d} = O(\text{poly}(\lambda))$. Hence the lemma shows that the above adversary succeeds with at least inverse polynomial probability, which is enough to satisfy the definition of Definition 4.6.

Proof. Recall the verification algorithm of SPHINCS . The algorithm begins by checking whether σ_{HORST} is a valid HORST signature on D . This check succeeds since σ_{HORST} is indeed an honestly generated HORST signature. Similarly, the authentication checks succeed until layer $d - 1$. Furthermore, checking whether $\sigma_{\text{W-OTS}^+, d-1}$ is a valid signature of root_{d-2} succeeds due to Lemma 4.3. Further, by Lemma 4.4, the authentication check succeeds provided that the provided leaf $\text{Node}_{0,j}$ of $\text{BST}_{0,d-1}$ was indeed the leaf chosen by R_2 . Recall that R_2 is selected at random. Then the probability that any particular leaf of $\text{BST}_{0,d-1}$ is chosen by R_2 is uniform. Since there are $2^{h/d}$ such leaves, the probability is $1/2^{h/d}$. $\square$

Removing the restriction (*). We now show how the above proof can be adapted to generator-signer pairs (K, S) where K performs arbitrary computation on the SPHINCS tree. This can be shown via a simple counting argument: since SPHINCS is a stateless signature scheme, the total number of leaves of the SPHINCS tree is exponential. Meanwhile, K can only perform a polynomial amount of computation. Suppose K performs $p(\lambda)$ steps of computation. Consider giving S $p(\lambda)$ messages to sign and then giving it an additional message m chosen uniformly at random. Then the final signature will correspond to some HORST node previously untouched by K , and consider the complete path from the node to the root of $\text{BST}_{0,d-1}$. The final signature consists of a series of d W-OTS^+ signatures that sign the root of the tree below. Consider the highest-level leaf of an internal BHT on this path which has not been used before for signing the $p(\lambda)$ signatures. Since signing on m succeeds with all but negligible probability, and since the root value to sign was previously undetermined, this particular leaf must be associated with a key pair used to sign any message. Furthermore, since the root associated with this leaf has been determined, K previously made a query that determined an associated public key for the W-OTS^+ key pair of this leaf by Lemma 3.1. Hence the argument of Lemma 4.3 applies to this particular W-OTS^+ key pair, and we can adjust the adversary such that instead of selecting R_2 uniformly at random it ensures that signature generation passes through the aforementioned key pair, and the proof is immediate.

To show that the minimum extraction size is $\ell + h/d$, we can apply the same construction algorithm as that of XMSS . $\square$

Since SPHINCS^+ carries forward the same hypertree structure as that of SPHINCS (the major changes being the usage of FORS instead of HORST as the few-time signature scheme used for bottom-level signing and the use of a tweakable hash), the following corollary follows immediately.

Corollary 4.4. *SPHINCS^+ is 1-extractable with extraction size $\ell + h/d$.*

5 Black-Box Distributed Key Generation is Impossible

In this section we show our main result, ie. that any extractable signature scheme in the ROM does not support distributed key generation. We first begin by defining our scope of distributed key generation, in the vein of [DKR24].

5.1 Distributed Key Generation in the ROM

We wish to limit ourselves to protocols which are oracle-respecting, ie. in which the parties have only black-box access to the random oracle. We will let the output of the key generation protocol to be a valid public key (in the clear) along with a shared signing state among n parties. The definition below is fairly broad: we do not limit ourselves merely to protocols that produce output shares of a secret key, but protocols that produce shares of any signing state that can be used to produce a valid signature.

Definition 5.1 (Distributed Key Generation in the ROM). *Let $\Pi = (\Pi.\text{KeyGen}, \Pi.\text{Sign}, \Pi.\text{Vrfy})$ be any signature scheme in the random oracle model. Let $\pi_{\text{DKG}}^n(1^\lambda)$ be an expected-PPT interactive n -party protocol in which all parties have oracle access to H . We say that π is a valid n -party ROM-DKG for Π if there exists a PPT secret sharing function with relativizing reconstruction¹ $(\text{Share}, \text{Reconstruct})$ where $\text{Share}_{n,\lambda} : \Pi.\text{SKSpace}_\lambda \rightarrow \mathbb{S}_\lambda^n$ along with a valid generator-signer pair (K, S) for Π such that*

$$\{(\text{Reconstruct}(\vec{ss}), \text{pk}) : (\vec{ss}, \text{pk}) \leftarrow \pi_{\text{DKG}}^n(1^\lambda)\} \cong \{(\text{ss}, \text{pk}) : (\text{ss}, \text{pk}) \leftarrow K(1^\lambda)\},$$

ie. π_{DKG}^n correctly realizes K .

Note that the above definition makes no requirement of *security*: our result in fact applies to any oracle-respecting protocol which correctly realizes any generating algorithm K . We next define the notion of forgery.

Definition 5.2 (Forgery with t -out-of- n corruptions). *Let Π be a signature scheme in the ROM, and let π_{DKG}^n be any ROM-DKG protocol for Π with associated generator-signer pair (K, S) . Furthermore, let $\text{View}(1^\lambda)$ be any algorithm that runs $\pi_{\text{DKG}}^n(1^\lambda)$ and then outputs $\{\mathbf{v}_i\}_{i \in [n]}$ where $\mathbf{v}_i$ is the view of the i^{th} party. Then we say that π_{DKG}^n allows forgery with t -out-of- n corruptions if there exists some PPT algorithm $\mathcal{A}_{\text{DKG}}$ such that*

$$\Pr \left[\begin{array}{l} \mathbf{C} \leftarrow 2^{[n]} : |\mathbf{C}| = t \\ \mathbf{v} \leftarrow \text{View}(1^\lambda) \\ (m, \sigma) \leftarrow \mathcal{A}_{\text{DKG}}(1^\lambda, \text{pk}, \{(i, \mathbf{v}_i)\}_{i \in \mathbf{C}}) \end{array} \right] \geq \frac{1}{\text{poly}(\lambda)}.$$

5.2 Impossibility of Key Generation

We now show our main result, presented below.

Theorem 5.1. *Let Π be an extractable signature scheme with a constructable extraction set that supports minimum extractable size m . Then for every constant k and $n = m + k$, any n -party ROM-DKG for Π allows forgery with m -out-of- n corruptions.*

Proof. Let π_{DKG}^n be an n -party ROM-DKG for Π . Note that π_{DKG}^n has an associated generator-signer pair; call it (K, S) . Consider the following alternate generator K' , presented in Fig. 5.

If $(\text{Share}, \text{Reconstruct})$ is a correct secret sharing scheme, then it is clear that (K', S) is a valid generator-signer pair for Π , since the output of K' is identically distributed to that of K . Since Π is extractable, it immediately follows that there exists some extractor $\mathcal{A}_{K'}$ which takes in as input $(\mathbf{Q}_{K'}, \text{pk})$ and produces a valid message-signature pair (m, σ) with non-negligible probability, where $\mathbf{Q}_{K'}$ is a set of queries made

¹We say that any secret sharing scheme has *relativizing reconstruction* if the **Reconstruct** algorithm (which takes as input some portion of shares and correctly produces the secret value) does not require access to the random oracle. This is a reasonable restriction that every secret sharing scheme in the literature satisfies.

K' takes as input only 1^λ .

1. K' sets up n ‘imaginary’ parties $\{P_i\}_{i \in [n]}$ in the head.
2. K' runs $(\vec{ss}, pk) \leftarrow \pi_{\text{DKG}}^n(1^\lambda)$ among the imaginary parties $\{P_i\}_{i \in [n]}$. Each time some party P_i requires a random oracle query, K' makes the query and returns the output to P_i .
3. K' outputs $(\text{Reconstruct}(\vec{ss}), pk)$.

Figure 5: An Alternate Generator K' .

by K' while producing (ss, pk) of size at most m . Furthermore, note that all queries made by K' were in fact made during Step 2, which consisted of a single evaluation of π_{DKG}^n (here we use the fact that the secret-sharing scheme has relativizing reconstruction, and hence no queries are made during Step 3). Hence the query set $\mathbf{Q}_{K'}$ consists of a set of queries made by some group of P_i .

Now consider a valid evaluation of π_{DKG}^n among parties $\{P_i\}_{i \in [n]}$. We note that since π_{DKG}^n is a ROM-DKG, it follows that there exists some (at most size- m) set of P_i which make the aforementioned set of queries $\mathbf{Q}_{K'}$ that the extractor for K' can use to extract a signature. With this in mind, we consider the following m -out-of- n forger $\mathcal{A}_{\text{DKG}}$.

1. $\mathcal{A}_{\text{DKG}}$ parses its input as $(pk, \{(i, \mathbf{v}_i)\}_{i \in \mathbf{C}})$. It initializes a set $\mathbf{Q}'$ and goes through $\mathbf{v}_i$ for each $i \in \mathbf{C}$. Each time it encounters a query q as part of $\mathbf{v}_i$ for any i , it adds q to $\mathbf{Q}'$.
2. $\mathcal{A}_{\text{DKG}}$ runs the algorithm $\{Q, \perp\} \leftarrow \text{Construct}_{K'}(\mathbf{Q}', pk)$. If the output is $\perp$ the adversary outputs $(0, 0)$.
3. $\mathcal{A}_{\text{DKG}}$ runs $(m, \sigma) \leftarrow \mathcal{A}_{K'}(Q, pk)$ and outputs (m, σ) .

Claim 5.1. $\mathcal{A}_{\text{DKG}}$ succeeds with noticeable probability.

Proof. Note that $\text{Construct}_{K'}$ succeeds with overwhelming probability if $\mathbf{Q}'$ contains an extraction set, and $\mathcal{A}_{K'}$ outputs a valid-message signature pair with probability at least $\frac{1}{\text{poly}(\lambda)}$. Let us now calculate the probability that $\mathbf{Q}'$ does in fact contain a minimum extractable set. By the previous argument, we had seen that the complete set of queries $\mathbf{Q}_{K'}$ made by the protocol contains a set of queries of size m that can be used to extract. Let the set be $\{q_i\}_{i \in [m]}$. Then each q_i was made by some party P_i . There are m such parties at most. Hence, since $\mathbf{C}$ is chosen at random, the probability that the adversary obtains the views of every such party P_i is at least

$$\frac{1}{\binom{n}{m}} = \frac{m!(n-m)!}{n!} = \frac{m!k!}{(m+k)!} = \frac{k!}{\prod_{i=1}^k (m+i)} = O\left(\frac{1}{\text{poly}(m)}\right)$$

where the last equality follows from the fact that k is a constant. Hence, we can calculate the total probability of success as at least

$$O\left(\frac{1}{\text{poly}(\lambda)}\right) \times \left(1 - \frac{1}{\text{negl}(\lambda)}\right) = O\left(\frac{1}{\text{poly}(\lambda)}\right).$$

□

From the previous claim, we can conclude that $\mathcal{A}_{\text{DKG}}$ is a successful forger for π_{DKG}^n , and we are done. □

6 What About 2-Party Signing?

In the previous sections we have shown the impossibility of black-box distributed signing of extractable hash-based signatures, in the setting where all but a constant number of parties is corrupted. More

precisely, we have shown that for each scheme there exists some number of parties n beyond which no secure distributed protocol for key generation exists. In this section we show that this problem is not mitigated for lower party sizes.

6.1 MPC for W-OTS⁺

We consider the problem of 2-Party Key Generation for the W-OTS⁺ signature scheme. By Theorem 5.1 and Lemma 4.1 we can conclude that there exists no black-box distributed protocol in the $(m+1)$ -out-of- m corruptions setting that securely generates W-OTS⁺ keys where $m > \ell$. We now show that in the specific case of W-OTS⁺ we are liable to get serious security losses even when attempting a black-box protocol for any $1 < m \leq \ell$. To start with, we consider the following simple lemma, which is key to the remainder of the section.

Lemma 6.1. *Let $\pi_{\text{W-OTS}^+}^m$ be any m -party protocol for distributed W-OTS⁺ key generation that outputs $(\vec{\text{sk}}, \text{pk})$ where sk has been shared among the m parties using a secret sharing scheme with relativizing reconstruction and which makes black-box use of the random oracle. Then with noticeable probability an $(m-1)$ -out-of- m adversary that corrupts parties at random learns at least $\ell_1 - \lfloor \ell_1/m \rfloor$ distinct q_i such that $c^{w-1}(q_i, \text{pk}_0) = \text{pk}_i$ for $1 \leq i \leq \ell_1$.*

Proof. Since W-OTS⁺ is extractable, the argument in the proof of Theorem 5.1 shows that $\pi_{\text{W-OTS}^+}^m$ must necessarily make the full ℓ -size set of queries that comprise the constructable extraction set of Theorem 4.1. The first ℓ_1 of these queries comprise the preimages of the message chain, while the final ℓ_2 comprise the preimages of the checksum chain. By the pigeonhole principle, there exists some party P_0 which makes at most $\lfloor \ell_1/m \rfloor$ -many of the first ℓ_1 queries. Thus, the other $m-1$ parties make at least $\ell_1 - \lfloor \ell_1/m \rfloor$ of the queries. The adversary learns all these queries with at least $1/m$ probability, which is inverse polynomial in λ . $\square$

Intuitively, the above lemma states that any adversary corrupting a large-enough number of parties learns a significant set of message chain preimages. For simplicity, we assume that it learns the *first* $\ell_1 - \lfloor \ell_1/m \rfloor$ such preimages; if not, the proof follows accordingly.

Asymptotic Analysis. We now show that there exists a PPT adversary which, given this set of preimages, can substantially degrade the concrete security level corresponding to the 1-Existential Unforgeability of W-OTS⁺. Taken together, the two results can be interpreted as showing that even 2-party black-box distributed key generation leaks enough key information to the adversary so as to result in a significant security loss.

Theorem 6.1. *There exists an adversary $\mathcal{A}$ against the 1-existential unforgeability of W-OTS⁺ with auxiliary information $\text{aux} = \{q_i\}_{i \in [\ell_1 - \lfloor \ell_1/m \rfloor]}$ such that $c^{w-1}(q_i, \text{pk}_0) = \text{pk}_i$ which succeeds with probability at least*

$$\frac{1}{w^{\lfloor \ell_1/m \rfloor} [(\ell_1 - \lfloor \ell_1/m \rfloor)(w-1) + 1]}$$

Proof. We define the adversary in Figure 6.

We now calculate the success probability $\text{AdvDKG}_{\mathcal{A}}.$ of $\mathcal{A}$. This is slightly involved, so we will split it into a number of lemmas. We define $D_i = m_i^1 - m_i^0$ over $\mathbb{Z}$.

Lemma 6.2. *The adversary succeeds iff (1) $D_i \geq 0$ for all $\ell_1 - \lfloor \ell_1/m \rfloor \leq i \leq \ell_1$, (2) $D = \sum_{i=0}^{\ell_1} D_i \leq 0$, and when the addition $-D + C_0$ is done in base w over $\mathbb{Z}$, there are no carry-overs.*

For each key pair $(\mathbf{sk}, \mathbf{pk})$ output by the key generation oracle, we set $\mathbf{aux} = \{q_i\}_{i \in [\ell_1 - \lfloor \ell/m \rfloor]}$ such that $c^{w-1}(q_i, \mathbf{pk}_0) = \mathbf{pk}_i$.

1. $\mathcal{A}$ queries the key generation oracle and receives $(\mathbf{pk}, \mathbf{aux})$ as output.
2. $\mathcal{A}$ parses $\mathbf{aux} := \{q_i\}_{i \in [\ell_1 - \lfloor \ell/m \rfloor]}$. It samples two messages m'_0 and m'_1 uniformly at random and calculates $m_0 := H(m'_0)$ and $m_1 := H(m'_1)$. Then it sets $m_i = (m_1^i, \dots, m_{\ell_1}^i)$ and computes $C_i = \sum_{j=0}^{\ell_1} (w-1 - m_j^i)$. It sends m_0 to the signing oracle.
3. $\mathcal{A}$ receives $\sigma := (\sigma_0, \sigma_1)$. It parses $\sigma_0 := (s_1, \dots, s_{\ell_1})$ and $\sigma_1 := (c_1, \dots, c_{\ell_2})$.
4. $\mathcal{A}$ sets $C_i := (b_1^i, \dots, b_{\ell_2}^i)$. It checks if $b_j^0 \leq b_j^1$ for all $j \leq \ell_2$. If not, it outputs $\perp$ and terminates.
5. $\mathcal{A}$ checks if $m_j^0 \leq m_j^1$ for all $\ell_1 - \lfloor \ell_1/m \rfloor < j \leq \ell_1$. If not, it outputs $\perp$ and terminates.
6. If both the checks pass, $\mathcal{A}$ first computes $s'_i := c^{m_i^1}(q_i, \mathbf{pk}_0)$ for each $0 < i \leq \ell_1 - \lfloor \ell_1/m \rfloor$. Then it computes $s'_i := c^{m_i^1 - m_i^0}(s_i, \mathbf{pk}_0)$ for $\ell_1 - \lfloor \ell_1/m \rfloor < i \leq \ell_1$. Finally, it computes $c'_i := c^{m_i^0 - m_i^1}(c_i, \mathbf{pk}_0)$ for all $0 < i \leq \ell_2$.
7. $\mathcal{A}$ outputs $(m'_1, (\{s_i\}_{i \in [\ell_1]} \parallel \{c_i\}_{i \in [\ell_2]}))$.

Figure 6: Key Leakage Attack Against W-OTS⁺.

Proof. The first condition is just the check in Step 5 restated. For the second condition, notice that

$$\begin{aligned} C_1 - C_0 &= \sum_{i=0}^{\ell_1} (w-1 - m_i^1) - \sum_{i=0}^{\ell_1} (w-1 - m_i^0) \\ &= \ell_1(w-1) - \ell_1(w-1) + \sum_{i=0}^{\ell_1} (m_i^0 - m_i^1) \\ &= \sum_{i=0}^{\ell_1} -D_i. \end{aligned}$$

Now, since we require $b_i^1 \geq b_i^0$ for each i , we also have $\sum_{i=0}^{\ell_1-1} (b_i^1 - b_i^0)w^i \geq 0$. But this is just $C_1 - C_0$, and hence we have that $\sum_{i=0}^{\ell_1-1} -D_i \geq 0$ which implies that $D \leq 0$, and so we have justified the second condition, since $b_i^1 \geq b_i^0$ implies there are no carry-overs in the addition. The other direction is immediate: the fact that there are no carry-overs in the addition implies $b_i^1 \geq b_i^0$ iff $D \leq 0$. $\square$

We note here that if $m = 1$, which is the case where there is only a single signer, then nothing is leaked and the condition reduces to $D_i = 0$ for each 0 , ie. the only signature the adversary can ‘forge’ is the signature which has *already been signed*: there is no space over which to look for different messages, and the ‘attack’ trivializes and simply becomes a preimage guessing game which succeeds with probability $1/2^\lambda$.

Proposition 6.1. *If $D = 0$, then there are no carry-overs in base w .*

From the above proposition we note that we can lower bound $\text{AdvDKG}_{\mathcal{A}}$ as $\text{AdvDKG}_{\mathcal{A}} \geq \Pr[\text{Condition } 1 \wedge D = 0]$, and hence we will focus on the latter probability. Let there be $\mathbf{a}$ constrained D_i (ie. the coordinates the adversary does not possess) and there be $\ell_1 - \mathbf{a}$ ‘free’ D_i .

Proposition 6.2. *If (1) $D_i = 0$ for all the $\mathbf{a}$ constrained coordinates and (2) $\sum_{i=0}^{\ell_1 - \mathbf{a}} m_i^0 = \sum_{i=0}^{\ell_1 - \mathbf{a}} m_i^1$ for all the $\ell_1 - \mathbf{a}$ free coordinates, then $D = 0$.*

The above proposition is just a subset of the event that $D = 0$, so functions as a lower bound. Note that (1) and (2) are independent, as each of the coordinates is independently chosen. Now we can easily calculate the probability of (1) as $(\frac{1}{w})^{\mathbf{a}}$, which follows simply from the fact that the probability that both digits chosen is the same is exactly $1/w$ (these are all independent events).

Now let $S_0 = \sum_{i=0}^{\ell_1 - \mathfrak{a}} m_i^0$ and $S_1 = \sum_{i=0}^{\ell_1 - \mathfrak{a}} m_i^1$. Since the m_i^j range from $[0..w-1]$, we have

$$0 \leq \sum_{i=0}^{\ell_1 - \mathfrak{a}} m_i^j \leq (\ell_1 - \mathfrak{a})(w-1)$$

and hence S_i has at most $(\ell_1 - \mathfrak{a})(w-1) + 1$ elements. Let $s \in [0..(\ell_1 - \mathfrak{a})(w-1)]$ and p_s be the probability that the sum is s . We have that

$$\Pr[S_0 = s] = \Pr[S_1 = s] = p_s^2 \implies \Pr[S_0 = S_1] = \sum_{s=0}^{(\ell_1 - \mathfrak{a})(w-1)} p_s^2.$$

Using Cauchy-Schwarz with the variables p_s and 1, we have that $(\sum_s p_s \cdot 1)^2 \leq (\sum_s p_s^2)(\sum_s 1^2)$, and hence we can bound the above as

$$\sum_{s=0}^{(\ell_1 - \mathfrak{a})(w-1)} p_s^2 \geq \frac{\left(\sum_{s=0}^{(\ell_1 - \mathfrak{a})(w-1)} p_s\right)^2}{(\ell_1 - \mathfrak{a})(w-1) + 1} = \frac{1}{(\ell_1 - \mathfrak{a})(w-1) + 1}.$$

It immediately follows that

$$\text{AdvDKG}_{\mathcal{A}} \geq \frac{1}{w^{\mathfrak{a}}[(\ell_1 - \mathfrak{a})(w-1) + 1]} = \frac{1}{w^{\lfloor \ell_1/m \rfloor}[(\ell_1 - \lfloor \ell_1/m \rfloor)(w-1) + 1]}. \quad (1)$$

With this we are done with the theorem. For bookkeeping, we can additionally bound the expectation of the runtime required to forge a signature with probability at least $1/2$. By setting $\text{AdvDKG}_{\mathcal{A}} = 1/x$, we note that this probability is equal to the number of bernoulli trials before the probability of success is at least 2 , and hence the number of trials for large x can be taken to be $x \log 2$. Setting the runtime per trial (which is dominated by Step 6) as $(\ell_1 + \ell_2)w$, we get a total running time of

$$\lceil \log 2(\ell_1 + \ell_2)w^{\lfloor \ell_1/m \rfloor + 1}[(\ell_1 - \lfloor \ell_1/m \rfloor)(w-1) + 1] \rceil.$$

□

Security Loss. We note that the above theorems immediately imply that any adversary corrupting m parties for distributed W-OTS^+ key generation can break the 1-existential unforgeability of W-OTS^+ with probability given above times an additional $1/m$ factor in the all-but-one corruptions setting. Set $k = \ell_1 / \log w$. Then we can take the negative logarithm of $\text{AdvDKG}_{\mathcal{A}}$ to get that the security loss is at least

$$\frac{k}{m} + O(\log k + \log m + \log w),$$

which is asymptotically a constant-fraction security loss.

Concrete Analysis. We can now calculate the actual success probability $\text{AdvDKG}_{\mathcal{A}}$ for various values of m and for standard W-OTS^+ parameters $w = 16, \ell_1 = 64, \ell_2 = 3$. Taking $m = 2$, we can plug the values into Eq. (1) to get

$$\text{AdvDKG}_{\mathcal{A}} \geq \frac{1}{2 \cdot 16^{32} \cdot 481} > 2^{-138}$$

which essentially halves the 256-bit security of the specified parameters.

We can do even better, by presenting an improved analysis of the case where $D = 0$ of Theorem 6.1.

Our analysis works as follows. We note that for each $d \in [0..w-1]$, the number of pairs m_0, m_1 such that $m_1 - m_0 = d$ is exactly $w - |d|$. Using this, we can define a *difference-counting polynomial*

$$F_w(z) = \sum_{d=-(w-1)}^{w-1} (w - |d|)z^d,$$

which works as follows: for each d , the coefficient of z^d counts the number of possible digit pairs which have a difference of d . Note that for the $\mathbf{a} = \lfloor \ell_1/m \rfloor$ constrained coordinates we have that $D_i \geq 0$, which means that the set of possible differences is $[0..w-1]$, and hence we only have to iterate over $w-1$ possibilities, each of which have a number of pairs equivalent to $w-d$. Hence the difference counting polynomial of this set can be defined as

$$G_w(z) = \sum_{d=0}^{w-1} (w-d)z^d.$$

Note that since there are $\mathbf{a}$ constrained coordinates and $\ell_1 - \mathbf{a}$ free coordinates, the polynomial that enumerates the total number of digit pairs is $F_w(z)^{\mathbf{a}}G_w(z)^{\ell_1 - \mathbf{a}}$. Let us consider the monomials of this polynomial. Note that if $\sum_{i=0}^{\ell_1} D_i = D$, then expanding the system above we get that the coefficient of the formed $z^{\sum_{i=0}^{\ell_1} D_i}$ term is going to be $(w - |D_0|) \dots (w - D_\ell)$, which counts exactly the number of digit pairs which simultaneously sum up to the D_i respectively. Now note that the coefficient of z^D in $F_w(z)^{\mathbf{a}}G_w(z)^{\ell_1 - \mathbf{a}}$ ranges over *all* possible joint values of the D_i which sum up to D . Hence, we can conclude that the coefficient of the monomial of z^0 (represented as $[z^0]$) enumerates the number of pair values which gives $D = 0$. Since the number of total possible digit pairs that do not have any constraints is precisely $w^{2\ell_1}$, we can calculate that

$$\text{AdvDKG}_{\mathcal{A}} \geq \Pr[D = 0] = \frac{[z^0]F_w(z)^{\lfloor \ell_1/m \rfloor} G_w(z)^{\ell_1 - \lfloor \ell_1/m \rfloor}}{w^{2\ell_1}}. \quad (2)$$

Note that the term on the right is not actually a closed-form expression, and hence is unsuitable as a bound. Nevertheless, for small values of m, ℓ_1, ℓ_2 and w , the term in Eq. (2) can be computed effectively using dynamic programming. For standard 256-bit W-OTS⁺ parameters, we tabulate the success probability of our attack in Table 1 (for $m > 67$, we revert to the setting of Theorem 5.1 and there is an attack against W-OTS⁺ DKG that succeeds with probability 1).

Number of parties m	Overall success probability	Runtime for success probability $> 1/2$
2	$> 2^{-45}$	$\approx 2^{55}$
3	$> 2^{-28}$	$\approx 2^{38}$
4	$> 2^{-22}$	$\approx 2^{32}$
6	$> 2^{-16}$	$\approx 2^{26}$
12	$> 2^{-11}$	$\approx 2^{21}$
32	$> 2^{-10}$	$\approx 2^{20}$

Table 1: Success bounds.

6.2 MPC for Other Schemes

The above partial key leakage attack can immediately be applied to schemes which make use of W-OTS⁺ as an interim, such as XMSS and SPHINCS/SPHINCS+.

References

- [ABCP23] Shahla Atapoor, Karim Bagheri, Daniele Cozzo, and Robi Pedersen. CSI-SharK: CSI-FiSh with sharing-friendly keys. In Leonie Simpson and Mir Ali Rezazadeh Bae, editors, *ACISP 23*, volume 13915 of *LNCS*, pages 471–502. Springer, Cham, July 2023.
- [AdSGK24] Shahla Atapoor, Cyprien Delpech de Saint Guilhem, and Al Kindi. Stark-based signatures from the RPO permutation. *IACR Cryptol. ePrint Arch.*, page 1553, 2024.
- [BBD⁺23] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST. Technical report, National Institute of Standards and Technology, 2023. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures>.
- [BBD⁺26] Dan Boneh, Benedikt Bünz, Justin D., Yavor Litchev, and Kamilla Nazirkhanova. PRAWNS: Threshold hash-based signatures from threshold PRFs. *Cryptology ePrint Archive*, Paper 2026/2142, 2026.
- [BCdP⁺25] Giacomo Borin, Sofía Celi, Rafaël del Pino, Thomas Espitau, Guilhem Niot, and Thomas Prest. Threshold signatures reloaded: ML-DSA and enhanced raccoon with identifiable aborts. *IACR Cryptol. ePrint Arch.*, page 1166, 2025.
- [BdCE⁺25] Alexander Bienstock, Leo de Castro, Daniel Escudero, Antigoni Polychroniadou, and Akira Takahashi. Efficient, scalable threshold ML-DSA signatures: An MPC approach. *IACR Cryptol. ePrint Arch.*, page 1163, 2025.
- [BDE⁺11] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. On the security of the Winternitz one-time signature scheme. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 363–378. Springer, Berlin, Heidelberg, July 2011.
- [BDH11] Johannes A. Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS - A practical forward secure signature scheme based on minimal security assumptions. In Bo-Yin Yang, editor, *PQCrypto 2011*, pages 117–129. Springer, Berlin, Heidelberg, November / December 2011.
- [BGG⁺18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Cham, August 2018.
- [BHH⁺15a] 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 Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 368–397. Springer, Berlin, Heidelberg, April 2015.
- [BHH⁺15b] 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*, page 368397. Springer Berlin Heidelberg, 2015.
- [BHK⁺19] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS⁺ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2129–2146. ACM Press, November 2019.
- [Bon26] Dan Boneh. How Bitcoin survives quantum computers (ft. Dan Boneh). YouTube video, a16z crypto, August 2026. Published August 22, 2026.

- [CAD⁺20] David A. Cooper, Daniel C. Apon, Quynh H. Dang, Michael S. Davidson, Morris J. Dworkin, and Carl A. Miller. *Recommendation for Stateful Hash-Based Signature Schemes*. October 2020.
- [CEN25] Sofia Celi, Daniel Escudero, and Guilhem Niot. Share the MAYO: thresholdizing MAYO. In Ruben Niederhagen and Markku-Juhani O. Saarinen, editors, *Post-Quantum Cryptography - 16th International Workshop, PQCrypto 2025, Taipei, Taiwan, April 8-10, 2025, Proceedings, Part I*, volume 15577 of *Lecture Notes in Computer Science*, pages 165–198. Springer, 2025.
- [CS20] Daniele Cozzo and Nigel P. Smart. Sashimi: Cutting up CSI-FiSh secret keys to produce an actively secure distributed signing protocol. In Jintai Ding and Jean-Pierre Tillich, editors, *PQCrypto 2020*, pages 169–186. Springer, Cham, 2020.
- [Des88] Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *CRYPTO’87*, volume 293 of *LNCS*, pages 120–127. Springer, Berlin, Heidelberg, August 1988.
- [DKLs18] Jack Doerner, Yashvanth Kondi, Eysa Lee, and abhi shelat. Secure two-party threshold ECDSA from ECDSA assumptions. In *2018 IEEE Symposium on Security and Privacy*, pages 980–997. IEEE Computer Society Press, May 2018.
- [DKM⁺24] Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 219–248. Springer, Cham, May 2024.
- [DKR24] Jack Doerner, Yashvanth Kondi, and Leah Namisa Rosenbloom. Sometimes you can’t distribute random-oracle-based proofs. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part V*, volume 14924 of *LNCS*, pages 323–358. Springer, Cham, August 2024.
- [DOTT22] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):14, April 2022.
- [GG18] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1179–1194. ACM Press, October 2018.
- [GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. Threshold-optimal DSA/ECDSA signatures and an application to Bitcoin wallet security. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 2016*, volume 9696 of *LNCS*, pages 156–174. Springer, Cham, June 2016.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
- [Gol04] Oded Goldreich. *Foundations of Cryptography: Basic Applications*, volume 2. Cambridge University Press, Cambridge, UK, 2004.
- [HBG⁺18] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: eXtended Merkle Signature Scheme. RFC 8391, May 2018.
- [Hül13] Andreas Hülsing. W-OTS+ - shorter signatures for hash-based signature schemes. In Amr Youssef, Abderrahmane Nitaj, and Aboul Ella Hassanien, editors, *AFRICACRYPT 13*, volume 7918 of *LNCS*, pages 173–188. Springer, Berlin, Heidelberg, June 2013.

- [KLL25] John Kelsey, Nathalie Lang, and Stefan Lucks. Turning hash-based signatures into distributed signatures and threshold signatures: Delegate your signing capability, and distribute it among trustees. *CiC*, 2(2):24, 2025.
- [Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.
- [Lin17] Yehuda Lindell. Fast secure two-party ECDSA signing. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 613–644. Springer, Cham, August 2017.
- [LN18] Yehuda Lindell and Ariel Nof. Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 1837–1854. ACM Press, October 2018.
- [NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st ACM STOC*, pages 33–43. ACM Press, May 1989.
- [Pas03] Rafael Pass. On deniability in the common reference string and random oracle model. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 316–337. Springer, Berlin, Heidelberg, August 2003.
- [PN25] Rafaël Del Pino and Guilhem Niot. Finally! A compact lattice-based threshold signature. In Tibor Jager and Jiaxin Pan, editors, *Public-Key Cryptography - PKC 2025 - 28th IACR International Conference on Practice and Theory of Public-Key Cryptography, Røros, Norway, May 12-15, 2025, Proceedings, Part III*, volume 15676 of *Lecture Notes in Computer Science*, pages 169–199. Springer, 2025.
- [Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *22nd ACM STOC*, pages 387–394. ACM Press, May 1990.
- [RR02] Leonid Reyzin and Natan Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP 02*, volume 2384 of *LNCS*, pages 144–153. Springer, Berlin, Heidelberg, July 2002.
- [Wil90] Herbert S. Wilf. *Generatingfunctionology*. Academic Press, 1990.
- [ZCD⁺20] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladimir Kolesnikov, and Daniel Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>.