Dynamic Optimization of Sphincs+ Signature Algorithm Parameters Based on A Ucb-Like Algorithm

Mingqi Zhang

School of Cyberspace Security, Shandong University, Qingdao, Shandong, 266237, China

Abstract. This study addresses the issues of redundant security and suboptimal resource efficiency in fixed parameter configurations of the post-quantum cryptographic algorithm Stateless PHoton-based Isogeny Nihil Crux Signature+ (SPHINCS+). Observing that the key generation overhead in this signature scheme is significantly lower than that of signature creation and verification processes, this work proposes a dynamic parameter optimization method based on a UCB-like algorithm. By modeling core parameters, including the total height of the HyperTree, the number of secret keys in Forest of Random Subsets (FORS) trees, and the leaf count of FORS trees as adjustable variables, our approach implements intelligent parameter selection through a multi-armed bandit framework. The algorithm autonomously balances cryptographic strength, signing speed, and storage costs by learning user signing patterns while adapting to real-time security requirements and system resource availability, thereby eliminating computational waste and excessive security design inherent in static parameter configurations. Experimental results demonstrate that this method achieves a 6% to 9% reduction in signature generation resource consumption while maintaining scenario-specific security guarantees. This research provides new optimization perspectives for enhancing the practical engineering applicability of the SPHINCS+ hash-based signature scheme.

1 Introduction

The rapid advancement of quantum computing technology poses significant security challenges to traditional public-key cryptosystems, particularly digital signature schemes relying on integer factorization and discrete logarithm mathematical problems [1]. In NIST's Post-Quantum Cryptography Standardization Project, the SPHINCS+ scheme has emerged as a seminal research achievement in the field of stateless digital signatures, owing to its hash-based design philosophy and formally provable security properties [2, 3].

First proposed by Bernstein et al. in 2019, SPHINCS+ evolved from its predecessor SPHINCS, the first practical post-quantum signature scheme in hash-based cryptography [4]. Compared to the original design, SPHINCS+ incorporates hypertree structures and the FORS framework [3, 5]. Its security foundation relies exclusively on the collision resistance of hash

202200460179@mail.sdu.edu.cn

functions and the one-time signature (OTS) paradigm, circumventing the reliance on intricate mathematical assumptions common in many post-quantum cryptographic approaches [2, 6].

However, SPHINCS+ still exhibits larger signature sizes (approximately 8-41 KB) compared to certain lattice-based or code-based alternatives [5]. To address this limitation, this study proposes a novel approach that dynamically adjusts parameter values through UCB-like algorithm-driven analysis of user signing patterns, while maintaining security guarantees. The optimized lightweight SPHINCS+ framework aims to eliminate redundant cryptographic operations, thereby mitigating unnecessary signing and verification overhead caused by excessive security margins.

The SPHINCS+ signature scheme inherently supports multiple signing operations under fixed parameters, eliminating the necessity of regenerating cryptographic keys for each signature instance. However, when employing dynamic parameter optimization strategies inspired by Upper Confidence Bound (UCB) algorithms, transient variations in algorithmic parameters during initial iterations inevitably introduce marginal overhead in key generation. This trade-off fundamentally stems from the scheme's adaptive capacity to balance computational efficiency and security robustness. Crucially, empirical analyses demonstrate that the computational resources required for key generation exhibit an order-of-magnitude reduction compared to those consumed by signature generation and verification processes. The proposed methodology strategically constrains parameter evolution within bounded temporal windows, thereby converging to locally optimal configurations that eliminate redundant security margins while preserving formal security guarantees. Such dynamic adaptation effectively substitutes transient key-generation overhead for sustained efficiency gains by circumventing perpetual execution under unnecessarily conservative parameter sets. This operational paradigm aligns with the resource-aware optimization framework formalized in National Institute of Standards and Technology (NIST) PQC standardization guidelines, where transient initialization costs are justified by long-term operational efficiency in practical deployment scenarios [7].

2 Related Work

The parametric design of SPHINCS+ and its performance trade-offs have been extensively studied in post-quantum signature research. Bernstein et al. (2019) systematically defined core parameters (e.g., hypertree height h, layer count d, FORS tree number k, and leaf size t) in Section 4 of the original paper, establishing mathematical relationships between these parameters and security strength, signature size, and computational cost [3]. Hülsing and Kudinov (2021) further optimized parameter configurations in Section 2.3 of their NIST submission document, proposing predefined parameter sets for different security levels (NIST Level 1–5). For instance, they reduced the signature size for Level 1 from 16.8 KB to 8.0 KB at the cost of increased hash function calls [5]. Yehia, M. et al. targeted the SPHINCS+ stateless hash-based signature scheme and proposed a v-ORS generation mechanism. By introducing additional hash operations to bind messages to the underlying FORS signature instances, this approach enhances security while enabling the exploration of more optimal parameter sets. The mechanism significantly reduces FORS computational costs and hash function invocations required for signature generation [8].

For resource-constrained environments, Kampanakis et al. Evaluate the applicability of two post-quantum signature schemes proposed by the Internet Engineering Task Force (IETF) in secure boot scenarios, confirm that their signing processes impose no significant performance overhead on firmware images, and analyze the feasibility of signature architectures across different hierarchical hardware secure boot designs [9]. Stefan Kolbl et al. optimized SPHINCS+ parameter configurations for low-frequency signing scenarios, achieving signature size reduction and verification speed improvement while maintaining

SLH-DSA compatibility through parameter adjustments. Systematic parameter space exploration and security degradation analysis demonstrate that security baselines can be preserved when moderately relaxing conventional constraints through scientific parameter selection. The optimized parameters demonstrated practical efficiency advantages in OpenTitan firmware signing use cases [10].

In studies addressing the security-performance trade-off in security-related domains, T. Li et al. designed a differential privacy-based communication-efficient algorithm to address communication bottlenecks and data privacy issues in federated multi-armed bandits, analyzing the trade-offs between privacy protection, communication efficiency, and learning performance in master-slave, decentralized, and hybrid architectures [11]. Chengshuai Shi et al. proposed a Personalized Federated Multi-Armed Bandit (PF-MAB) framework, developing the PF-UCB algorithm that balances global information sharing with local personalized learning to resolve the equilibrium between collaborative exploration and personalized decision-making for multiple agents in federated environments [12].

3 Methodology

3.1 Model Principles and Optimization

3.1.1 The SPHINCS+ Framework

SPHINCS+, as a stateless, hash-based post-quantum signature scheme, consists of three components.

1) Hypertree: At the hypertree layer, the structure comprises a recursive aggregation of binary Merkle trees arranged in d levels, where each level contains $2^{(h/d)}$ subtrees for a total tree height h. Each subtree root serves as a leaf node in the parent tree, forming a nested authentication chain. Within individual subtrees, leaves are generated by applying the Winternitz One-Time Signature+ (WOTS+) algorithm to hashed message blocks [3].

2) FORS: Replaces HORST in SPHINCS, The FORS layer operates at the message encoding stage, decomposing a 256-bit message digest into k segments of $\log_2 t$-bit indices, where $k \cdot \log_2 t = 256$ [4]. For each segment, the scheme selects t secret keys from a precomputed pool of k*t keys, arranged in k parallel key groups. During signing, the selected keys form a $k \cdot \log_2 t$-bit authentication path, with each key hashed $\log_2 t$ times to generate verifiable commitments. This creates a combinatorial structure where the total signature component spans $k \cdot t \cdot n$-bits (n=hash output size).

3) WOTS+: The WOTS+ base layer implements a hash chain network with w-ary encoding. For a 256-bit hash function, WOTS+ constructs chains, the len formula is:

$$len = \left\lceil \frac{256}{log_2 w} \right\rceil + \left\lfloor log_2 \left(\left\lceil \frac{256}{log_2 w} \right\rceil \cdot (w-1) \right) \Big/ log_2 w \right\rfloor \quad (1)$$

where w (Winternitz parameter) typically takes values 4, 16, or 256. Each chain consists of w-1 sequential hash iterations applied to a 32-byte secret seed. The public key compiles the final hash outputs of all chains, while signatures encode the chain positions corresponding to message bits through base-w decomposition [6].

The framework eliminates state synchronization requirements (unlike XMSS) and resists quantum attacks by reducing security to hash function properties (collision resistance and second-preimage resistance) [3, 6]. Parameters (e.g., hash functions: SHA-256/SHAKE-128)

are tunable for target scenarios: smaller signatures (8 KB) for Internet of Things (IoT) devices or higher security (NIST Level 5, 41 KB) for critical infrastructure [5].

3.1.2 UCB-like Algorithm

The UCB algorithm family constitutes a class of online learning strategies derived from the multi-armed bandit framework. Its core mechanism dynamically optimizes the exploration-exploitation trade-off to minimize cumulative regret through iterative decision-making. The canonical UCB1 algorithm (Auer et al., 2002) operates via a deterministic policy:

$$a_t = arg\ \max_i(\mu_i(t) + \sqrt{\frac{2\ ln\ t}{n_i(t)}}) \tag{2}$$

Where a_t represents the action selected, $\mu_i(t)$ denotes empirical reward mean, $n_i(t)$ the action attempt count, and the total iterations [1]. UCB-like variants diverge primarily in the exploration term's formulation.

This study investigates the integration of UCB-like algorithms with the SPHINCS+ signature scheme, focusing on learning user signing patterns to dynamically optimize its core parameters (h, t, k). Within the UCB-like framework, distinct parameter combinations are modeled as individual "arms" (decision alternatives in multi-armed bandit terminology).

This paper employs network traffic datasets to simulate user signing patterns, selecting entries ‘Timestamp’, ‘Total Fwd Packets’, ‘Total Backward Packets’, ‘Total Length of Fwd Packets’, and ‘Total Length of Bwd Packets’ as primary observation targets. A temporal window is defined during the signing process, where only data packets within this interval can be exploited by adversaries to construct collisions; packets outside this window are considered invalid for such attacks [13].

A higher volume of packets within the window increases adversarial advantages for collision attempts, thereby necessitating stronger cryptographic parameters. Any parameter configuration allowing collision probabilities exceeding 2^{-128} is deemed insecure.

However, enhanced parameters escalate computational cycles—including key pair generation, signing, and verification operations—with signing/verification cycles significantly exceeding those of key generation. This trade-off motivates the design of a UCB-based reward mechanism to balance security and efficiency.

$$Reward = \alpha \cdot f_1(SR) + (1 - \alpha) \cdot f_2(CY) \tag{3}$$

Here, SR denotes security redundancy, i.e., the difference between 128 and the negative logarithm of the security level under given parameters, quantifying security margin. CY represents cycles, i.e., computational costs for key pair generation and signing/verification operations. Notably, key generation cycles are only counted when the UCB algorithm switches parameter configurations, as SPHINCS+ permits multiple signatures under fixed keys. The reward function components operate as: f_1 satisfies that when SecRedundancy is larger, the output Reward is larger (note: this should not be inversely correlated, as SecRedundancy represents additional security and is not inherently a drawback—its induced extra cycles are the drawback, which will be reflected in f_2); but when SecRedundancy is less than 0 (insufficient security), f_1 outputs a sufficiently small value to ensure this parameter combination is not selected. f_2 satisfies that when Cycles are larger, the output Reward is smaller. The weighting factor α governs security-efficiency prioritization: a higher α means greater emphasis on security, while a lower α prioritizes lightweight processes.

3.2 Evaluation

3.2.1 Security Evaluation

As demonstrated by Bernstein, D. et al., the security of the SPHINCS+ signature scheme is formally quantified by the expression below:

$$SL \approx (q_{hash} + 1) \sum_{\gamma} \left(1 - \left(1 - \frac{1}{t}\right)^{\gamma}\right)^{k} \binom{q}{\gamma} \left(1 - \frac{1}{2^h}\right)^{q-\gamma} \frac{1}{2^{h\gamma}} \tag{4}$$

where SL denotes security loss, q means the number of queries, and q_{hash} represents the maximum number of hash queries an adversary can execute in cryptographic analysis [3].

When $\frac{1}{2^h}$ attains a cryptographically negligible magnitude, (1-$\left(1 - \left(1 - \frac{1}{t}\right)^{\gamma}\right)^{k}$ can be simplified to $\left(\frac{\gamma}{t}\right)^{k}$. Thus, the security bound can be simplified as $(q_{hash} + 1) \cdot E\left[\left(\frac{\gamma}{t}\right)^{k}\right]$. Where γ follows a binomial distribution $\gamma \sim Binomial(q, \frac{1}{2^h})$. Applying Poisson moment estimation and omitting lower-order terms, it can be derived as below:

$$E\left[\left(\frac{\gamma}{t}\right)^{k}\right] \approx \left(\frac{q}{2^h t}\right)^{k} \tag{5}$$

Consequently, the security bound reduces to $(q_{hash} + 1) \cdot \left(\frac{q}{2^h t}\right)^{k}$. This work intentionally disregards adversarial capabilities and instead focuses on the inherent security properties of the algorithmic framework itself. To this end, this work will establish a security evaluation criterion requiring $\left(\frac{q}{2^h t}\right)^{k}$.

3.2.2 Cumulative Reward Evaluation

For the evaluation of cumulative regret, this paper will compare the cumulative rewards under three scenarios:

1) Optimal fixed parameters: Selecting the optimal parameter combination from the first round and maintaining it throughout.

2) UCB-like algorithm: Dynamically adjusting parameters using a UCB-inspired method.

3) NIST-compliant parameters: Adhering to the fixed parameter specifications mandated by the NIST standard.

The first scenario represents the theoretical upper bound (i.e., the most favorable outcome) but is probabilistically infeasible – the likelihood of consistently selecting optimal parameters from the outset is vanishingly small. Thus, its cumulative reward serves solely as a reference benchmark with no practical significance. For Scenarios 1 and 3, where parameter combinations remain static, the cumulative reward will be estimated by multiplying the per-round reward by the total number of rounds. For the UCB-like algorithm (Scenario 2), the cumulative reward will be calculated by subtracting the accumulated regret delta from the theoretical maximum reward.

4 Experiment and Results

4.1 Setup

For signature algorithm setup, in this work, the hash kernel of SPHINCS+ is uniformly implemented using ShangMi 3 (SM3). For technical details, refer to the open-source repository by Seallver [14].

Tables 1 and 2 present the hardware equipment used in the experiments mentioned in this article, including performance specifications under Windows and Linux operating systems. However, these are not strict requirements, meaning other versions of operating systems may not affect the feasibility of the experiments.

Table 1 Windows Hardware Setup

CPU	AMD Ryzen 7 8845H with Radeon 780M Graphics, 3.80 GHz
Installed RAM	32.0 GB
Operating System	Windows 11 24H2

Table 2 Linux Hardware Setup

CPU Model Name	AMD Ryzen 7 8845H with Radeon 780M Graphics, 3.80 GHz
CPU(s)	16
Thread(s) per core	2
Core(s) per socket	8

Table 3 lists the versions of Python and its libraries used in the experiments described in this paper. These are also not mandatory requirements, but Python and library versions no lower than those specified in the table are recommended. Additionally, the SPHINCS+ using SM3 code from Seallver might encounter compatibility issues with newer versions. If such problems occur, it is advised to downgrade the relevant libraries to the versions specified in the table.

Table 3 Software Setup

Python	3.10.123
Numpy	1.26.4
Pandas	2.0.3
Matplotlib	3.8.4

4.2 Results

This work formally defines:

$$f_1 = \begin{cases} log_2\big(log_2\big(g_1(h,t,k)\big) - 128\big), & SL \le 2^{128} \\ -10000, & SL > 2^{128} \end{cases} \tag{6}$$

$$f_2 = -log_2\big(g_2(h,t,k)\big) \tag{7}$$

where $g_1(h,t,k)$ computes the security loss metric based on parameters (h,t,k), and $g_2(h,t,k)$ outputs the corresponding computational cycles consumption derived from Seallver's cycle counting methodology. The experiment comprises 40,000 rounds, ensuring statistically sufficient exploration of all parameter combinations by the UCB-like algorithm.

4.2.1 Security Evaluation

This work conducted dynamic parameter optimization using the UCB algorithm with α = 0.1, 0.3, 0.5, and 0.7. The cumulative regret values, selected parameter combinations, and the frequency of each combination being chosen across all rounds under different α values are summarized in Table 4.

Table 4 Experimental Results of Different α

α	Cumulative Regret	Parameter Set	Chosen Times
0.1	27575.85	h=50, t=10, k=10	30853
0.3	43517.69	h=55, t=5, k=10	31164
0.5	118664.05	h=55, t=10, k=25	26511
0.7	146458.86	h=60, t=5, k=30	36289

The cumulative regret curves under different alpha values are shown in Fig. 1. During the initial 500 rounds of the total 40,000 rounds, the experiment undergoes a UCB-like algorithm's exploration phase, where the Cumulative Delta increases significantly. Since different alpha values require distinct parameters, higher parameter values consume more computational cycles, thereby resulting in a larger Cumulative Delta. In the remaining rounds, the experiment enters the exploitation phase, and the Cumulative Delta eventually converges to a linearly increasing trend.

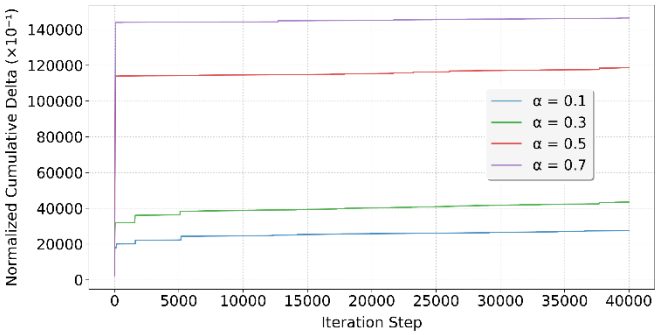

Fig. 1 Cumulative Delta Comparison Across α Values (Photo credit: Original)

4.2.2 Cumulative Reward Evaluation

For this phase, α is fixed at 0.1 to attenuate the weighting of supplementary security considerations. The selected configuration will undergo 40,000 rounds of dynamic parameter optimization.

Under the NIST security standards, the SPHINCS+ parameters (60,15,30) yield a per-round computational cycle consumption of approximately 2^{34}. This allows the estimated per-round reward (calculated as $0.1f_1(60,15,30) + 0.9f_2(60,15,30)$) to be quantified as -29.56.

Experimental results demonstrate that the parameter combination (50, 10, 10) was most frequently selected (30,853 times in 40,000 rounds), achieving a final cumulative regret of 27,575.85 (rounded from 27,575.847567615414).

For comparison, an oracle-driven baseline—where the optimal parameters (50, 10, 10) are fixed from the first round—yields a per-round cycle cost of 2^30.6 and a corresponding reward of -26.8 (via $0.1f_1(50,10,10) + 0.9f_2(50,10,10)$).

A comparative analysis reveals approximate cumulative rewards of -1.07×10^6 (optimal parameter set), -1.10×10^6 (UCB-based dynamic parameter optimization), and -1.18×10^6 (NIST-compliant parameter set).

This work visualizes the comparative results of three approaches using bar charts in Fig. 2. While the UCB-like algorithm sacrifices some reward during the initial exploration phase, it consumes fewer computational cycles compared to directly adopting the NIST standard recommended parameters, ultimately achieving higher reward values.

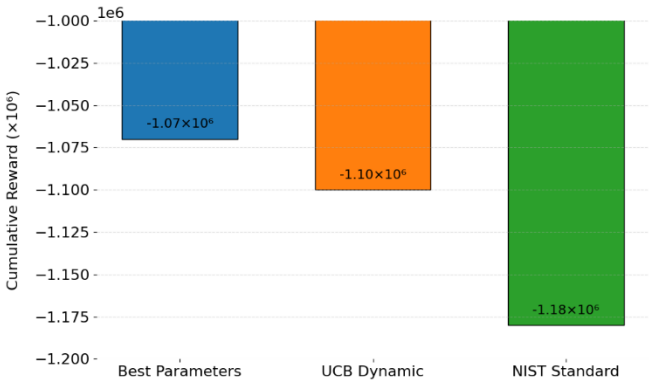

Fig. 2 Cumulative Rewards Across Parameter Selection Strategies (Photo credit: Original)

4.3 Analysis

For security evaluation, as demonstrated in Fig. 1, increasing α values bias the algorithm toward selecting parameter sets with higher computational demands, resulting in elevated per-round cycle consumption. This behaviour induces a proportional escalation in cumulative delta.

For cumulative reward analysis, Fig. 2 illustrates that UCB-like dynamic parameter optimization achieves a higher cumulative delta compared to the NIST-standard parameter set. This performance gap primarily stems from reduced cycle consumption enabled by smaller parameter selections. Notably, some scholars might argue that manually fixing a low-parameter configuration from the outset, rather than deploying UCB-like optimization, could yield comparable cumulative rewards.

However, such reasoning is flawed. The NIST-standard parameters are designed to satisfy post-quantum cryptographic security requirements. Arbitrarily adopting lower parameters risks compromising security, whereas UCB-driven optimization inherently enforces safety constraints: insecure parameter combinations are eliminated during the exploration phase due to their catastrophically low reward valuations.

5 Conclusion

This study addresses redundant security provisions and suboptimal resource efficiency inherent in fixed parameter configurations of the post-quantum cryptographic algorithm SPHINCS+. Capitalizing on the observation that key generation incurs lower overhead than signature creation and verification processes, this work implements dynamic parameter optimization via a UCB-like algorithm. By learning user signing patterns, this approach identifies contextually optimal parameter combinations for SPHINCS+ while preserving security guarantees. The derived configurations frequently demand lower specifications than

NIST-recommended post-quantum secure parameters, as excessive security levels often exceed practical requirements. However, preemptive adoption of reduced parameters without behavioral analysis risks cryptographic vulnerability. Experimental results demonstrate adaptive parameter adjustments achieving a 6%-9% reduction in signature generation resource consumption without compromising security. This work establishes a novel optimization framework to enhance SPHINCS+'s engineering viability.

The research acknowledges limitations in focusing solely on UCB-like algorithms within the multi-armed bandit framework, as time constraints precluded exploration beyond standard UCB variants. Serving as a preliminary investigation, it primarily lays methodological groundwork for subsequent studies. Future work should expand to comparative analyses of diverse MAB-based optimization strategies (e.g., MOSS, OCUCB) for SPHINCS+ parameter selection, systematically evaluating algorithmic performance across heterogeneous operational environments.

References

1. Shor, P.W.: 'Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer', SIAM Journal on Computing, 1997, 26, (5), pp. 1484–1509
2. National Institute of Standards and Technology (NIST): 'Post-quantum cryptography standardization', 2016
3. Bernstein, D., Hülsing, A., Kölbl, S., et al.: 'The SPHINCS+ signature framework'. Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS '19), New York, USA, 2019, pp. 2123–2146
4. Bernstein, D., Hopwood, D., Hülsing, A., et al.: 'SPHINCS: Practical stateless hash-based signatures'. Proc. Advances in Cryptology – EUROCRYPT 2015, Berlin, Germany, 2015, pp. 368–397
5. Hülsing, A., Kudinov, M.: 'SPHINCS+ submissions to the NIST PQC standardization project', NIST Post-Quantum Cryptography Round 3 Submissions, 2021, https://sphincs.org/resources.html, accessed 15 April 2025
6. Buchmann, J., Dahmen, E., Hülsing, A.: 'XMSS – A practical forward secure signature scheme based on minimal security assumptions'. Proc. Post-Quantum Cryptography (PQCrypto 2011), Berlin, Germany, 2011, pp. 117–139
7. Moody, D., Alagic, G., Apon, D., et al.: 'Status report on the second round of the NIST post-quantum cryptography standardization process', National Institute of Standards and Technology, Gaithersburg, 2020, https://doi.org/10.6028/NIST.IR.8309, accessed 15 April 2025
8. Yehia, M., AlTawy, R., Gulliver, T.A.: 'Verifiable obtained random subsets for improving SPHINCS+'. Proc. Information Security and Privacy, Cham, Switzerland, 2021, pp. 13083
9. Kampanakis, P., Panburana, P., Curcio, M., et al.: 'Post-quantum hash-based signatures for secure boot'. Proc. Silicon Valley Cybersecurity Conference, Cham, Switzerland, 2021, pp. 1383
10. Danielsen, L.E., Melkonian, T.: 'A note on SPHINCS+ parameter sets'. Proc. 15th International Conference on Post-Quantum Cryptography, Berlin, Germany, 2021, pp. 342–356

11. Li, T., Song, L.: 'Privacy-preserving communication-efficient federated multi-armed bandits', IEEE Journal on Selected Areas in Communications, 2022, 40, (3), pp. 773–787
12. Shi, C., Shen, C., Yang, J.: 'Federated multi-armed bandits with personalization'. Proc. 24th International Conference on Artificial Intelligence and Statistics, 2021, pp. 2917–2925
13. JSROJAS: 'IP Network Traffic Flows Labeled with 87 Apps', Kaggle, 2018, https://www.kaggle.com/datasets/jsrojas/ip-network-traffic-flows-labeled-with-87-apps, accessed 20 March 2025
14. SEALLVER: 'SphincsplusSM3', GitHub, 2018, https://github.com/Seallver/SphincsplusSM3, accessed 20 March 2025