Clarifying the subset-resilience problem

Jean-Philippe Aumasson[1] and Guillaume Endignoux[2]

[1] Kudelski Security, Switzerland
[2] firstname.surname@m4x.org

Abstract. We investigate the *subset-resilience* problem, defined in 2002 by Reyzin and Reyzin to analyze their HORS signature scheme. We show that textbook HORS is insecure against adaptive attacks, and present a practical attack based on a greedy algorithm.
We also describe *weak messages* for HORS, that map to smaller subsets than expected, and are thus easier to cover. This leads to an improved attack against HORS and to an improved classical attack against the signature scheme SPHINCS, of complexity 2^{270} instead of 2^{277}. We propose the *PRNG to obtain a random subset* construction (PORS), which avoids weak messages, for a tiny computational overhead.
We adapt PORS to SPHINCS to also deterministically select the HORST instance that is used to sign the input message. This new construction reduces the attack surface and increases the security level, improving the security of SPHINCS by 67 bits against classical attacks and 33 bits against quantum attacks. A version of SPHINCS using our PORS construction can work with smaller parameters that reduce the signature size by 4616 bytes and speed up signature and verification, for the same 128-bit post-quantum security as the original SPHINCS.

1 Introduction

In 2002, Reyzin and Reyzin [10] presented HORS, a few-time signature scheme based on the *hash to obtain a random subset* construction. This construction uses a hash function to select a subset of a finite set; the signature is then derived from this subset. The security of HORS relies on the *subset-resilience* of the hash function, a property defined by Reyzin and Reyzin for the purpose of this scheme. Several signature schemes are variants of this idea, e.g. HORS++ [9] and HORST [3]. The latter is an essential building block of SPHINCS, a practical stateless hash-based signature scheme [3].

However, the *subset-resilience* problem was only partially studied; Reyzin and Reyzin proposed two security notions to capture the security against both adaptive and non-adaptive attacks, but they only studied non-adaptive attacks. This paper investigates adaptive attacks and shows that security decreases dramatically when HORS is used more than once. We propose a greedy algorithm that breaks HORS and we give examples of forgeries.

We also use our observations to attack and improve SPHINCS. Contrary to textbook HORS, SPHINCS is not vulnerable to simple adaptive attacks because the hash is unpredictable to attackers – thanks to the use of a *salt*. Yet, we show

that both HORS and SPHINCS are susceptible to what we call *weak messages* attacks, due to how HORS maps messages to subsets. Indeed, HORS maps some messages to small subsets – when index collisions occur in the output of the hash function – and these small subsets are easier to cover. This yields an improved classical attack against SPHINCS, of complexity 2^{270} instead of 2^{277}. Nonethless, this new attack does not improve over known quantum attacks, based on Grover's search algorithm.

To eliminate weak message attacks we propose the PORS construction – *PRNG to obtain a random subset*. This new construction increases the security of HORS and SPHINCS against both classical and quantum attacks. In the case of SPHINCS, we extend PORS to also select the hyper-tree leaf, instead of having this leaf freely chosen by a potential forger. We show that with this last improvement, the security of SPHINCS increases to 171 bits (post-quantum). We therefore propose smaller parameters in order to save 4616 bytes per signature while guaranteeing 128-bit post-quantum security.

Roadmap. §2 introduces the subset-resilience problem, then §3 presents new bounds on adaptive attacks, and §4 reports on practical attacks against textbook HORS. Last, §5 reviews our new attacks against SPHINCS and §6 describes the integration of PORS in SPHINCS and benefits thereof.

2 The subset-resilience problem and HORS signatures

In this section, we recall known results about the subset-resilience problem.

2.1 General definitions

Given integer parameters k and τ such that $0 < k \leq 2^\tau$, we let $t = 2^\tau$ and denote by T the set $\{0, \ldots, t-1\}$. We denote by $\mathcal{P}_k(T)$ the set of subsets of T of size at most k. We consider a finite key space $\mathcal{K}$, a message space $\mathcal{M}$ and a family $\mathcal{O}_n$ of functions to *obtain random subsets* (ORS) from messages $\mathcal{O}_n = \{\mathsf{ORS}_K : \mathcal{M} \to \mathcal{P}_k(T) | K \in \mathcal{K}\}$. Given a key K and $r+1$ messages $M_1, \ldots, M_{r+1}$ we define the *r-subsets-cover* relation C_K as:

$$C_K(M_1, \ldots, M_{r+1}) \Leftrightarrow \mathsf{ORS}_K(M_{r+1}) \subseteq \bigcup_{j=1}^{r} \mathsf{ORS}_K(M_j)$$

which means that the image of M_{r+1} by ORS_K is covered by the images of $M_1, \ldots, M_r$.

2.2 Hash to obtain a random subset

Except in §6, we restrict our analysis to function families $\mathcal{O}_n$ that follow the *hash to obtain a random subset* (HORS) construction, which we now describe. Given a

PRF $\mathcal{H}_n = \{H_K : \mathcal{M} \to \{0,1\}^{k\tau} | K \in \mathcal{K}\}$ that maps messages to $k\tau$-bit strings, the associated family $\mathcal{O}_n$ is constructed as follows. To compute $\mathsf{ORS}_K(M)$, take the string $x := H_K(M)$, split it into τ-bit blocks $x_1 || \ldots || x_k := x$ and interpret each block x_i as the encoding of a number $\overline{x_i} \in T$. The result $\mathsf{ORS}_K(M)$ is the subset $\mathsf{split}(x,k) := \{\overline{x_1}, \ldots, \overline{x_k}\}$.

We note that given a uniformly distributed key K and an arbitrary message M, the $\overline{x_i}$ are independent and uniformly distributed in T, because $\mathcal{H}_n$ is a PRF.

2.3 Few-time signature scheme

Given such a family $\mathcal{O}_n$ and a hash function family $\mathcal{F}_n$, you can construct a few-time signature scheme, as proposed by Reyzin and Reyzin [10] improving over Lamport [8] and Bos and Chaum [4]. To generate a key pair, select t random values $s_1, \ldots, s_t \in \{0,1\}^n$ for some security parameter n, along with a key $K \in \mathcal{K}$. The secret key is $(s_1, \ldots, s_t)$ and the public key is $(F(s_1), \ldots, F(s_t))$. To sign a message M, compute $\{x_1, \ldots, x_k\} = \mathsf{ORS}_K(M)$ and reveal the subset of secret values $\{s_{x_i} | 1 \leq i \leq k\}$.

Assuming that the one-way property of $\mathcal{F}_n$ is not broken, the security of this signature scheme reduces to the *subset-resilience problem*. Namely, given the signatures of r messages, an attacker can forge another message M_{r+1} if the associated $\mathsf{ORS}_K(M_{r+1})$ is covered by the hashes of the r messages, because the associated secret values have already been revealed.

Practical parameters. To give some intuition about the scheme, we recall practical choices of (k,t). For the original HORS scheme, Reyzin and Reyzin proposed $(k = 20, t = 256)$ or $(k = 16, t = 1024)$. For HORST as used in SPHINCS, Bernstein et al. proposed more conservative parameters $(k = 32, t = 2^{16})$.

2.4 Subset-resilience

Informally, $\mathcal{O}_n$ is r-subset-resilient if it is hard for an adversary to find $r+1$ messages that form a r-subsets-cover, under a key uniformly chosen in $\mathcal{K}$. Two adversarial scenarios were considered by Reyzin and Reyzin [10]. In the adaptive scenario, an adversary is given a key K and can compute ORS_K on any messages before selecting the $r+1$ messages. In the non-adaptive scenario, also called *target-subset-resilience* (as it is a generalization of *target-collision-resistance* [1]), the adversary must first select r messages, after which they are given the key K to select the last message.

Formally, the definitions by Reyzin and Reyzin can be given in terms of adversarial advantage. We define the advantage of an adversary $\mathcal{A}$ against the *r-subset-resilience* property of $\mathcal{O}_n$ as:

$$\textsc{Succ}^{r-\mathrm{SR}}_{\mathcal{O}_n}(\mathcal{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K}; (M_1, \ldots, M_{r+1}) \leftarrow \mathcal{A}(K) : C_K(M_1, \ldots, M_{r+1})\right]$$

where the probability is over the choice of K (uniform in the key space $\mathcal{K}$) and the internal coins of $\mathcal{A}$.

Similarly, we define the advantage of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ against the *r-target-subset-resilience* of $\mathcal{O}_n$ as:

$$\textsc{Succ}^{r-\text{TSR}}_{\mathcal{O}_n}(\mathcal{A}) = \Pr\Big[M_1, \ldots, M_r \xleftarrow{\$} \mathcal{A}_1; K \xleftarrow{\$} \mathcal{K}; M_{r+1} \leftarrow \mathcal{A}_2(K, M_1, \ldots M_r) : C_K(M_1, \ldots, M_{r+1})\Big]$$

Reyzin and Reyzin originally defined (target-)subset-resilience as negligibility of the advantage of probabilistic polynomial-time adversaries w.r.t. the variables (t, k), more precisely as $\textsc{Succ}^{r-(\text{T})\text{SR}}_{\mathcal{O}_n}(\mathcal{A}) < negl(t, k)$. However, we argue that such asymptotic definitions have little practical interest because for concrete schemes what matters is the security level for some fixed (t, k). Also, they did not explicitly define $negl(t, k)$, and for example an adversary could easily generate a subsets-cover if t and k go to infinity with $t = k$. Last, in practice r is not fixed and we are also interested in asymptotics in r, as in the case of SPHINCS.

Instead, we consider more precise notions of security. Given a set of resources ξ (e.g. runtime, number of queries), we define the insecurity of $\mathcal{O}_n$ for property P as:

$$\textsc{InSec}^{\text{P}}(\mathcal{O}_n; \xi) = \max_{|\mathcal{A}| \leq \xi} \{\textsc{Succ}^{\text{P}}_{\mathcal{O}_n}(\mathcal{A})\}$$

where $|\mathcal{A}| \leq \xi$ means that adversary $\mathcal{A}$ uses at most resources ξ. In a simplified model of computation ξ is typically the runtime (which is lower bounded by the number of queries to $\mathcal{O}_n$), but in a more realistic model of computation one must also include costs related to memory, communication, hardware, energy consumption, etc.

In our analysis, we consider only generic attacks against $\mathcal{O}_n$. In practice, if a particular family $\mathcal{O}_n$ has structural weaknesses that allow more efficient attacks, this family is considered broken and one can replace it with another family $\mathcal{O}'_n$.

Note. The HORST scheme [3] – for "HORS with trees" – simply adds a Merkle tree on top of HORS to compress the public key. This is irrelevant to attacks against subset-resilience, so unless otherwise specified the results in this paper hold for both HORS and HORST.

2.5 Previous results

Reyzin and Reyzin studied the non-adaptive scenario (target-subset-resilience) and considered a brute-force attack, i.e. given $M_1, \ldots, M_r$ iterate M_{r+1} over the message space until a match is found. In that case, given $r+1$ arbitrary messages and a uniformly distributed key K, the probability that $\mathsf{ORS}_K(M_{r+1})$ is covered by the first r messages is at most $(kr/t)^k$. This is the probability that k elements chosen uniformly at random in T are a subset of $\cup_{j=1}^{r} \mathsf{ORS}_K(M_j)$, which contains

at most kr elements of T. In other words, the security level against this generic attack is:

$$k(\log_2 t - \log_2 k - \log_2 r)$$

In this paper, we study the adaptive scenario and show that the security level decreases much faster when r increases.

3 Adaptive attacks against subset-resilience

In this section, we give a new lower bound on the subset-resilience security level, assuming only generic attacks against $\mathcal{O}_n$. We show that compared to target-subset-resilience, the (logarithmic) security level decreases by a factor proportional to $r+1$.

We establish the following theorem on adaptive attacks.

Theorem 1. *Assuming only generic attacks against $\mathcal{O}_n$, we have the following bound on the r-subset-resilience against adversaries performing at most q queries:*

$$\textsc{InSec}^{r-\mathrm{SR}}(\mathcal{O}_n; q) \leq \frac{(q+r+1)^{r+1}}{r!}\left(\frac{kr}{t}\right)^k$$

Proof. Let $\mathcal{A}$ be an adversary against the subset-resilience property of $\mathcal{O}_n$. Given a key K, we assume that $\mathcal{A}$ makes q distinct (offline) queries to ORS_K and outputs $r+1$ messages. We construct an adversary $\mathcal{A}'$ that runs $\mathcal{A}$ and additionally queries the $r+1$ messages output by $\mathcal{A}$, before outputting them. Adversary $\mathcal{A}'$ has the same success probability as $\mathcal{A}$ and makes at most $q' = q+r+1$ queries. We denote as $(M_i, \mathsf{ORS}_K(M_i))_{1\leq i\leq q'}$ the q' queries of $\mathcal{A}'$.

Under these assumptions, the advantage of $\mathcal{A}'$ is bounded by the probability that there exists a permutation σ of $\{1,\ldots,q'\}$ yielding the following r-subsets-cover:

$$C_K(M_{\sigma(1)},\ldots,M_{\sigma(r+1)})$$

Given a permutation σ, the probability that this condition holds is at most $(kr/t)^k$, as in the non-adaptive case. Besides, there are $q'\binom{q'-1}{r}$ distinct configurations: each configuration is given by a choice of $\sigma(r+1)$ among q' indices, followed by the choice of a subset $\{\sigma(1),\ldots,\sigma(r)\}$ of r indices among the remaining $q'-1$. By a union bound over all configurations, we obtain the following bound on the adversarial advantage.

$$\textsc{Succ}^{r-\mathrm{SR}}_{\mathcal{O}_n}(\mathcal{A}) = \textsc{Succ}^{r-\mathrm{SR}}_{\mathcal{O}_n}(\mathcal{A}') \leq q'\binom{q'-1}{r}\left(\frac{kr}{t}\right)^k \leq \frac{q'^{r+1}}{r!}\left(\frac{kr}{t}\right)^k$$

The result follows. □

For practical parameters ($q \gg r^2$), we have the following approximation:

$$\frac{(q+r+1)^{r+1}}{r!}\left(\frac{kr}{t}\right)^k \approx \frac{q^{r+1}}{r!}\left(\frac{kr}{t}\right)^k$$

The associated security level corresponds to the number of queries necessary for the advantage to be close to 1, that is:

$$\frac{k}{r+1}(\log_2 t - \log_2 k - \log_2 r) + \frac{\log_2 r!}{r+1}$$

As we can see, contrary to non-adaptive attacks, security degrades quickly with r due to the denominator, in a birthday-like manner. The case $r = 1$ is similar to collision resistance.

4 Practical attacks against HORS

In the previous section, we gave conservative bounds by considering only whether a subsets-cover exist, but we did not provide practical algorithms to find them. In particular, even though a r-subsets-cover exists, a naive algorithm that iterates over all configurations σ would take a time proportional to $q\binom{q-1}{r}$, introducing no improvement over a non-adaptive attack. Hence, we look for faster algorithms to find subsets-covers.

4.1 Reduction to *set cover*

Given q messages and their hashes, finding a subsets-cover reduces to the *set cover* problem. Indeed, for a given $M_{\sigma(r+1)}$, we try to cover the set $X = \mathsf{ORS}_K(M_{\sigma(r+1)})$ with a cover C of r elements from the family $\mathcal{Y} = \{Y_i\}_{i \neq \sigma(r+1)}$, where $Y_i = \mathsf{ORS}_K(M_i) \cap X$. Set cover is an NP-complete decision problem; the associated optimization problem (finding the smallest r) is NP-hard [7,5]. However, approximations can be computed with polynomial complexity.

Greedy algorithm. Given a set X to be covered by a family $\mathcal{Y}$, a simple heuristic is the following greedy algorithm:

1. Initialize an empty cover $C \leftarrow \emptyset$.
2. Select $Y_{\max} \in \mathcal{Y}$ such that $|Y_{\max}|$ is maximal.
3. Add $Y_{\max}$ to the cover C.
4. Update $X \leftarrow X \backslash Y_{\max}$ and $\mathcal{Y} \leftarrow \{Y_i \backslash Y_{\max} | Y_i \in \mathcal{Y}\}$, where $A \backslash B$ denotes the set difference.
5. Repeat steps 2-4 until C contains r subsets or X is empty.
6. If $X = \emptyset$, the algorithm succeeds and outputs C.

The worst-case complexity of this greedy algorithm is $O(qr)$, much better than the naive $O(q^r/r!)$. The question is now: depending on the parameters k, t, r, q, what is the success probability of this algorithm?

Previous work. The general *set cover* problem has long been studied [7,5] and Feige has shown that the greedy algorithm achieves the best approximation ratio for probabilistic polynomial-time algorithms [5]. However, these results are for the worst-case scenario, where the family $\mathcal{Y}$ can be chosen in an adversarial manner w.r.t. the algorithm. In our case, $\mathcal{Y}$ is chosen according to the HORS construction, which yields a very specific probability distribution and we are interested in the *average* approximation ratio.

4.2 Targeting weak messages

Instead of running the greedy algorithm on an arbitrary message, the adversary can select a better target. Indeed, due to the HORS construction, some messages have an image by ORS_K that contain only $\kappa < k$ elements. We call them *weak messages*. For example, following is the SHA-256 of "88681", grouped by blocks of 8 bits, with some repeated bytes underlined.

`98 32 3d bf 2a 64 75 32 0f f6 64 7e 98 75 64 98 f6 f5 54 02 ...`

Hence, we propose the following optimized algorithm: first scan through the q messages to find M such that $\mathsf{ORS}_K(M)$ has the least number of elements, then try to cover $\mathsf{ORS}_K(M)$ greedily. This new algorithm still has a complexity of $O(qr)$ and its success probability can be slightly better than the naive greedy algorithm.

To summarize, we have three algorithms:

- the **greedy algorithm**, that takes as input a set X to be covered by a family $\mathcal{Y}$,
- the **naive greedy algorithm**, that takes as input messages $M_1, \ldots, M_q$ and applies the greedy algorithm with an arbitrary $X = \mathsf{ORS}_K(M_1)$ and $\mathcal{Y} = \{\mathsf{ORS}_K(M_j) | j \neq 1\}$,
- the **optimized greedy algorithm**, that takes as input q messages and applies the greedy algorithm with the smallest $\mathsf{ORS}_K(M_i)$ as X, and $\mathcal{Y} = \{\mathsf{ORS}_K(M_j) | j \neq i\}$.

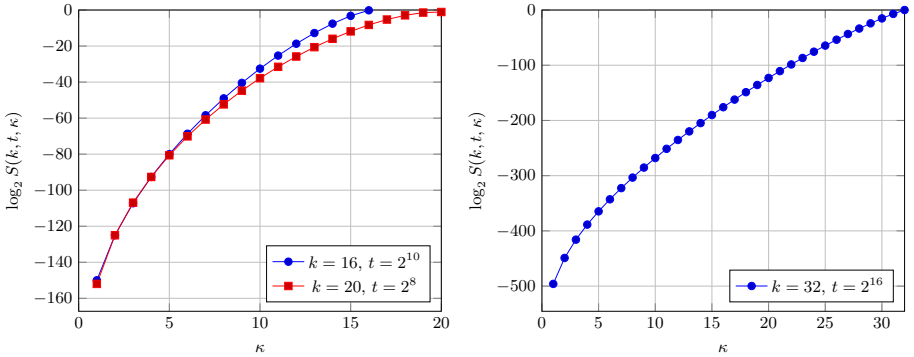
Probability distribution of weak messages. We now give a useful combinatorics result about the distribution of weak messages. For clarity, proofs of theorems and lemmas in this section are given in Appendix A.

Lemma 1. *Let $S(k, t, \kappa)$ be the probability that when we throw k balls independently and uniformly into t bins, exactly κ bins are non-empty. Then we have the following equality:*

$$S(k, t, \kappa) = \frac{\kappa!}{t^k} \binom{t}{\kappa} \left\{ {k \atop \kappa} \right\}$$

where $\left\{ {k \atop \kappa} \right\}$ is the notation for Stirling numbers of the second kind.

The probability $\mathsf{Pr}[|\mathsf{ORS}_K(M)| = \kappa]$ that a message is mapped to exactly κ elements is precisely $S(k, t, \kappa)$. Figure 1 shows the evolution of $\log_2 S(k, t, \kappa)$ for the choices of the parameters (k, t) that were proposed for HORS [10] and SPHINCS [3].



(a) Textbook HORS. (b) HORST as used in SPHINCS.

Fig. 1: Evolution of $S(k,t,\kappa)$ for various choices of (k,t). $S(k,t,\kappa)$ is the probability that when we throw k balls independently and uniformly into t bins, exactly κ bins are non-empty (Lemma 1).

4.3 Complexity analysis

We now give a complexity analysis of the greedy and optimized greedy algorithms, to estimate the security level that one can obtain against this generic attack. Given ORS parameters (k,t), a family $\mathcal{Y}$ of q elements, a number of iterations r and an initial subset X of size κ, we denote by $P^{k,t}_{\mathsf{Greedy}}(\kappa,r,q)$ the success probability of the greedy algorithm. Given q queries, we denote by $P^{k,t}_{\mathsf{GreedyNaive}}(r,q)$ the success probability of the naive greedy algorithm that takes an arbitrary message as the initial subset X. We denote by $P^{k,t}_{\mathsf{GreedyOptim}}(r,q)$ the success probability of the optimized greedy algorithm that selects the weakest of the q messages as the initial subset X.

Naive greedy algorithm We first establish a recurrence relation on the probability $P^{k,t}_{\mathsf{Greedy}}(\kappa,r,q)$.

Theorem 2. *Given a target subset of κ elements, the success probability of the greedy algorithm $P^{k,t}_{\mathsf{Greedy}}(\kappa,r,q)$ verifies the following recurrence relation.*

$$P^{k,t}_{\mathsf{Greedy}}(\kappa,r,q) = \begin{cases} 1 & \text{if } \kappa = 0 \\ 0 & \text{if } \kappa > 0 \wedge (r = 0 \vee q = 0) \\ \sum_{\ell=1}^{\kappa} \mathsf{Pr}[|Y_{\max}| = \ell | k,t,\kappa,q] P^{k,t}_{\mathsf{Greedy}}(\kappa-\ell, r-1, q-1) & \text{if } \kappa, r, q > 0 \end{cases}$$

We now aim at evaluating the probability $\mathsf{Pr}[|Y_{\max}| = \ell | k,t,\kappa,q]$ that the largest intersection $|\mathsf{ORS}_K(M) \cap X|$ contains ℓ elements. We start with the following lemma.

Lemma 2. *For $0 \leq \ell \leq \kappa$ and M a message uniformly distributed in $\mathcal{M}$, we denote by $P(k,t,\kappa,\ell)$ the probability that $|\mathsf{ORS}_K(M) \cap X| = \ell$ given that $|X| = \kappa$. This probability is equal to:*

$$P(k,t,\kappa,\ell) = \sum_{\lambda=\ell}^{k} B\left(\lambda, k, \frac{\kappa}{t}\right) S(\lambda,\kappa,\ell)$$

where S is defined in Lemma 1 and $B(\lambda,k,p)$ is the binomial distribution:

$$B(\lambda,k,p) = \binom{k}{\lambda} p^{\lambda}(1-p)^{k-\lambda}$$

We can now relate $\mathsf{Pr}[|Y_{\max}| = \ell | k,t,\kappa,q]$ to $P(k,t,\kappa,\ell)$.

Lemma 3. *We have the following equality:*

$$\mathsf{Pr}[|Y_{\max}| \geq \ell | k,t,\kappa,q] = 1 - \left(\sum_{\lambda=0}^{\ell-1} P(k,t,\kappa,\lambda)\right)^{q}$$

Corollary 1. *We can then compute $\mathsf{Pr}[|Y_{\max}| = \ell | k,t,\kappa,q]$ as:*

$$\mathsf{Pr}[|Y_{\max}| = \ell | k,t,\kappa,q] = \mathsf{Pr}[|Y_{\max}| \geq \ell | k,t,\kappa,q] - \mathsf{Pr}[|Y_{\max}| \geq \ell+1 | k,t,\kappa,q]$$

Although finding a more explicit analytic formula seems challenging, Theorem 2 and Lemmas 2 and 3 allow to compute $P^{k,t}_{\mathsf{Greedy}}(\kappa,r,q)$ by dynamic programming for concrete values of the parameters.

We can now estimate the success probability of the naive algorithm.

Theorem 3. *The success probability of the naive greedy algorithm is equal to:*

$$P^{k,t}_{\mathsf{GreedyNaive}}(r,q) = \sum_{\kappa=1}^{k} S(k,t,\kappa) P^{k,t}_{\mathsf{Greedy}}(\kappa,r,q-1)$$

Optimized greedy algorithm We now relate the success probability of the optimized greedy algorithm to the success probability of the greedy algorithm.

Theorem 4. *The success probability of the optimized greedy algorithm is equal to:*

$$P^{k,t}_{\mathsf{GreedyOptim}}(r,q) = \sum_{\kappa=1}^{k} \mathsf{Pr}[|Y_{\min}| = \kappa | k,t,q] P^{k,t}_{\mathsf{Greedy}}(\kappa,r,q-1)$$

where $|Y_{\min}|$ is the size of the weakest message:

$$|Y_{\min}| = \min_{Y \in \mathcal{Y}} |Y|$$

We now relate $\mathsf{Pr}[|Y_{\min}| = \kappa | k,t,q]$ to $S(k,t,\kappa)$.

Lemma 4. *We have the following equality:*

$$Pr[|Y_{\min}| \leq \kappa | k, t, q] = 1 - \left(\sum_{\lambda=\kappa+1}^{k} S(k, t, \lambda) \right)^{q}$$

Corollary 2. *We can then compute* $Pr[|Y_{\min}| = \kappa | k, t, q]$ *as:*

$$Pr[|Y_{\min}| = \kappa | k, t, q] = Pr[|Y_{\min}| \leq \kappa | k, t, q] - Pr[|Y_{\min}| \leq \kappa - 1 | k, t, q]$$

Theorem 4 and Lemma 4 allow to compute $P^{k,t}_{\mathsf{GreedyOptim}}(r, q)$.

4.4 Practical security level

In the previous section, we studied how to compute the success probability of the naive and optimized greedy algorithms. We now reduce the complexity of these attacks to these success probabilities.

Given q queries and a target r, the naive greedy algorithm has a complexity proportional to $1 + r(q-1)$ (one query for the covered message and r iterations over the $q-1$ other queries) and a success probability of $P^{k,t}_{\mathsf{GreedyNaive}}(r, q)$, so we expect to repeat it $1/P^{k,t}_{\mathsf{GreedyNaive}}(r, q)$ times on average to obtain a r-subsets-cover. The optimal number of queries q is the one that minimizes the attack complexity:

$$\frac{1 + r(q-1)}{P^{k,t}_{\mathsf{GreedyNaive}}(r, q)}$$

Likewise, given q and r, the optimized greedy algorithm has a complexity proportional to $q + r(q-1)$ (q queries to find the weakest message and r iterations over the remaining $q-1$ queries to cover it) and a success probability of $P^{k,t}_{\mathsf{GreedyOptim}}(r, q)$. The optimal number of queries q is the one that minimizes the attack complexity:

$$\frac{q + r(q-1)}{P^{k,t}_{\mathsf{GreedyOptim}}(r, q)}$$

We estimated these attack complexities by taking the minimum over $q \in \{2^n | n \in \mathbb{N}\}$. Figure 2 shows a comparison of the lower bounds on adaptive and non-adaptive attacks, and the complexity of the naive and optimized greedy algorithms, for some of the parameters originally proposed for HORS ($k = 16, t = 1024$). We can see that the optimized greedy algorithm is very close to the lower bound on adaptive attacks, despite its simplicity. Also, the difference between the naive and optimized algorithms becomes small as r grows.

4.5 Forgeries on textbook HORS

In [10], Reyzin and Reyzin proposed several combinations of (k, t) for the HORS signature scheme: ($k = 20, t = 256$) and ($k = 16, t = 1024$). Considering only non-adaptive attacks, they claimed a security level of 53 bits in the first case

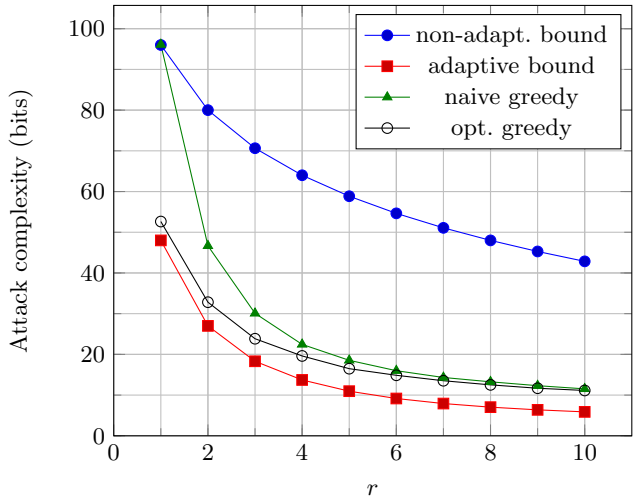


Fig. 2: Security level for $k = 16, t = 1024$. Lower bounds for adaptive and non-adaptive attacks and complexity of naive and optimized greedy algorithms.

with $r = 2$ signatures and 64 bits in the second case with $r = 4$ signatures. We challenge these claims in the case of adaptive attacks – which are totally possible in this textbook version of HORS – and give forgeries for these sets of parameters. We computed these forgeries with the optimized greedy algorithm.

Reyzin and Reyzin originally proposed to instantiate $\mathcal{O}_n$ with a hash function such as SHA-1, but given that this function is not collision-resistant [12] we used SHA-256 instead for our proof-of-concept attack, trimming the output to 160 bits to obtain compatible parameters. In other words, we used $\mathsf{ORS}_K(\cdot) = \mathsf{split}(\mathsf{trim}(\text{SHA-256}(\cdot), 160), k)$ where $\mathsf{trim}(\cdot, 160)$ returns the first 160 bits of its input, and split groups the output into k blocks of $\log_2 t$ bits. We limited ourselves to $q = 2^{22}$ queries, with each query M_i being the decimal representation of i as an ASCII string.

First case $k = 20, t = 256$. With only $q = 2^{21}$ queries, we found a 2-subsets-cover with the optimized greedy algorithm (Table 1). This is much smaller than the advertised security level of 53 bits against non-adaptive attacks [10], and a little larger than our adaptive lower bound of 18.1 bits estimated in Section 3. We note that the target subset $\mathsf{ORS}_K(M_{88681})$ contains only $\kappa = 13$ distinct elements.

Second case $k = 16, t = 1024$. With only $q = 2^{22}$ queries, we found a 3-subsets-cover with the optimized greedy algorithm (Table 2). This is much smaller than the advertised security level of 70 bits against non-adaptive attacks [10], and a little larger than our adaptive lower bound of 18.3 bits estimated in Section 3. The target subset $\mathsf{ORS}_K(M_{88817})$ contains only $\kappa = 12$ distinct elements.

i	160 first bits of SHA-256(i) (by groups of 8 bits)
88681	98 32 3d bf 2a 64 75 32 0f f6 64 7e 98 75 64 98 f6 f5 54 02
1468639	54 7e 64 39 f6 61 1f 4d 02 32 3d 23 68 62 9d 3e 38 bc 75 5c
80937	4b 98 e7 f5 05 9b ee 8d f4 0f 89 bb 13 7e bf 08 89 fc be 2a

Table 1: Example of 2-subsets-cover for $k = 20, t = 256$. The first line is the covered message. Underlined blocks are repeated in the same line. Colored blocks are common to several lines.

i	160 first bits of SHA-256(i) (by groups of 10 bits)
88817	3f0 2b4 193 087 38c 1e0 2b4 193 1b6 116 1c1 087 046 33a 243 243
2530852	1a9 2c6 1a2 0fe 15f 279 33a 026 2b4 3e6 2a9 116 09f 087 111 3f0
2351182	00e 02a 39c 243 292 378 152 22c 201 0ab 06d 1b6 1ff 204 1e0 1c1
216522	277 38c 06d 39c 193 124 376 2a9 08a 046 351 2df 072 219 34e 1fe

Table 2: Example of 3-subsets-cover for $k = 16, t = 1024$.

We also found a 4-subsets-cover with 2^{17} queries, again much smaller than the advertised 64 bits of security (our adaptive lower bound is 13.7 bits in this case).

5 Application to SPHINCS

We now study attacks against subset-resilience in the context of the SPHINCS construction. At first glance, SPHINCS is not susceptible to adaptive attacks, because the signer selects for each message M_i a key K_i in a manner non-predictable by an adversary. More precisely, $K_i = \mathsf{hash}(\mathsf{salt}, M_i)$ where salt is a secret value known only by the signer. That way, an adversary cannot compute subsets-covers in advance; they have to first query the signatures of some messages and then try to find a key-message pair (K', M') such that $\mathsf{ORS}_{K'}(M')$ is covered by the previous signatures.

However, the SPHINCS construction uses $N = 2^h$ instances of HORST in parallel (typically 2^{60}). For each query, the signer also selects a HORST instance i in a deterministic but non-predictable manner. The problem is then different than target-subset-resilience because an adversary can attack multiple instances in parallel. In particular, although looking for weak messages does not seem to help to break a single HORST instance in a non-adaptive attack, this can provide a speed-up for the multi-target scenario.

The adversary can also focus on a HORST instance of their choice, a strategy outlined in the original analysis of SPHINCS [3].

5.1 Search strategies

More formally, we let $\overline{\mathcal{M}} = \mathcal{K} \times \mathcal{M}$ (the extended message space that also includes the ORS key). Given q known signatures, the goal of an adversary is to find a pair $(i, (K, M)) \in \{1, \ldots, N\} \times \overline{\mathcal{M}}$ such that $\mathsf{ORS}_K(M)$ is covered by the known subsets of the i-th HORST instance. A naive strategy is to brute-force over the full space $\{1, \ldots, N\} \times \overline{\mathcal{M}}$.

However, each dimension of the search space $\{1, \ldots, N\} \times \overline{\mathcal{M}}$ has weaknesses. On the one hand, we have already seen that $\overline{\mathcal{M}}$ contains weak messages. On the other hand, $\{1, \ldots, N\}$ has weak HORST instances: the ones that produced many signatures. A more clever strategy, outlined in the SPHINCS paper [3] is to focus on weak HORST instances and try to find matching messages for these instances – via classical brute-force over the message space or Grover's quantum search algorithm. A symmetric strategy is to look for weak messages – via brute-force or Grover's algorithm – and try to find matching HORST instances. We now evaluate the complexity of each strategy.

5.2 Naive search

We let $\rho = q/N$ be the ratio of the number of queries by the number of HORST instances. Given an arbitrary pair $(i, \overline{M}) \in \{1, \ldots, N\} \times \overline{\mathcal{M}}$, the probability $P_\rho(r)$ that r messages were signed with the i-th HORST instance can be approximated by a Poisson distribution, as shown in [3].

$$P_\rho(r) \approx e^{-\rho} \rho^r / r!$$

Then, the probability that $\mathsf{ORS}_K(M)$ is covered by this instance is at most $(kr/t)^k$ – a tight approximation if $kr \ll t$. Hence, the success probability for the pair $(i, \overline{M})$ is at most:

$$\sum_{r=0}^{\infty} P_\rho(r) \left(\frac{kr}{t}\right)^k = \left(\frac{k}{t}\right)^k \sum_{r=0}^{\infty} P_\rho(r) r^k$$

The last sum corresponds to the k-th moment of a Poisson distribution and is equal to a Touchard polynomial T_k [11].

$$\sum_{r=0}^{\infty} P_\rho(r) r^k = T_k(\rho) := \sum_{i=0}^{k} \left\{ {k \atop i} \right\} \rho^i$$

Hence, the classical complexity to obtain a forgery with this strategy is (in bits):

$$k(\log_2 t - \log_2 k) - \log_2 T_k(\rho)$$

With SPHINCS parameters ($k = 32, t = 2^{16}, \rho = 2^{-10}$), we obtain a classical complexity of 331 bits for this naive attack.

5.3 Quantum naive search

Given a quantum computer, one could use Grover's search algorithm, dividing by 2 the bit complexity. However, this is neglecting memory access costs! Indeed, given a function $F : \mathcal{X} \mapsto \{0,1\}$ such that $\mathsf{Pr}[x \xleftarrow{\$} \mathcal{X} : F(x) = 1] = 2^{-b}$, Grover's algorithm can find a preimage $x \in F^{-1}(1)$ with $\Theta(2^{b/2} f)$ quantum operations on $\Theta(f)$ qubits where f is the cost of evaluating F [6]. For the naive algorithm we consider $\mathcal{X} = \{1, \ldots, N\} \times \overline{\mathcal{M}}$ and $b = 331$ as shown in the previous section.

In our case, F must first compute $\mathsf{ORS}_K(M)$ and then compare this result to the q queries, outputting 1 if and only if a subsets-cover is found. While evaluating $\mathsf{ORS}_K(M)$ is somewhat cheap, the comparison circuit to a large database of q queries is expensive. In particular, one cannot use classical algorithms that perform conditional memory access such as binary search, because the quantum comparison circuit operates on a *quantum superposition* of values (and each value would need to access distinct parts of the memory).

In practice, a quantum straight-line comparison circuit would need $\Theta(q)$ operations on $\Theta(q)$ qubits [2]. Besides, communication costs must be taken into account in a realistic model of memory access, e.g. random access in a 2-dimensional table of size q takes time $\Theta(\sqrt{q})$, due to physical constraints – namely the speed of light. These remarks are similar to the analysis of quantum collision search by Bernstein [2].

With SPHINCS parameters $(k = 32, t = 2^{16}, h = 60)$ and $q = 2^{50}$, we obtain a quantum complexity of 215 bits, assuming an evaluation cost $f = q$, but we will see that more clever methods reduce memory costs and achieve better results.

5.4 Search over weak HORST instances

To speed up the search, one can focus on the weakest HORST instance i – found in time $\Theta(q)$ – and then look for messages $\overline{M} \in \overline{\mathcal{M}}$. For a given r, the probability that there exists an instance i that signed r messages can be approximated as (assuming $\rho \ll 1$):

$$1 - (1 - P_\rho(r))^{2^h} \approx 1 - \exp(-2^h P_\rho(r)) \lesssim \min\{1,\ 2^h P_\rho(r)\}$$

Then, the success probability for each message $\overline{M}$ is again close to $(kr/t)^k$. It follows that the classical complexity of this attack is (in bits):

$$\min_r \left[k(\log_2 t - \log_2 k - \log_2 r) + \max\left\{0,\ -h - \log_2 P_\rho(r)\right\}\right]$$

One can also use Grover's search algorithm over $\overline{\mathcal{M}}$. Contrary to the naive algorithm, memory access is limited to the selected HORST instance, so we neglect memory costs. The associated quantum complexity is:

$$\min_r \left[\frac{k}{2}(\log_2 t - \log_2 k - \log_2 r) + \max\left\{0,\ -h - \log_2 P_\rho(r)\right\}\right]$$

With SPHINCS parameters $(k = 32, t = 2^{16}, h = 60)$ and $q = 2^{50}$, we find a classical complexity of 277 bits and a quantum complexity of 138 bits, both for $r = 5$. This is in accordance with the SPHINCS paper [3].

Worldwide attack. An interesting question is whether one can benefit from this strategy to break *any* SPHINCS instance in the world. Indeed, if there are u SPHINCS instances, each containing 2^h HORST instances, and that each SPHINCS instance is used for at most q queries, then the probability that at least one HORST instance in the world was used for r messages is:

$$1 - (1 - P_\rho(r))^{2^h u}$$

The previous analysis can be adapted by replacing 2^h by $2^h u$. For example, if $u = 2^{40}$, this worldwide attack has a classical complexity of 256 bits and a quantum complexity of 128 bits (both for $r = 8$).

5.5 Search over weak messages

With a classical computer, an adversary can find a weak message of size κ in time $\Theta(1/S(k,t,\kappa))$. With a quantum computer, this can be reduced to $\Theta(\sqrt{1/S(k,t,\kappa)})$ with Grover's search algorithm.

Once a weak message $\overline{M}$ of subset size κ is found, it is covered by each HORST instance with probability $\approx (kr/t)^\kappa$, where r is the number of signatures already generated by this instance. The success probability of this message against a HORST instance is then:

$$\sum_{r=0}^{\infty} P_\rho(r) \left(\frac{kr}{t}\right)^\kappa = \left(\frac{k}{t}\right)^\kappa T_\kappa(\rho)$$

If the adversary only scans through the $\approx q = 2^h \rho$ HORST instances that have already been used to sign messages, the success probability for each instance increases to:

$$\frac{1}{1 - P_\rho(0)} \left(\frac{k}{t}\right)^\kappa T_\kappa(\rho) \approx \rho^{-1} \left(\frac{k}{t}\right)^\kappa T_\kappa(\rho)$$

Each weak message $\overline{M}$ is processed with complexity $\Theta(\max\{q, 1/S(k,t,\kappa)\})$ and is successful with approximate probability:

$$1 - \exp(-2^h (k/t)^\kappa T_\kappa(\rho)) \lesssim \min\{1,\ 2^h (k/t)^\kappa T_\kappa(\rho)\}$$

It follows that the classical complexity of this attack is (in bits):

$$\min_\kappa \left[\max\left\{0,\ \kappa(\log_2 t - \log_2 k) - \log_2 T_\kappa(\rho) - h\right\} + \max\left\{h + \log_2 \rho,\ -\log_2 S(k,t,\kappa)\right\}\right]$$

One can also use Grover's search algorithm over $\overline{\mathcal{M}}$, yielding a quantum complexity of:

$$\min_\kappa \left[\max\left\{0,\ \kappa(\log_2 t - \log_2 k) - \log_2 T_\kappa(\rho) - h\right\} + \max\left\{h + \log_2 \rho,\ \frac{-\log_2 S(k,t,\kappa)}{2}\right\}\right]$$

With SPHINCS parameters ($k = 32, t = 2^{16}, h = 60$) and $q = 2^{50}$, we find a classical complexity of 270 bits (for $\kappa = 26$) and a quantum complexity of 187 bits (for $\kappa = 5$). Compared to search over weak HORST instances, this attack is more efficient classically, but less efficient quantumly.

6 Fixing HORS: PRNG to obtain a random subset

As we have seen, the existence of weak messages for $\mathcal{O}_n$ allows to perform more efficient attacks. We propose to improve the HORS construction to remove weak messages and mitigate these attacks. Instead of splitting the output of a hash function into k possibly non-unique indices, we propose to use a PRNG and collect the first k distinct values.

We have also seen that SPHINCS has weak HORST instances, that can be targeted by an attacker. Therefore, we then extend our PORS construction to also select the hyper-tree leaf in SPHINCS, and show that this improvement increases the security level of SPHINCS by a significant margin.

6.1 PORS construction

Given a PRF $\mathcal{H}_n = \{H_K : \mathcal{M} \to \{0,1\}^n | K \in \mathcal{K}\}$ that maps messages to n-bit strings, and a PRNG $\mathcal{G}$ that expands a n-bit seed into an arbitrary long stream of bits, we consider the following *PRNG to obtain a random subset* construction. Given a key K and a message M, we let $\mathsf{seed} = H_K(M)$. We define the sequence $(x_i)_{i>0}$ computed by grouping the output of the PRNG by blocks of τ bits, i.e. $x_1||x_2||\ldots = \mathcal{G}(\mathsf{seed})$. We then define $\mathsf{ORS}_K(M)$ as the set containing the first k distinct values of $(x_i)_{i>0}$.

Soundness of the construction. Since $\mathcal{G}$ is a PRNG, the x_i are indistinguishable from independent random elements uniformly chosen in $\{0,1\}^\tau$. Collecting k distinct elements among t is a variant of the coupon's collector problem, and succeeds in an average of $t(H_t - H_{t-k}) \approx k$ steps, where $H_n \approx \log n$ is the n-th harmonic number. Besides, the failure probability of coupon collection after x steps decreases exponentially in $O(\varepsilon^x)$ with $\varepsilon = (k-1)/t$, so a small number of steps are enough with very high probability.

Comparison to other algorithms. Other algorithms [4,10] have been proposed to generate k distinct elements among T, but they are not as simple and practical. First, they take as input an integer uniformly distributed between 0 (inclusive) and $\binom{t}{k}$ (exclusive), which is non-trivial because $\binom{t}{k}$ is in general not a power of 2. Second, they require arithmetic operations on large integers up to $\binom{t}{k}$, i.e. 395-bit integers for SPHINCS parameters ($k = 32, t = 2^{16}$). Consequently, Reyzin and Reyzin's algorithm has a complexity of $O(k^2 \log t \log k)$ [10]. In contrast, our PORS construction has an average complexity of $O(k)$ for $k \ll t$. Indeed, we need to generate approximately k values and we can test if each value is new in $O(1)$ (e.g. with a hash table).

Besides, PORS offers a large flexibility on the choice of k, whereas HORS constrains it to $k \leq n/\log_2 t$. PORS can also be used to generate auxiliary data, as we will see in the case of SPHINCS.

6.2 Security of PORS

A non-adaptive adversary performing a brute-force attack against PORS has at most the following success probability for each message.

$$\frac{(kr)!}{(kr-k)!}\frac{(t-k)!}{t!} = \frac{kr}{t}\cdot\frac{kr-1}{t-1}\cdots\frac{kr-k+1}{t-k+1}$$

This corresponds to the probability that k distinct values uniformly distributed in a set of size t all fall into a subset of size kr. This is lower than the success probability bound against HORS, and the associated security level increases to:

$$\sum_{j=0}^{k-1}\log_2(t-j) - \log_2(kr-j) \geq k(\log_2 t - \log_2(kr))$$

The difference in security is especially large when r is small. For example, with SPHINCS parameters ($k = 32, t = 2^{16}$) and $r = 1$:

$$-\log_2\left[\left(\frac{kr}{t}\right)^k\right] = 352 \quad -\log_2\left[\frac{(kr)!}{(kr-k)!}\frac{(t-k)!}{t!}\right] \approx 394$$

Application to SPHINCS. If we apply the PORS construction to SPHINCS, the best attack of Section 5.4 (search over weak HORST instances) has an increased complexity, namely:

$$\min_r\left[\alpha\left(\sum_{j=0}^{k-1}\log_2(t-j) - \log_2(kr-j)\right) + \max\left\{0,\ -h - \log_2 P_\rho(r)\right\}\right]$$

where $\alpha = 1$ in the classical case and $1/2$ in the quantum case. We obtain 282 bits of classical complexity and 141 bits of quantum complexity (both for $r = 5$).

The naive search attack also has an increased complexity by a few bits, and is still non-competitive. The attack against weak messages (Section 5.5) does not apply to PORS, because there are no weak messages.

6.3 SPHINCS leaf selection

As we saw in Section 5.4, SPHINCS suffers from attacks against weak HORST instances. Indeed, even though a honest signer deterministically selects a HORST leaf as a function of the message, a potential forger has full control over this choice because the leaf index is part of the signature. To reduce the attack surface, we propose to extend PORS to also generate this index.

More precisely, to generate a h-bit index i – for a hyper-tree of size 2^h – one can use the first h bits of PORS's PRNG. Namely, we compute $i||x_1||x_2||\ldots = \mathcal{G}(\mathsf{seed})$ where seed is obtained from the message as before. A schematic comparison of textbook SPHINCS and our new construction is shown on Figure 3.

We note that our construction also improves flexibility, because h and k are not constrained by the output size of a hash function.

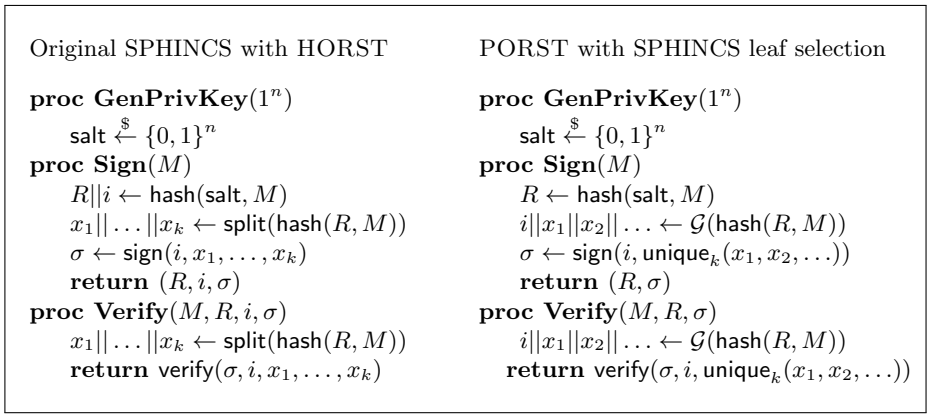


Fig. 3: Simplified comparison of HORST and PORST in SPHINCS.

Increased security level. With this new construction, attacks targeting a single weak PORST instance have a much higher complexity. The success probability for each message is divided by 2^h, so the classical complexity increases by h bits, and the quantum complexity by $h/2$ bits (for Grover-like attacks). Applied to SPHINCS, the classical complexity is 342 bits and the quantum complexity is 171 bits.

Naive attacks become more efficient in the classical case, with 337 bits of complexity. Intuitively, targeting a single PORST instance is not worth it because the probability to hit it is too low. However, quantum naive search is not competitive, due to memory access and comparison costs, so targeting a single weak PORST instance is still the best strategy.

Overall, replacing HORST by PORST with leaf selection in SPHINCS yields a gain of 67 bits of classical security (from 270 to 337) and 33 bits of post-quantum security (from 138 to 171). As a side effect, removing the leaf index from the signature also saves 8 bytes.

6.4 Revised parameters for SPHINCS

Thanks to the better security margin offered by PORST, we propose to decrease SPHINCS parameters to reduce signature size. For an equivalent number of queries $q = 2^{50}$ and similar signature times, we propose to reduce the total height to $h = 50$, divided into $d = 10$ layers of Merkle trees (each of height 5 as in SPHINCS). Signatures for SPHINCS-PORST have a size of 36384 bytes instead of 41000 bytes for textbook SPHINCS, due to the removal of 2 layers of Merkle trees (and of the leaf index).

The corresponding security level is 267 bits classical (naive search) and 136 bits post-quantum (search on weak HORST instance), similar to the original SPHINCS. For a slight improvement, one can also choose to reduce k to 29 instead of 32 to save 1152 more bytes, with 128 bits of post-quantum security.

The total height h can be reduced if the total number of signatures q is smaller, and the number of layers d can be decreased if more computation time is allotted to each signature.

7 Conclusion

In this paper, we have shown that adaptive attacks against HORS are much more dramatic than non-adaptive attacks. We showed how to break textbook HORS with a greedy algorithm and we gave example of forgeries, found with at most 2^{22} queries. We then studied the SPHINCS construction, reviewed multi-target attacks, and presented a new classical attack of complexity of 2^{270}, against 2^{277} for previously known attacks.

We then proposed to improve SPHINCS security by two means: using a PRNG to ensure uniqueness of generated indices, and to select a leaf. Although HORS shines by the simplicity of its subset generation – a single call to a hash function – it is the security foundation of a much more complex scheme such as SPHINCS, so it is worth taking a few more CPU cycles with a PRNG to select a strong subset. Besides, we don't see any reason why leaf selection is adversarially-controlled in textbook SPHINCS, and removing this degree of freedom makes known attacks much more complex.

In the end, we can trade the increased security margin for lower signature size and faster signature generation, pushing stateless hash-based signature schemes one step further towards practicality. An open question is to which extent our adaptive attacks could affect non-repudiability.

References

1. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making UOWHFs practical. In: CRYPTO (1997)
2. Bernstein, D.J.: Cost analysis of hash collisions: Will quantum computers make sharcs obsolete? SHARCS09 Special-purpose Hardware for Attacking Cryptographic Systems p. 105 (2009)
3. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O'Hearn, Z.: SPHINCS: practical stateless hash-based signatures. In: EUROCRYPT (2015)
4. Bos, J.N., Chaum, D.: Provably unforgeable signatures. In: CRYPTO (1992)
5. Feige, U.: A threshold of ln n for approximating set cover. J. ACM 45(4) (1998)
6. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: ACM Symposium on the Theory of Computing (1996)
7. Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. Syst. Sci. 9(3) (1974)
8. Lamport, L.: Constructing digital signatures from a one-way function. Tech. rep., Technical Report CSL-98, SRI International Palo Alto (1979)
9. Pieprzyk, J., Wang, H., Xing, C.: Multiple-time signature schemes against adaptive chosen message attacks. In: SAC (2003)
10. Reyzin, L., Reyzin, N.: Better than BiBa: Short one-time signatures with fast signing and verifying. In: ACISP (2002)

11. Riordan, J.: Moment recurrence relations for binomial, poisson and hypergeometric frequency distributions. The Annals of Mathematical Statistics 8(2) (1937)
12. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: CRYPTO (2017)

A Proofs of Lemmas and Theorems

Proof (Lemma 1). There are t^k equiprobable combinations of k balls into t bins. Out of those, we count the number of combinations for which exactly κ bins are non-empty:

- there are $\binom{t}{\kappa}$ ways to choose κ bins out of t;
- there are $\left\{ {k \atop \kappa} \right\}$ partitions of k labelled balls into κ classes (this is precisely the definition of Stirling numbers of the second kind);
- there are $\kappa!$ ways to associate the κ classes of balls to the κ bins.

This yields the result. □

Proof (Theorem 2). First, the greedy algorithm always succeeds if there is nothing to cover ($\kappa = 0$), and always fails if there is something to cover ($\kappa > 0$) but there is no iteration or message left ($r = 0$ or $q = 0$).

Second, if the first iteration finds a maximal subset of size ℓ, the success probability of the following iterations is equal to the success probability of the greedy algorithm on a subset of size $\kappa - \ell$ with one less iteration and one less available message. □

Proof (Lemma 2). We recall that $\mathsf{ORS}_K(\cdot)$ simulates throwing k balls uniformly and independently into t bins. With that in mind, $B(\lambda, k, \kappa/t)$ is the probability that λ balls out of k fall into the κ elements of X. Then, $S(\lambda, \kappa, \ell)$ is the probability that these λ balls cover exactly ℓ bins out of κ.

The sum starts at $\lambda = \ell$ because $S(\lambda, \kappa, \ell) = 0$ when $\lambda < \ell$ (it is impossible to cover ℓ bins with less than ℓ balls). □

Proof (Lemma 3). By independence of the Y_j:

$$\begin{aligned}
\mathsf{Pr}[|Y_{\max}| \geq \ell | k, t, \kappa, q] &= \mathsf{Pr}[\exists j \in \{1, \ldots, q\} \ |Y_j| \geq \ell | k, t, \kappa] \\
&= 1 - \mathsf{Pr}[\forall j \in \{1, \ldots, q\} \ |Y_j| < \ell | k, t, \kappa] \\
&= 1 - \prod_{j=1}^{q} \mathsf{Pr}[|Y_j| < \ell | k, t, \kappa] \\
&= 1 - \mathsf{Pr}[|Y_1| < \ell | k, t, \kappa]^q
\end{aligned}$$

where

$$\mathsf{Pr}[|Y_1| < \ell | k, t, \kappa] = \sum_{\lambda=0}^{\ell-1} P(k, t, \kappa, \lambda)$$

The result follows. □

Proof (Theorem 3). The naive algorithm selects an arbitrary message, that has a probability $S(k,t,\kappa)$ of containing κ distinct elements. Given that, there is a probability $P^{k,t}_{\mathsf{Greedy}}(\kappa, r, q-1)$ that the algorithm succeeds. □

Proof (Theorem 4). For $1 \leq \kappa \leq k$, there is a probability $\mathsf{Pr}[|Y_{\min}| = \kappa | k, t, q]$ that the weakest message found by the algorithm contains κ distinct elements. Given that, the algorithm succeeds with probability $P^{k,t}_{\mathsf{Greedy}}(\kappa, r, q-1)$. □

Proof (Lemma 4). The proof is similar to Lemma 3. By independence of the Y_j:

$$\begin{aligned}\mathsf{Pr}[|Y_{\min}| \leq \kappa | k, t, q] &= \mathsf{Pr}[\exists j \in \{1, \ldots, q\} \ |Y_j| \leq \kappa | k, t] \\ &= 1 - \mathsf{Pr}[\forall j \in \{1, \ldots, q\} \ |Y_j| > \kappa | k, t] \\ &= 1 - \mathsf{Pr}[|Y_1| > \kappa | k, t]^q\end{aligned}$$

Besides,

$$\mathsf{Pr}[|Y_1| > \kappa | k, t] = \sum_{\lambda=\kappa+1}^{k} S(k,t,\lambda)$$

which yields the result. □