Extending the SPHINCS$^+$ Framework: Varying the Tree Heights and Chain Lengths

Zhen Qin[1,2], Siwei Sun[1,2,3⋆], Fangyu Zheng[1]

[1] School of Cryptology, University of Chinese Academy of Sciences, China
[2] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[3] National Engineering Research Center of Multi-dimensional Identification and Trusted Authentication Technology
qinzhen23@mails.ucas.ac.cn, siweisun.isaac@gmail.com, zhengfangyu@ucas.ac.cn

Abstract. The SPHINCS$^+$ framework provides the underlying architecture for modern quantum resistant stateless hash-based signatures. Notable examples include the NIST standard SLH-DSA and its recent variants such as SPHINCS-α and SPHINCS$^+$C. We extend the hypertree structure that underlies the SPHINCS$^+$ framework by allowing trees of different heights to appear on different layers, and we plug generalized hash-based one-time signatures with chains of different lengths into the hypertree. We give a reduction showing how the security proof for the SPHINCS$^+$ framework extends to these structural generalizations. These generalizations lead to an enlarged design space, opening up the possibility for finer-grained trade-offs. We perform a systematic exploration of the parameter space for the generalized structure guided by a thorough theoretical cost analysis that minimizes the number of variables to be enumerated in the searching process. As a result, we identify many parameter sets superior to state-of-the-art stateless hash-based signature schemes in terms of signature size, signing or verification efficiency. In particular, we provide some parameter settings not only enjoying smaller signature size, but also more efficient in signing and verification. The improvement can be significant if we do not pursue optimizing all performance metrics *simultaneously*. One of our 128-bit secure constructions intended to *balance all metrics* with size and verification speed as priorities is 9.8% smaller than SPHINCS$^+$C-128s (27.6% smaller than SPHINCS$^+$-128s, 18.3% smaller than SPHINCS-α-128s, and 22.5% smaller than SPHINCS$^+$FP-128s). Further size reduction is possible with a greater sacrifice in speed. Moreover, we apply our technique to schemes supporting less than 2^{64} signatures. We provide implementations and benchmark results for representative parameter sets.

Keywords: Digital signature, Hash-based signature, WOTS, SPHINCS$^+$, SPHINCS-α, SPHINCS$^+$C

⋆ The corresponding author.

1 Introduction

Digital signatures are public-key cryptographic schemes used to ensure authenticity, integrity, and non-repudiation. They are widely applied in public-key infrastructures, secure software distributions, device secure boot mechanisms, etc. However, today's massively deployed digital signatures that rely on the computational hardness of integer factorization and discrete logarithm problems face the looming threat posed by large scale quantum computers [Sho99]. To prepare for the post-quantum age, a wide range of approaches have been explored to develop quantum-resistant digital signature schemes, including lattices [DLL+17, NIS24a, PFH+20], codes [Ste93, CFS01, BBB+20], isogenies [FKL+20, DLRW24, SEMR24], systems of multivariate polynomial equations [Beu21], MPCitH or VOLEitH techniques [CDG+17, BBB+25], and hash functions [BDH11, ZCY23, HKRY23, NIS24b].

1.1 Related Work

Since the security of hash-based digital signatures reduces to well-studied symmetric assumptions—including pseudorandom-function security and preimage, target-collision, and subset-resilience properties of hash functions—they are considered the most conservative approach to post-quantum digital signatures, whose key techniques date back to works by Lamport [Lam79], Merkle [Mer79], and Goldreich [Gol86]. Due to the confidence in their security, *stateful* hash-based signatures [MCF19, HBG+18] were the earliest quantum-resistant digital signatures to be standardized without going through a competition-based selection process like the NIST Post-Quantum Cryptography project. However, these standardized hash-based signatures update the secret keys after each signing operation, deviating from standard APIs. Moreover, updating the secret keys correctly is essential to the security of these schemes. In general, the statefulness leads to complicated state management issues and limit the scope of applications of these signatures [WBK+24]. Bernstein et al. introduced the `SPHINCS` framework [BHH+15] for designing practical stateless hash-based signatures schemes. After a series of enhancement and optimization, it evolved into the `SPHINCS`$^+$ framework [BHK+19] whose instances specified in FIPS 205 have been adopted by NIST as the stateless hash-based digital signature standard [NIS24b]. Despite its conservative security, the intended application scenarios of hash-based digital signatures is limited due to the cost and efficiency issues. Further optimizing the `SPHINCS`$^+$ framework to improve or balance signature size, signing and verification speed poses significant challenges.

In essence, the `SPHINCS`$^+$ framework integrates instances of one-time signatures (`WOTS`$^+$) and few-time signatures (`FORS`) into a virtual hypertree – a tree of Merkle-like trees. The signature of a message is formed by combining a `FORS` signature and a series of `WOTS`$^+$ signatures and their corresponding authentication paths all the way up to the root of the hypertree. To sign a data element `m` by `WOTS`$^+$, we first encode `m` into a vector $(x_0, \cdots, x_{l-1})$ in $[w]^l = \{0, \cdots, w-1\}^l$, and then the signature is generated by revealing certain data elements of the

virtual hash chains associated with the underlying $\texttt{WOTS}^+$ instances according to $(x_0, \cdots, x_{l-1})$. One optimization strategy for the cost and efficiency of the $\texttt{SPHINCS}^+$ framework is to improve the encoding functions used in the one time signatures based on hash chains.

Injective Encodings. In $\texttt{SPHINCS}^+$ [BHK+19], the Winternitz encoding $f : \mathbb{F}_2^{8n} \to [w]^{l_0+l_1}$ is employed, where $\mathbb{F}_2^{8n}$ (the set of n-byte strings) is the message space, w is a positive integer power of 2 such that $l_0 \cdot \log_2 w = 8n$, and $l_1 = \lfloor \log_2(l_0 \cdot (w-1))/\log_2 w \rfloor + 1$. In Winternitz encoding, we first compute an $l_1 \cdot \log_2 w$-bit checksum $\mathfrak{c}(\mathsf{m})$ of the message $\mathsf{m} \in \mathbb{F}_2^{8n}$, and then break $\mathsf{m} \parallel \mathfrak{c}(\mathsf{m})$ into $l_0 + l_1$ $\log_2 w$-bit chunks with each chunk representing an integer in $[w]$. The virtual data structure associated with the $\texttt{WOTS}^+$ instance employing Winternitz encoding has l_0+l_1 hash chains, and a signature generated by the $\texttt{WOTS}^+$ instance contains $l_0 + l_1$ n-byte elements. In CRYPTO 2023 [ZCY23], Zhang, Cui, and Yu considered general injective encodings from $\mathbb{F}_2^{8n}$ to an *incomparable* subset of $[w]^l$ and proved that the smallest value of l for which such an encoding exits is the smallest positive integer ℓ such that $\left|\mathcal{C}_{\lfloor \frac{\ell\cdot(w-1)}{2} \rfloor}([w]^\ell)\right| \geq |\mathbb{F}_2^{8n}| = 2^{8n}$, where

$$\mathcal{C}_{\lfloor \frac{\ell\cdot(w-1)}{2} \rfloor}([w]^\ell) = \left\{ (x_0, \cdots, x_{\ell-1}) \in [w]^\ell : \sum_{j=0}^{\ell-1} x_j = \left\lfloor \frac{\ell \cdot (w-1)}{2} \right\rfloor \right\}.$$

An injective encoding from $\mathbb{F}_2^{8n}$ to $\mathcal{C}_s([w]^\ell)$ for some constant s is named as a constant-sum encoding. Using such encodings to construct one-time signatures based on hash chains requires less hash chains, leading to smaller signatures.

Non-injective Encodings. Khovratovich, Kudinov, and Wagner proposed to use non-injective, non-uniform encoding functions mapping messages into the top layers (0-th layer to d_0-th layer) of a hyper cube $[w]^\ell$ much larger than the message space [KKW25], where the d-th layer of $[w]^\ell$ is defined as

$$\mathcal{L}_d = \left\{ (x_0, \cdots, x_{\ell-1}) \in [w]^\ell : \ell \cdot (w-1) - \sum_{j=0}^{\ell-1} x_j = d \right\}.$$

In [KKW25], this approach is applied to improve the verifier efficiency at the cost of the degradation of the signing performance. In fact, the proposed encoding strategy exhibits a dual relationship where the fastest verifier corresponds to the slowest signer and vice versa.

The $\texttt{WOTS}^{\texttt{C}}$ Encoding. We name the $\texttt{WOTS}$-like one-time signature scheme used in $\texttt{SPHINCS}^+\texttt{C}$ [HKRY23] by $\texttt{WOTS}^{\texttt{C}}$. In $\texttt{WOTS}^{\texttt{C}}$, an n-byte message is repeatedly hashed with a random counter until the n-byte digest fulfills the following condition: When the digest is divided into $\log_2 w$-bit chunks and regarded as a sequence $(x_0, \cdots, x_{l-1}) \in [w]^l$ of integers, $\sum_{j=0}^{l-1} x_j = s$ and $x_0 = \cdots = x_{z-1} = 0$ for some positive constant integer s and z, or $\sum_{j=0}^{l-1} x_j = s$ with $z = 0$. With this approach, only $l - z$ hash chains are required to build the signature scheme.

Another line of research focuses on improving the few-time signatures [AK25, HKRY23, ZCY22, AE18, AE17]. Notably, Abri and Katz perform a detailed theoretical analysis of the *forced pruning* technique proposed by Hülsing et al. [HKRY23] along with a more thorough exploration of its benefits, showing that forced pruning can improve upon SPHINCS$^+$C in all respects, and can reduce the overall signature size for the "smaller SPHINCS$^+$" variants proposed by Fluhrer and Dang [FD24] by up to 20% with minimal effect on signing time.

In addition, by reducing the supported number of signatures, it is natural to identify smaller and faster SPHINCS$^+$ variants [KP22, FD24]. Recently, NIST plans to standardize a set of SPHINCS$^+$ variants supporting significantly less than 2^{64} signatures which is enough for certain application scenarios.

1.2 Our Contributions

We generalize the one-time signatures based on hash chains with chains of different lengths, and the resulting design is named as *inhomogeneous* hash-chain-based one-time signatures. For WOTS$^+$ and WOTS$^\alpha$, we design *injective* and *incomparable* encodings for this generalization; for WOTS^C, we adapt its counter-based preprocessing to the inhomogeneous chains. Plugging the resulting one-time signatures into SPHINCS$^+$ and its variants leads to size and signing and verification speed improvements. We further extend the hypertree structure of SPHINCS$^+$ and its variants by allowing the Merkle-like trees in different layers to have different heights. Let $h > 0$ be the total height and d be the number of layers such that $h = h_0 + \cdots + h_{d-1}$, where $h_i > 0$ is the height of the trees in the i-th layer of the extended hypertree. For a given h, we show that in essence there is a unique choice of $(h_0, \cdots, h_{d-1})$ that optimizes the performance.

These generalizations enlarge the design space and open up the possibility for finer-grained trade-offs. We conduct a thorough theoretical cost and performance analysis for the inhomogeneous hash-chain-based signatures and the extended hypertree structure, enabling us to design clear strategies for exploring the design space to identify desired parameter settings with minimal searching efforts. With this approach, we find new hash-based signature schemes superior to SPHINCS$^+$, SPHINCS-α, and SPHINCS$^+$C in terms of signature size, signing and/or verification speed. We also show further optimizations and richer trade-offs are possible by combining our framework and the few-time signatures PORS+FP proposed by Abri and Katz's [AK25]. [1] We present some typical results in Table 1. The benchmark results (also with AVX2-optimized implementations) of the schemes listed in Table 1 are given in Table 19 in Section S of the Supplementary Material. We emphasize that Table 1 only gives some representative results, and more comprehensive (but still *non-exhaustive*) trade-offs can be found in Section 5. In addition, we explore in Section 5.5 how to apply our technique to

[1] We identified some minor incorrect calculations in [AK25] and had a discussion with the authors. Abri and Katz confirmed it and the results listed in this paper have taken this into account.

Table 1: A comparison between state-of-the-art hash-based signatures and our representative schemes with 128-bit classical security.

Scheme	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)	Source
SPHINCS$^+$-128f	17088	105194	11870	[NIS24b]
CEDRUS$^+$-128f	16528	104788	5401	Sect. 5.1
CEDRUS$^+$[0x02]	12096	207456	8151	
SPHINCS$^+$-128s	7856	2186220	3928	[NIS24b]
CEDRUS$^+$-128s	7840	2109933	3759	Sect. 5.1
CEDRUS$^+$[0x05]	7184	3762161	3900	
SPHINCS-α-128f	17040	91681	5157	[ZCY23]
CEDRUS$^\alpha$-128f	16560	91576	4670	Sect. 5.2
CEDRUS$^\alpha$[0x02]	12000	177213	5042	
SPHINCS-α-128s	6960	2128599	3581	[ZCY23]
CEDRUS$^\alpha$-128s	6960	2126559	3573	Sect. 5.2
CEDRUS$^\alpha$[0x05]	6528	3896450	3155	
SPHINCS$^+$C-128f	14904	111925	5294	[HKRY23]
CEDRUS$^+$C-128f	14340	111771	4971	Sect. 5.3
CEDRUS$^+$C[0x03]	10504	223680	3253	
SPHINCS$^+$C-128s	6304	2110892	12766	[HKRY23]
CEDRUS$^+$C-128s	6304	2089934	11710	Sect. 5.3
CEDRUS$^+$C[0x06]	5684	4207118	5745	
SPHINCS$^+$FP-128f	16420	105281	12777	[AK25]
CEDRUS$^+$FP-128f	15780	105668	6472	Sect. 5.4
CEDRUS$^+$FP[0x01]	11444	207692	4439	
SPHINCS$^+$FP-128s	7332	2188970	2195	[AK25]
CEDRUS$^+$FP-128s	7172	2155750	2100	Sect. 5.4
CEDRUS$^+$FP[0x03]	6660	3724420	2155	
SPHINCS$^+$C-FP-128f	14504	115271	5289	[AK25]
CEDRUS$^+$C-FP-128f	13908	111580	4968	Sect. 5.4
CEDRUS$^+$C-FP-[0x01]	10040	229803	3191	
SPHINCS$^+$C-FP-128s	6096	2109422	12763	[AK25]
CEDRUS$^+$C-FP-128s	6008	2079282	5866	Sect. 5.4
CEDRUS$^+$C-FP-[0x03]	5400	4139889	9319	

smaller SPHINCS$^+$ variants [FD24] supporting significantly less than 2^{64} signatures, which are expected to be standardized by NIST.

Remark 1. Intuitively, it seems that different chain lengths make parallel implementations and optimizations more difficult. However, this is not the case since in actual signing or verification, chains with different lengths are always produced for both the original SPHINCS$^+$ and our framework.

2 Notations and Preliminaries

For a finite set $\mathbb{G}$, the number of elements in $\mathbb{G}$ is denoted by $|\mathbb{G}|$. We use $\mathbb{Z}_+$ to denote the set of all positive integers and let $\mathbb{N} = \mathbb{Z}_+ \cup \{0\}$. The set of real numbers is denoted by $\mathbb{R}$. Let $\mathbb{F}_2 = \{0, 1\}$ be the binary field and $\mathbb{F}_2^{8n}$ be the set of all n-byte strings. For a positive integer w, $[w]$ is defined to be the set $\{0, \cdots, w-1\}$. The Cartesian product of two sets $\mathbb{A}$ and $\mathbb{B}$ is represented by $\mathbb{A} \times \mathbb{B}$. For $\mathbf{w} = (w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$, the set $[w_0] \times \cdots \times [w_{l-1}]$ is abbreviated as $\prod_{j=0}^{l-1}[w_j]$. When $w_0 = \cdots = w_{l-1} = w$, $\prod_{j=0}^{l-1}[w_j] = [w]^l$. For $i \in \mathbb{N}$ and $m \in \mathbb{Z}_+$, $\mathtt{Bin}(i, m)$ is the m-bit binary representation of i. For example, $\mathtt{Bin}(5, 8) = \mathtt{00000101}$. Let x be a binary string, $\mathtt{Int}(x)$ denotes the unsigned integer it represents.

Definition 1. *Let* $\mathbf{a} = (a_0, \cdots, a_{l-1})$ *and* $\mathbf{b} = (b_0, \cdots, b_{l-1})$ *be elements in* $\mathbb{N}^l$. *Then,* $\mathbf{a}$ *and* $\mathbf{b}$ *are comparable if* $a_i \leq b_i$ *for all* $i \in \{0, \cdots, l-1\}$ *or* $b_i \leq a_i$ *for all* $i \in \{0, \cdots, l-1\}$. *A set* $\mathbb{A} \subseteq \mathbb{N}^l$ *is incomparable if* $|\mathbb{A}| = 1$ *or any two different elements in* $\mathbb{A}$ *are incomparable and a function* $f : \mathbb{F}_2^{8n} \to \mathbb{B} \subseteq \mathbb{N}^l$ *is incomparable if* $f(\mathbb{F}_2^{8n}) \subseteq \mathbb{B}$ *is incomparable.*

Definition 2. *A set* $\mathbb{D} \subseteq \mathbb{R}^n$ *is convex if for any* $(x, y) \in \mathbb{D} \times \mathbb{D}$ *and* $0 \leq \lambda \leq 1$, $\lambda x + (1-\lambda)y \in \mathbb{D}$. *Let* $f : \mathbb{D} \to \mathbb{R}$ *be a real-valued function defined over a convex set* $\mathbb{D}$. f *is called a convex function if for any* $(x, y) \in \mathbb{D} \times \mathbb{D}$ *and* $0 \leq \lambda \leq 1$, $f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y)$.

Theorem 1 (Jensen's Inequality). *Let* $f : \mathbb{R} \to \mathbb{R}$ *be a strictly convex function. For any* $(x_0, \cdots, x_{d-1}) \in \mathbb{R}^d$, *and* $(\lambda_0, \cdots, \lambda_{d-1}) \in \mathbb{R}_+^d$ *with* $\sum_{i=0}^{d-1} \lambda_i = 1$,

$$f\left(\sum_{i=0}^{d-1} \lambda_i x_i\right) \leq \sum_{i=0}^{d-1} \lambda_i f(x_i),$$

and the equality holds if and only if $x_0 = \cdots = x_{d-1}$.

Lemma 1. *Let* $x_0, \cdots, x_{d-1}$ *be positive integers with* $x_0 \leq x_1 \leq \cdots \leq x_{d-1}$ *and* $x_0 + \cdots + x_{d-1} = h$. *If* $|x_i - x_j| \leq 1$ *for any* $(i, j) \in [d] \times [d]$, *then* $x_0 = \cdots = x_{d-k-1} = \lfloor \frac{h}{d} \rfloor$ *and* $x_{d-k} = \cdots = x_{d-1} = \lceil \frac{h}{d} \rceil$, *where* $k = h - d \cdot \lfloor \frac{h}{d} \rfloor$.

Proof. See Section A in Supplementary Material. □

Definition 3. *For given positive integers* A *and* l *with* $A \geq l$, *we define the notation*

$$\left\langle \frac{A}{l} \right\rangle = \min_{\substack{(x_0, \cdots, x_{l-1}) \in \mathbb{Z}_+^l \\ x_0 + \cdots + x_{l-1} = A}} \sum_{i=0}^{l-1} 2^{x_i}.$$

The Definition 3 abbreviates a discrete balancing problem in which an integer budget A is divided among l positive integer components and the resulting exponential costs are summed. Since the function is strictly convex, an unbalanced allocation increases the sum, whereas the minimum is attained when the components differ by at most one.

This same optimization problem arises in both structural generalizations considered in this work. When $x_i = \log_2 w_i$, the quantity gives the minimum possible value of $\sum_i w_i$, and equivalently of $\sum_i (w_i - 1)$ for fixed l, among l power-of-two hash-chain lengths with a fixed total encoding capacity A. When the x_i's represent hypertree-layer heights, it gives the minimum possible value of $\sum_i 2^{x_i}$ for a fixed total height. The following lemma gives an explicit expression for this quantity.

Lemma 2. *For given positive integers A and l with $A \geq l$,*

$$\left\langle \frac{A}{l} \right\rangle = \min_{\substack{(x_0,\cdots,x_{l-1})\in\mathbb{Z}_+^l \\ x_0+\cdots+x_{l-1}=A}} \sum_{i=0}^{l-1} 2^{x_i} = (l-k)\cdot 2^{\lfloor A/l \rfloor} + k \cdot 2^{\lceil A/l \rceil},$$

where $k = A - l \cdot \lfloor \frac{A}{l} \rfloor$. That is, $\sum_{i=0}^{l-1} 2^{x_i}$ reaches its minimum when $(l-k)$ x_i's take on the value $\lfloor \frac{A}{l} \rfloor$, and k x_i's take on the value $\lceil \frac{A}{l} \rceil$.

Proof. See Section B in Supplementary Material. □

3 Inhomogeneous WOTS-like One-time Signatures

Firstly, we briefly describe how to build a one-time signature scheme based on hash chains. Since in the SPHINCS$^+$ framework, messages to be signed by a one-time signature scheme are typically the outputs of some hash functions, we restrict the message space of a one-time signature to be $\mathbb{F}_2^{8n}$, i.e., the set of all n-byte strings. In addition, we first present the construction using a hash function modeled as a random oracle. Modern hash-based one-time signatures instead use address-separated tweakable hash functions. The precise multi-instance reduction for that setting is stated in Theorem 5.

For $(w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$, let $\mathtt{Encode} : \mathbb{F}_2^{8n} \to [w_0] \times \cdots \times [w_{l-1}]$ be an injective and *incomparable* function, that is, $\mathtt{Encode}$ is injective and $\mathtt{Encode}(\mathbb{F}_2^{8n}) = \{\mathtt{Encode}(x) : x \in \mathbb{F}_2^{8n}\} \subseteq [w_0] \times \cdots \times [w_{l-1}]$ is incomparable. Let $\mathsf{H} : \mathbb{F}_2^* \to \mathbb{F}_2^{8n}$ be a random oracle. Then, we can construct a one-time signature scheme $\mathsf{OTS}[\mathsf{H}, \mathtt{Encode}, \mathbb{F}_2^{8n}, (w_0, \cdots, w_{l-1})] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ as follows.

Construction 1 *Let $\mathsf{H} : \mathbb{F}_2^* \to \mathbb{F}_2^{8n}$ be a random oracle. For $(w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$, and an injective and incomparable function $\mathtt{Encode} : \mathbb{F}_2^{8n} \to [w_0] \times \cdots \times [w_{l-1}]$, $\mathsf{OTS}[\mathsf{H}, \mathtt{Encode}, \mathbb{F}_2^{8n}, (w_0, \cdots, w_{l-1})] = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ is a one-time signature scheme with message space $\mathbb{F}_2^{8n}$ such that*

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^{8n})$:

- For $i \in [l]$: $\mathsf{sk}_i \xleftarrow{\$} \mathbb{F}_2^{8n}$, $\mathsf{pk}_i \leftarrow \mathsf{H}^{w_i-1}(\mathsf{sk}_i)$
- $\mathsf{sk} := (\mathsf{sk}_0, \cdots, \mathsf{sk}_{l-1})$, $\mathsf{pk} := (\mathsf{pk}_0, \cdots, \mathsf{pk}_{l-1})$

- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{m})$:
  - $x := (x_0, \cdots, x_{l-1}) \leftarrow \mathtt{Encode}(\mathsf{m})$
  - For $i \in [l]$: $\sigma_i \leftarrow \mathsf{H}^{x_i}(\mathsf{sk}_i)$
  - $\sigma := (\sigma_0, \cdots, \sigma_{l-1}) \in (\mathbb{F}_2^{8n})^l$
- $b \leftarrow \mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma)$:
  - $x = (x_0, \cdots, x_{l-1}) \leftarrow \mathtt{Encode}(\mathsf{m})$
  - For $i \in [l]$: $\mathsf{pk}'_i \leftarrow \mathsf{H}^{w_i-1-x_i}(\sigma_i)$
  - $b := (\mathsf{pk}_0 = \mathsf{pk}'_0) \wedge \cdots \wedge (\mathsf{pk}_{l-1} = \mathsf{pk}'_{l-1})$

Construction 1 gives a simplified random-oracle presentation. A precise multi-instance EU-naCMA security statement for inhomogeneous WOTS-TW is given in Theorem 5 (Section Q in Supplementary Material). Under the same PRF, SM-TCR, SM-PRE, and SM-UD assumptions as in the WOTS-TW security proof, the uniform-chain quantities ℓw and w are replaced by

$$W_\Sigma = \sum_{i=0}^{\ell-1} w_i \qquad \text{and} \qquad w_{\max} = \max_{0 \le i < \ell} w_i,$$

respectively. The proof relies on injectivity and antichainness: for any fresh codeword $\mathbf{x}^*$ and the signed codeword $\mathbf{x}$, there exists an index i such that $x_i^* < x_i$, so a forgery requires recovering an earlier node on at least one hash chain.

For the random-oracle instantiation, we use

$$F(P, T, X) = H(P \parallel T \parallel X),$$

where T is instantiated by the SPHINCS+ address structure ADRS. Its function-type and position fields provide domain separation between different primitive types, WOTS instances, chains, and chain steps. The theorem applies directly to the deterministic injective antichain encodings used by inhomogeneous WOTS+ and inhomogeneous WOTS$^\alpha$. The counter-based preprocessing of WOTS^C is not such an encoding; its reduction additionally contains an $\mathsf{S\text{-}TCR}(\mathsf{Good}_{\mathbf{w},z_b})$ term.

In the following sections, we will see that allowing chains with different lengths leads to improved signature size, signing and verification efficiency. In this work, we consider three variants of WOTS-like one-time signatures, including WOTS$^+$ (used in SPHINCS$^+$ [NIS24b]), WOTS$^\alpha$ (used in SPHINCS-α [ZCY23]), and WOTS$^\mathtt{C}$ (used in SPHINCS$^+$C [HKRY23]). We generalize these constructions to allow them to use hash chains with different lengths, and the generalized constructions are named as *inhomogeneous* WOTS$^+$, WOTS$^\alpha$ and WOTS$^\mathtt{C}$.

To simplify the presentation, we introduce the helper function $\mathtt{IntSeq}_{\mathbf{w}}(\cdot)$, which converts a binary string or an integer to a sequence of integers. Let $\mathbf{w} = (w_0, \cdots, w_{v-1}) \in \mathbb{Z}_+^v$ and s be a $(w_0 + \cdots + w_{v-1})$-bit binary string. Then, $\mathtt{IntSeq}_{\mathbf{w}}(s) = (i_0, \cdots, i_{v-1})$, where i_t for $0 \le t < v$ is the non-negative integer represented by the w_t-bit substring formed by the $((\sum_{j=0}^{t} w_j) - w_t)$-th

bit to $((\sum_{j=0}^{t} w_j) - 1)$-th bit segment of s. When the input is a non-negative integer $k \in \{0, \cdots, 2^{w_0+\cdots+w_{v-1}} - 1\}$, $\texttt{IntSeq}_{\mathbf{w}}(k)$ should be understood as $\texttt{IntSeq}_{\mathbf{w}}(\texttt{Bin}(k, w_0 + \cdots + w_{v-1}))$.

Example 1. Let $s = \texttt{101011}$ be a 6-bit binary string, $k = 27$ be an integer whose binary representation is $\texttt{11011}$, and $\mathbf{w} = (2, 3, 1)$. Then, $\texttt{IntSeq}_{\mathbf{w}}(s) = (2, 5, 1)$, and $\texttt{IntSeq}_{\mathbf{w}}(k) = \texttt{IntSeq}_{\mathbf{w}}(\texttt{Bin}(k, 6)) = \texttt{IntSeq}_{\mathbf{w}}(\texttt{011011}) = (1, 5, 1)$.

3.1 Inhomogeneous $\texttt{WOTS}^+$

First, we adapt the encoding method employed by the regular $\texttt{WOTS}^+$ signatures in $\texttt{SPHINCS}^+$ to work with hash chains of different lengths. Let m be an $8n$-bit string, $w_0, \cdots, w_{l_0-1}, w_{l_0}, \cdots, w_{l_0+l_1-1}$ be positive integer powers of 2 such that the following constraints are fulfilled:

$$\begin{cases} 8n = \log_2 w_0 + \cdots + \log_2 w_{l_0-1} \\ \left\lfloor \log_2 \sum_{i=0}^{l_0-1} (w_i - 1) \right\rfloor + 1 \leq \log_2 w_{l_0} + \cdots + \log_2 w_{l_0+l_1-1} \end{cases}. \tag{1}$$

The new encoding function $\texttt{Encode} : \mathbb{F}_2^{8n} \to [w_0] \times \cdots \times [w_{l_0-1}] \times [w_{l_0}] \times \cdots \times [w_{l_0+l_1-1}]$ maps m to $\mathbf{x} = (x_0, \cdots, x_{l_0-1}, x_{l_0}, \cdots, x_{l_0+l_1-1})$, where

$$\begin{cases} (x_0, \cdots, x_{l_0-1}) = \texttt{IntSeq}_{(\log_2 w_0, \cdots, \log_2 w_{l_0-1})}(\mathsf{m}) \\ (x_{l_0}, \cdots, x_{l_0+l_1-1}) = \texttt{IntSeq}_{(\log_2 w_{l_0}, \cdots, \log_2 w_{l_0+l_1-1})}(\texttt{Checksum}(\mathsf{m})) \end{cases},$$

and $\texttt{Checksum}(\mathsf{m}) = \sum_{j=0}^{l_0-1} (w_j - 1 - x_j)$.

Example 2. Let $n = 1$, $\mathsf{m} = \texttt{10100111} \in \mathbb{F}_2^{8n}$, $w_0 = 4$, $w_1 = 8$, $w_2 = 8$, $w_3 = 4$, $w_4 = 8$, $l_0 = 3$, $l_1 = 2$, and the constraint given by Equation (1) is fulfilled. Then, $f(\mathsf{m}) = (x_0, x_1, x_2, x_3, x_4) \in [4] \times [8] \times [8] \times [4] \times [8]$. Then $(x_0, x_1, x_2) = \texttt{IntSeq}_{(2,3,3)}(\mathsf{m}) = (2, 4, 7)$, $\texttt{Checksum}(\mathsf{m}) = \sum_{j=0}^{2} (w_j - 1 - x_j) = (3 - 2) + (7 - 4) + (7 - 7) = 1 + 3 + 0 = 4$, and $(x_3, x_4) = \texttt{IntSeq}_{(2,3)}(4) = (0, 4)$.

Lemma 3. *The set* $\{\texttt{Encode}(\mathsf{m}) : \mathsf{m} \in \mathbb{F}_2^{8n}\}$ *is incomparable.*

Proof. Let m and m' be any two different elements in $\mathbb{F}_2^{8n}$, and

$$\begin{cases} \texttt{Encode}(\mathsf{m}) = \mathbf{x} = (x_0, \cdots, x_{l_0-1}, x_{l_0}, \cdots, x_{l_0+l_1-1}) \\ \texttt{Encode}(\mathsf{m}') = \mathbf{x}' = (x'_0, \cdots, x'_{l_0-1}, x'_{l_0}, \cdots, x'_{l_0+l_1-1}) \end{cases}.$$

We show that $x_0 \leq x'_0, \cdots, x_{l_0+l_1-1} \leq x'_{l_0+l_1-1}$ is impossible. If the inequality holds, there exists an integer $t \in \{0, \cdots, l_0 - 1\}$ such that $x_t < x'_t$. Otherwise, $x_0 = x'_0, \cdots, x_{l_0-1} = x'_{l_0-1}$ and thus $x_{l_0} = x'_{l_0}, \cdots, x_{l_0+l_1-1} = x'_{l_0+l_1-1}$, which contradicts that $\mathbf{x}$ and $\mathbf{x}'$ are different. Then, for m and m', we have

$$\texttt{Checksum}(\mathsf{m}) = \sum_{j=0}^{l_0-1} (w_j - 1 - x_j) > \sum_{j=0}^{l_0-1} (w_j - 1 - x'_j) = \texttt{Checksum}(\mathsf{m}').$$

However, according to the definition of $\mathtt{IntSeq}_{\mathbf{w}}(\cdot)$,

$$\begin{cases} \mathtt{Checksum}(\mathsf{m}) = x_{l_0} \cdot \prod_{i=l_0+1}^{l_0+l_1-1} w_i + x_{l_0+1} \cdot \prod_{i=l_0+2}^{l_0+l_1-1} w_i + \cdots + x_{l_0+l_1-2} \cdot w_{l_0+l_1-1} + x_{l_0+l_1-1} \\ \mathtt{Checksum}(\mathsf{m}') = x'_{l_0} \cdot \prod_{i=l_0+1}^{l_0+l_1-1} w_i + x'_{l_0+1} \cdot \prod_{i=l_0+2}^{l_0+l_1-1} w_i + \cdots + x'_{l_0+l_1-2} \cdot w_{l_0+l_1-1} + x'_{l_0+l_1-1} \end{cases},$$

which implies that $\mathtt{Checksum}(\mathsf{m}) \leq \mathtt{Checksum}(\mathsf{m}')$. This is a contradiction. □

According to the definition of `Encode` and Lemma 3, `Encode` is injective and incomparable. Therefore,

$$\Phi = \mathsf{OTS}[\mathsf{H}, \mathtt{Encode}, \mathbb{F}_2^{8n}, (w_0, \cdots, w_{l_0-1}, w_{l_0}, \cdots, w_{l_0+l_1-1})]$$

is a one-time signature scheme. Let $\mathfrak{C}_{\mathsf{Sign}}(\Phi)$, $\mathfrak{C}_{\mathsf{Ver}}(\Phi)$, and $\mathfrak{C}_{\mathsf{Size}}(\Phi)$ denote the signing cost, verification cost, and signature size of Φ, respectively. According to Construction 1, we have the following estimation

$$\begin{cases} \mathfrak{C}_{\mathsf{Size}}(\Phi) = (l_0 + l_1) \cdot n \\ \mathfrak{C}_{\mathsf{Sign}}(\Phi) = w_0 + \cdots + w_{l_0-1} + w_{l_0} + \cdots + w_{l_0+l_1-1} \\ \mathfrak{C}_{\mathsf{Ver}}(\Phi) = (w_0 - 1) + \cdots + (w_{l_0-1} - 1) + (w_{l_0} - 1) + \cdots + (w_{l_0+l_1-1} - 1) \end{cases}, \quad (2)$$

where the unit for $\mathfrak{C}_{\mathsf{Sign}}(\Psi)$ and $\mathfrak{C}_{\mathsf{Ver}}(\Psi)$ is the number of hash function evaluations and the unit for $\mathfrak{C}_{\mathsf{Size}}(\Psi)$ is the number of bytes. Next, we present a searching strategy for identifying parameters that achieve a specific design goal concerning the signature size and signing and verification efficiency.

Searching Strategy. First of all, we assume that n is given. Our search needs to determine l_0, l_1, and $(w_0, \cdots, w_{l_0-1}, w_{l_0}, \cdots, w_{l_0+l_1-1})$. In the following, we give some lemmas and propositions that reduce the search space significantly. Basically, these lemmas indicates that when the values of some parameters are given, the optimal values of some other parameters can be computed deterministically *without* any search, and using other values rather than the computed ones definitely leads to inferior one-time signatures with respect to signature size, signing and verification performance.

Lemma 4. *Let A and l be given positive integers and $A \geq l$. Then,*

$$\left\langle \frac{A}{l} \right\rangle = \min_{\substack{(\log_2 w_0, \cdots, \log_2 w_{l-1}) \in \mathbb{Z}_+^l \\ \log_2 w_0 + \cdots + \log_2 w_{l-1} = A}} \sum_{i=0}^{l-1} w_i.$$

Proof. Apply Lemma 2 with $x_i = \log_2 w_i$. □

Proposition 1. *Let $(w_0, \cdots, w_{l_0+l_1-1})$ be any given vector in $\mathbb{Z}_+^{l_0+l_1}$ such that* Equation (1) *is fulfilled. Let $\Phi = \mathsf{OTS}[\mathsf{H}, \mathtt{Encode}, \mathbb{F}_2^{8n}, (w_0, \cdots, w_{l_0-1}, w_{l_0}, \cdots, w_{l_0+l_1-1})]$ be a one-time signature scheme. Let $\hat{\Phi} = \mathsf{OTS}[\mathsf{H}, \mathtt{Encode}, \mathbb{F}_2^{8n}, (u_0, \cdots, u_{l_0-1}, w_{l_0}, \cdots, w_{l_0+l_1-1})]$ be another one-time signature scheme with $\log_2 u_0 = \cdots = \log_2 u_{l_0-k-1} = \left\lfloor \frac{8n}{l_0} \right\rfloor$, $\log_2 u_{l_0-k} = \cdots = \log_2 u_{l_0-1} = \left\lceil \frac{8n}{l_0} \right\rceil$, and $k = 8n - l_0 \cdot \left\lfloor \frac{8n}{l_0} \right\rfloor$. Then $\mathfrak{C}_{\mathsf{Size}}(\hat{\Phi}) \leq \mathfrak{C}_{\mathsf{Size}}(\Phi)$, $\mathfrak{C}_{\mathsf{Sign}}(\hat{\Phi}) \leq \mathfrak{C}_{\mathsf{Sign}}(\Phi)$, and $\mathfrak{C}_{\mathsf{Ver}}(\hat{\Phi}) \leq \mathfrak{C}_{\mathsf{Ver}}(\Phi)$.*

Proof. According to Equation (2), $\mathfrak{C}_{\mathsf{Size}}(\hat{\Phi}) = \mathfrak{C}_{\mathsf{Size}}(\Phi) = (l_0 + l_1) \cdot n$. Due to Lemma 4 and Definition 3,

$$\left\langle \frac{8n}{l_0} \right\rangle = \min_{\substack{(\log_2 v_0, \cdots, \log_2 v_{l_0-1}) \in \mathbb{Z}_+^{l_0} \\ \log_2 v_0 + \cdots + \log_2 v_{l_0-1} = 8n}} \sum_{i=0}^{l_0-1} v_i = u_0 + \cdots u_{l_0-1}.$$

Hence, by Equation (2), $\mathfrak{C}_{\mathsf{Sign}}(\hat{\Phi}) = u_0 + \cdots + u_{l_0-1} + w_{l_0} + \cdots + w_{l_0+l_1-1}$ is smaller or equal to $\mathfrak{C}_{\mathsf{Sign}}(\Phi) = w_0 + \cdots + w_{l_0-1} + w_{l_0} + \cdots + w_{l_0+l_1-1}$, and similarly, $\mathfrak{C}_{\mathsf{Ver}}(\hat{\Phi}) \leq \mathfrak{C}_{\mathsf{Ver}}(\Phi)$, □

Proposition 1 shows that for a given value of (n, l_0), we should always choose $(u_0, \cdots, u_{l_0-1})$ with $\log_2 u_0 = \cdots = \log_2 u_{l_0-k-1} = \left\lfloor \frac{8n}{l_0} \right\rfloor$, $\log_2 u_{l_0-k} = \cdots = \log_2 u_{l_0-1} = \left\lceil \frac{8n}{l_0} \right\rceil$, and $k = 8n - l_0 \cdot \left\lfloor \frac{8n}{l_0} \right\rfloor$ to be the value of $(w_0, \cdots, w_{l_0-1})$. Since there is no hope for other choices to outperform $(u_0, \cdots, u_{l_0-1})$.

Next, we show that given $(n, l_0, (w_0, \cdots, w_{l_0-1}), l_1)$, the optimal value of $(w_{l_0}, \cdots, w_{l_0+l_1-1})$ can be determined.

Lemma 5. *Let n, l_0, $l_1 \in \mathbb{Z}_+$ and $(w_0, \cdots, w_{l_0-1})$ be positive integer powers of 2 such that $8n = \log_2 w_0 + \cdots + \log_2 w_{l_0-1}$. For $T = \left\lfloor \log_2 \sum_{i=0}^{l_0-1} (w_i - 1) \right\rfloor + 1$,*

$$\min_{\substack{(\log_2 w_{l_0}, \cdots, \log_2 w_{l_0+l_1-1}) \in \mathbb{Z}_+^{l_1} \\ T \leq \log_2 w_{l_0} + \cdots + \log_2 w_{l_0+l_1-1}}} \sum_{i=l_0}^{l_0+l_1-1} w_i = (l_1 - k) \cdot 2^{\lfloor T/l_1 \rfloor} + k \cdot 2^{\lceil T/l_1 \rceil},$$

where $k = T - l_1 \cdot \lfloor \frac{T}{l_1} \rfloor$.

Proof. See Section C in Supplementary Material. □

Note that when the value of $(n, l_0, (w_0, \cdots, w_{l_0-1}), l_1)$ is given, the size of the signature is fixed to $(l_0 + l_1) \cdot n$. Lemma 5 tells us choosing $(w_{l_0}, \cdots, w_{l_0+l_1-1})$ to be $(\lfloor T/l_1 \rfloor, \cdots, \lfloor T/l_1 \rfloor, \lceil T/l_1 \rceil, \cdots, \lceil T/l_1 \rceil)$ minimizes $w_0 + \cdots + w_{l_0-1} + w_{l_0} + \cdots + w_{l_0+l_1-1}$ or maximizes the signing and verification speed.

Based on the above lemmas and proposition, we come up with the following search procedure. First, we globally assume that for $0 \leq i < l_0 + l_1$, $4 \leq w_i \leq 256$. Then, we have $\frac{8n}{\log_2 256} \leq l_0 \leq \frac{8n}{\log_2 4}$ or $l_0 \in \{n, n+1, \cdots, 4n\}$. For each l_0, we can derive the optimal value of $(w_0, \cdots, w_{l_0-1})$ based on Proposition 1. Then,

$$\frac{\left\lfloor \log_2 \left(\left\langle \frac{8n}{l_0} \right\rangle - l_0 \right) \right\rfloor + 1}{\log_2 256} \leq l_1 \leq \frac{\left\lfloor \log_2 \left(\left\langle \frac{8n}{l_0} \right\rangle - l_0 \right) \right\rfloor + 1}{\log_2 4}.$$

For any given $(n, l_0, (w_0, \cdots, w_{l_0-1}), l_1)$, we can derive the optimal value of $(w_{l_0}, \cdots, w_{l_0+l_1-1})$ according to Lemma 5. Therefore, for any given n, we only

need to check at most $(3n+1)\cdot|\mathcal{L}|$ parameter sets, where

$$\mathcal{L}=\left\{l_1\in\mathbb{Z}:\frac{\left\lfloor\log_2\left(\left\langle\frac{8n}{l_1}\right\rangle-l_0\right)\right\rfloor+1}{\log_2 256}\le l_1\le\frac{\left\lfloor\log_2\left(\left\langle\frac{8n}{l_0}\right\rangle-l_0\right)\right\rfloor+1}{\log_2 4}\right\}.$$

For parameter settings with the same value of l_0+l_1, we only retain the one minimizing $\sum_{i=0}^{l_0+l_1-1}w_i$. For example, when $n=32$, we only need to check 99 different parameter settings, which can be done within seconds. With this procedure, we identify several inhomogeneous $\texttt{WOTS}^+$ schemes outperforming the regular $\texttt{WOTS}^+$ schemes used in the $\texttt{SPHINCS}^+$ algorithms specified in FIPS 205 [NIS24b]. Some representative results are listed in Table 2 and more trade-offs can be found in Table 10, Table 11 and Table 12 in Supplementary Material.

Table 2: Representative regular and inhomogeneous $\texttt{WOTS}^+$ schemes, where the signature size is in bytes, and signing and verification are in terms of the number of hash function calls.

BitSec	Regular $\texttt{WOTS}^+$				Inhomogeneous $\texttt{WOTS}^+$			
	w	Signature Size	Signing	Verification	**w**	Signature Size	Signing	Verification
128	$[16]^{35}$	560	560	525	$[16]^{32}\times[8]^3$	560	536	501
					$[16]^{32}\times[16]\times[32]$	544	560	526
192	$[16]^{51}$	1224	816	765	$[16]^{48}\times[8]^2\times[16]$	1224	800	749
					$[16]^{48}\times[32]^2$	1200	832	782
256	$[16]^{67}$	2144	1072	1005	$[16]^{64}\times[8]^2\times[16]$	2144	1056	989
					$[16]^{64}\times[32]^2$	2112	1088	1022

3.2 Inhomogeneous $\texttt{WOTS}^{\alpha}$

In this section, we adapt the constant-sum encoding employed by the $\texttt{WOTS}^{\alpha}$ signatures in $\texttt{SPHINCS}$-α to work with hash chains of different lengths. Let l, w_0, $\cdots$, w_{l-1} be positive integers. For $s\in[1+\sum_{j=0}^{l-1}(w_j-1)]$, we define $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$ to be the set $\left\{(v_0,\cdots,v_{l-1})\in\prod_{j=0}^{l-1}[w_j]:\sum_{j=0}^{l-1}v_j=s\right\}$.

Lemma 6. *For $s\in[1+\sum_{j=0}^{l-1}(w_j-1)]$, $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$ is incomparable.*

Proof. For any two different elements $\mathbf{v},\mathbf{v}'\in\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$ with $v_0\le v_0'$, $\cdots$, $v_{l-1}\le v_{l-1}'$, there exist an integer $t\in\{0,\cdots,l-1\}$ such that $v_t<v_t'$. Hence, $v_0+\cdots+v_{l-1}<v_0'+\cdots+v_{l-1}'$, which constraints with $v_0+\cdots+v_{l-1}=v_0'+\cdots+v_{l-1}'=s$. □

Since $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$ is incomparable, for each injective function $f:\mathbb{F}_2^{8n}\to\mathcal{C}_{\mathbf{s}}(\prod_{j=0}^{l-1}[w_j])$, we can build a one-time signature scheme based on Construction 1. Before we give a concrete injective function $f:\mathbb{F}_2^{8n}\to\mathcal{C}_{\mathbf{s}}(\prod_{j=0}^{l-1}[w_j])$, we investigate the number of elements in $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$.

Lemma 7. *If* $s \in [1 + \sum_{j=0}^{l-1}(w_j - 1)]$ *and* $l > 1$, *then*

$$\left|\mathcal{C}_s\left(\prod_{j=0}^{l-1}[w_j]\right)\right| = \sum_{i=0}^{\min\{w_{l-1}-1,s\}}\left|\mathcal{C}_{s-i}\left(\prod_{j=0}^{l-2}[w_j]\right)\right|.$$

Proof. See Section G in Supplementary Material. □

Hereafter, for $\mathbf{w} = (w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$, $1 < k < l$, and $0 \le u$, let $\zeta_{\mathbf{w}}(k,u) = |\mathcal{C}_u(\prod_{j=0}^{k-1}[w_j])|$. Therefore, when $u > (w_0 - 1) + \cdots + (w_{k-1} - 1)$,

$$\zeta_{\mathbf{w}}(k,u) = \left|\left\{\mathbf{x} = (x_0, \cdots, x_{k-1}) \in \prod_{j=0}^{k-1}[w_j] : \sum_{j=0}^{k-1} x_j = u\right\}\right| = |\emptyset| = 0. \tag{3}$$

According to Lemma 7 and Equation (3), for $\mathbf{w} = (w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$, $1 < k < l$, and $0 \le u$,

$$\zeta_{\mathbf{w}}(k,u) = \sum_{i=0}^{\min\{w_{k-1}-1,u\}} \zeta_{\mathbf{w}}(k-1, u-i). \tag{4}$$

In the recurrence relation given in Equation (4), $k > 1$ is required since when $k = 1$, the right-hand side of Equation (4) has terms like $\zeta_{\mathbf{w}}(0,j)$, which is not well defined. To make Equation (4) true for $k = 1$, we artificially define

$$\begin{cases} \zeta_{\mathbf{w}}(0,0) = 1 \\ \zeta_{\mathbf{w}}(0,t) = 0, \quad t > 0 \end{cases}.$$

Under this definition, we have the following proposition.

Proposition 2. *For* $\mathbf{w} = (w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$, $1 \le k < l$, *and* $0 \le u$,

$$\zeta_{\mathbf{w}}(k,u) = \sum_{i=0}^{\min\{w_{k-1}-1,u\}} \zeta_{\mathbf{w}}(k-1, u-i).$$

Proposition 2 gives us a method for computing the number of elements in $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$, and Table 3 presents an example for $\mathbf{w} = (w_0, w_1, w_2, w_3) = (2, 3, 4, 5)$. Note that in Table 3, $\zeta_{\mathbf{w}}(l,s)$ is equal to the sum of $\min\{w_{l-1}-1, s\}+1$ elements in the row directly above it. For example

$$\begin{cases} \zeta_{\mathbf{w}}(3,2) = \zeta_{\mathbf{w}}(2,2) + \zeta_{\mathbf{w}}(2,1) + \zeta_{\mathbf{w}}(2,0) = 2 + 2 + 1 = 5 \\ \zeta_{\mathbf{w}}(4,6) = \zeta_{\mathbf{w}}(3,6) + \zeta_{\mathbf{w}}(3,5) + \zeta_{\mathbf{w}}(3,4) + \zeta_{\mathbf{w}}(3,3) + \zeta_{\mathbf{w}}(3,2) = 20 \end{cases}.$$

It is also possible to derive the analytical formula of $\zeta_{\mathbf{w}}(l,s)$.

Table 3: $\zeta_{\mathbf{w}}(l,s)$ for $\mathbf{w}=(2,3,4,5)$

	$s=0$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$
$l=1$	1	1	0	0	0	0	0
$l=2$	1	2	2	1	0	0	0
$l=3$	1	3	5	6	5	3	1
$l=4$	1	4	9	15	20	22	20

Theorem 2. *For $l \geq 1$, $\mathbf{w}=(w_0,\cdots,w_{l-1}) \in \mathbb{Z}_+^l$, and $s \in [1+\sum_{j=0}^{l-1}(w_j-1)]$,*

$$\zeta_{\mathbf{w}}(l,s) = \binom{s+l-1}{l-1} - \sum_{k=1}^{l}(-1)^{k-1} \sum_{\substack{(i_1,\cdots,i_k)\in[l]^k \\ i_1<\cdots<i_k \\ w_{i_1}+\cdots+w_{i_k}\leq s}} \binom{s+l-w_{i_1}-\cdots-w_{i_k}-1}{l-1}.$$

Proof. See Section H in Supplementary Material. □

Theorem 3. *Let $\mathbf{w}=(w_0,\cdots,w_{l-1}) \in \mathbb{Z}_+^l$, $l \geq 1$, and $0 \leq s \leq \left\lceil \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rceil$. Then, $\zeta_{\mathbf{w}}(l,j) \leq \zeta_{\mathbf{w}}(l,s)$ for any $j \in \{0,\cdots,s\}$.*

Proof. See Section I in Supplementary Material. □

Theorem 4. *Let $\mathbf{w}=(w_0,\cdots,w_{l-1}) \in \mathbb{Z}_+^l$ and $l \geq 1$. If $\mathcal{C} \subseteq \prod_{j=0}^{l-1}[w_j]$ is incomparable, then $|\mathcal{C}| \leq |\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])|$, where $\mathsf{s} = \left\lfloor \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rfloor$.*

Proof. See Section J in Supplementary Material. □

To build an injective encoding from $\mathbb{F}_2^{8n}$ to $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$, similar to [ZCY23], we first establish a one-to-one mapping $\rho_{\mathbf{w}}^s : [|\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])|] \to \mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$ for $\mathbf{w}=(w_0,\cdots,w_{l-1}) \in \mathbb{Z}_+^l$ and $0 \leq s \leq \left\lfloor \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rfloor$.

Lemma 8. *Let l be a positive integer, $\mathbf{w}=(w_0,\cdots,w_{l-1}) \in \mathbb{Z}_+^l$, and $0 \leq s \leq \left\lfloor \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rfloor$. The mapping $\rho_{\mathbf{w}}^s$ implemented by* Algorithm 1 *is bijective.*

Proof. See Section K in Supplementary Material. □

To help the readers understand the one-to-one mapping given by Algorithm 1, we provide an example in Table 4. It is quite straightforward to build an injective encoding $\tau_{\mathbf{w}}^s : \mathbb{F}_2^{8n} \to \mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$ with $\rho_{\mathbf{w}}^s$ when $|\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])| \geq |\mathbb{F}_2^{8n}|$. In this encoding, we simply map $x \in \mathbb{F}_2^{8n}$ to $\tau_{\mathbf{w}}^s(x) = \rho_{\mathbf{w}}^s(\mathtt{Int}(x))$. Since $\tau_{\mathbf{w}}^s$ is injective and incomparable, $\Phi = \mathsf{OTS}[\mathsf{H}, \tau_{\mathbf{w}}^s, \mathbb{F}_2^{8n}, \mathbf{w}=(w_0,\cdots,w_{l-1})]$ is a

Algorithm 1: $\rho^s_{\mathbf{w}} : [|\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])|] \to \mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$

Input: $\mathsf{x} \in [|\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])|]$

Output: $y = \rho^s_{\mathbf{w}}(\mathsf{x}) \in \mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])$

$x \leftarrow \mathsf{x}$
$s \leftarrow \mathsf{s}$
for i from l to 1 **do**
  for u from 0 to $\min(w_{i-1} - 1, s)$ **do**
    if $x \geq \zeta_{\mathbf{w}}(i-1, s-u)$ **then**
      $x \leftarrow x - \zeta_{\mathbf{w}}(i-1, s-u)$
    else
      $y_{l-i} \leftarrow u$
      break
  $s \leftarrow s - y_{l-i}$
return $(y_0, \cdots, y_{l-1})$

one-time signature scheme. Let $\mathfrak{C}_{\mathsf{Sign}}(\Phi)$, $\mathfrak{C}_{\mathsf{Ver}}(\Phi)$, and $\mathfrak{C}_{\mathsf{Size}}(\Phi)$ denote the signing cost, verification cost, and signature size of Φ, respectively. According to Construction 1, we have the following estimation

$$\begin{cases} \mathfrak{C}_{\mathsf{Size}}(\Phi) = l \cdot n \\ \mathfrak{C}_{\mathsf{Sign}}(\Phi) = \left\lfloor \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rfloor + l \\ \mathfrak{C}_{\mathsf{Ver}}(\Phi) = \left\lceil \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rceil \end{cases}, \tag{5}$$

where the unit for $\mathfrak{C}_{\mathsf{Sign}}(\Psi)$ and $\mathfrak{C}_{\mathsf{Ver}}(\Psi)$ is the number of hash function evaluations and the unit for $\mathfrak{C}_{\mathsf{Size}}(\Psi)$ is the number of bytes.

Table 4: $\rho^s_{\mathbf{w}}$ with $\mathbf{w} = (3, 4, 5)$ and $s = 4$

$0 \mapsto (0,3,2)$	$1 \mapsto (1,2,2)$	$2 \mapsto (1,3,1)$	$3 \mapsto (2,1,2)$	$4 \mapsto (2,2,1)$
$5 \mapsto (2,3,0)$	$6 \mapsto (3,0,2)$	$7 \mapsto (3,1,1)$	$8 \mapsto (3,2,0)$	$9 \mapsto (4,0,1)$
$10 \mapsto (4,1,0)$				

Searching Strategy. First of all, we assume that n is given. Our search needs to determine l and $(w_0, \cdots, w_{l-1})$. In the following, we give some lemmas based on which our search strategy is developed.

Lemma 9. *Let $A \leq A'$. For any $\mathbf{w} = (w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$ with $w_0 + \cdots + w_{l-1} = A$, there exists $\mathbf{w}' = (w'_0, \cdots, w'_{l-1}) \in \mathbb{Z}_+^l$ with $w'_0 + \cdots + w'_{l-1} = A'$ such that $|\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])| \leq |\mathcal{C}_{\mathsf{s}'}(\prod_{j=0}^{l-1}[w'_j])|$, where $\mathsf{s} = \left\lfloor \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rfloor$ and $\mathsf{s}' = \left\lfloor \frac{(w'_0-1)+\cdots+(w'_{l-1}-1)}{2} \right\rfloor$.*

Proof. For $A \leq A'$, and $w_0+\cdots+w_{l-1} = A$, it is easy to find $\mathbf{w}' = (w'_0, \cdots, w'_{l-1})$ in $\mathbb{Z}_+^l$ with $w'_0 + \cdots + w'_{l-1} = A'$ and $w_j \leq w'_j$ for $0 \leq j < l$. According to Theorem 4, we have $|\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w'_j])| \leq |\mathcal{C}_{\mathsf{s}'}(\prod_{j=0}^{l-1}[w'_j])|$. In addition,

$$\mathcal{C}_{\mathsf{s}}\left(\prod_{j=0}^{l-1}[w_j]\right) = \left\{(x_0, \cdots, x_{l-1}) \in \prod_{j=0}^{l-1}[w_j] : x_0 + \cdots + x_{l-1} = \mathsf{s}\right\}$$

is a subset of

$$\mathcal{C}_{\mathsf{s}}\left(\prod_{j=0}^{l-1}[w'_j]\right) = \left\{(x_0, \cdots, x_{l-1}) \in \prod_{j=0}^{l-1}[w'_j] : x_0 + \cdots + x_{l-1} = \mathsf{s}\right\},$$

which implies that $|\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])| \leq |\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w'_j])| \leq |\mathcal{C}_{\mathsf{s}'}(\prod_{j=0}^{l-1}[w'_j])|$. □

Lemma 10. *Let A and l be positive integers. Then*

$$\max_{\substack{(w_0,\cdots,w_{l-1})\in\mathbb{Z}_+^l \\ w_0+\cdots+w_{l-1}=A}} \left|\mathcal{C}_{\left\lfloor \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rfloor}\left(\prod_{j=0}^{l-1}[w_j]\right)\right| = \left|\mathcal{C}_{\left\lfloor \frac{\sum_{i=0}^{l-1}(\hat{w}_i-1)}{2} \right\rfloor}\left(\prod_{j=0}^{l-1}[\hat{w}_j]\right)\right|,$$

where $\hat{w}_0 = \cdots = \hat{w}_{l-k-1} = \lfloor \frac{A}{l} \rfloor$, $\hat{w}_{l-k} = \cdots = \hat{w}_{l-1} = \lceil \frac{A}{l} \rceil$, and $k = A - l \cdot \lfloor \frac{A}{l} \rfloor$.

Proof. See Section L in Supplementary Material. □

Based on Equation (5), we develop the following search strategy for identifying parameters that achieve a specific design goal. In short, for each $\mathsf{w} \in \{8, 9, \cdots, 49\}$, with minimizing the signature size as the first priority under the constraint $\mathbf{w} \in [\mathsf{w}]^l$, we determine the value of l and then identify the parameter setting for $\mathsf{OTS}[\mathsf{H}, \tau_{\mathbf{w}}^s, \mathbb{F}_2^{8n}, \mathbf{w} = (w_0, \cdots, w_{l-1})]$ that achieves the fastest signing and verification speeds. To elaborate, for each $\mathsf{w} \in \{8, 9, \cdots, 49\}$, we compute

$$\ell = \min_{l \in \mathbb{Z}_+} \left\{ l : \mathcal{C}_{\lfloor \frac{l\cdot(\mathsf{w}-1)}{2} \rfloor}\left([\mathsf{w}]^l\right) \geq |\mathbb{F}_2^{8n}| \right\}.$$

According to Theorem 4, Lemma 9 and Lemma 10, ℓ is the smallest value l can take such that $\mathsf{OTS}[\mathsf{H}, \tau_{\mathbf{w}}^s, \mathbb{F}_2^{8n}, \mathbf{w} = (w_0, \cdots, w_{l-1})]$ forms a valid one-time signature under the constraint $\mathbf{w} \in [\mathsf{w}]^l$. For given (w, ℓ), we identify

$$\pi = \min_{(w_0,\cdots,w_{\ell-1})\in\mathbb{Z}_+^\ell} \mathbb{A},$$

where $\mathbb{A} = \left\{ A : \sum_{i=0}^{\ell-1} w_i = A, \mathcal{C}_{\left\lfloor \frac{\sum_{i=0}^{\ell-1}(w_i-1)}{2} \right\rfloor}\left(\prod_{j=0}^{\ell-1}[w_j]\right) \geq |\mathbb{F}_2^{8n}| \right\}$ through a binary search procedure given by Algorithm 2. Note that $\mathsf{w} \cdot \ell \in \mathbb{A}$, and thus $\pi \leq \mathsf{w} \cdot \ell$. Then, we set $\mathbf{w} = (w_0, \cdots, w_{\ell-1})$ with $w_0 = \cdots = w_{\ell-k-1} = \lfloor \frac{\pi}{\ell} \rfloor$,

Algorithm 2: A binary search for π

Input: l and w
Output: π

$\alpha \leftarrow l$
$\beta \leftarrow w \cdot l$
while $\alpha \leq \beta$ **do**
  $\theta \leftarrow \alpha + \lfloor \frac{\beta-\alpha}{2} \rfloor$
  $k \leftarrow \theta - l \cdot \lfloor \frac{\theta}{l} \rfloor$
  $(w_0, \cdots, w_{l-k-1}) \leftarrow \left(\lfloor \frac{\theta}{l} \rfloor, \cdots, \lfloor \frac{\theta}{l} \rfloor\right)$
  $(w_{l-k}, \cdots, w_{l-1}) \leftarrow \left(\lceil \frac{\theta}{l} \rceil, \cdots, \lceil \frac{\theta}{l} \rceil\right)$
  $s \leftarrow \left\lfloor \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rfloor$
  if $\left|\mathcal{C}_s\left(\prod_{j=0}^{l-1}[w_j]\right)\right| \geq |\mathbb{F}_2^{8n}|$ **then**
    $\pi \leftarrow \theta$
    $\beta \leftarrow \theta - 1$
  else
    $\alpha \leftarrow \theta + 1$
return π

$w_{\ell-k} = \cdots = w_{\ell-1} = \lceil \frac{\pi}{\ell} \rceil$, and $k = \pi - \ell \cdot \lfloor \frac{\pi}{\ell} \rfloor$. According to Lemma 9 and Equation (5), this assignment identifies the scheme in

$$\left\{\mathsf{OTS}[\mathsf{H}, \tau_{\mathbf{w}}^s, \mathbb{F}_2^{8n}, \mathbf{w} = (w_0, \cdots, w_{\ell-1})] : \mathbf{w} \in [\mathsf{w}]^\ell\right\}$$

with the fastest signing and verification speed. With this strategy, we identify several inhomogeneous $\mathtt{WOTS}^\alpha$ schemes outperforming the regular $\mathtt{WOTS}^\alpha$ schemes used in the $\mathtt{SPHINCS}$-α algorithms specified in [ZCY23]. Some representative results are listed in Table 5 and more trade-offs can be found in Table 13, Table 14 and Table 15 in Supplementary Material.

Table 5: Representative regular and inhomogeneous $\mathtt{WOTS}^\alpha$ schemes, where the signature size is in bytes, and signing and verification are in terms of the number of hash function calls.

BitSec	Regular $\mathtt{WOTS}^\alpha$				Inhomogeneous $\mathtt{WOTS}^\alpha$			
	w	Signature Size	Signing	Verification	**w**	Signature Size	Signing	Verification
128	$[13]^{37}$	592	259	222	$[12]^{27} \times [13]^{10}$	592	245	209
	$[32]^{27}$	432	445	419	$[31] \times [32]^{26}$	432	445	418
192	$[15]^{51}$	1224	408	357	$[14]^{42} \times [15]^{9}$	1224	403	353
	$[38]^{38}$	912	741	703	$[38]^{38}$	912	741	703
256	$[16]^{66}$	2112	561	495	$[15]^{15} \times [16]^{51}$	2112	553	488
	$[46]^{48}$	1536	1128	1080	$[45]^{41} \times [46]^{7}$	1536	1107	1060

3.3 Inhomogeneous $\texttt{WOTS}^{\mathsf{C}}$

We adapt the counter-based preprocessing employed by $\texttt{WOTS}^{\mathsf{C}}$ in $\texttt{SPHINCS}^{+}\mathsf{C}$ to hash chains of different lengths. Let $\mathsf{m} \in \mathbb{F}_2^{8n}$, let $z_b \in \mathbb{N}$, and let $(w_0, \ldots, w_{l-1})$ be positive powers of two such that $8n - z_b = \log_2 w_0 + \cdots + \log_2 w_{l-1}$. For an the $\texttt{WOTS}^{\mathsf{C}}$ preprocessing hash $\mathsf{H}_{\mathsf{Root}}$, let T denote the base $\texttt{WOTS}^{\mathsf{C}}$-instance address consisting of all non-counter fields. The signer enumerates a four-byte counter γ and writes

$$d = \mathsf{H}_{\mathsf{Root}}(P, T, \mathsf{m} \parallel \gamma)$$

for the derived hash obtained by inserting γ into the dedicated counter field of the implementation's $\texttt{WOTS}^{\mathsf{C}}$ root-hash address. Since γ occupies a dedicated counter field, distinct pairs (T, γ) yield distinct $\texttt{WOTS}^{\mathsf{C}}$ addresses, thereby preserving domain separation. Thus T remains fixed throughout the counter search in the abstract S-TCR experiment, although γ occupies an address word in the implementation. The signer continues until $\mathsf{Good}_{\mathbf{w}, z_b}(d) = 1$. This predicate requires the designated z_b bits of d to be zero and, for the remaining string $\mathbf{y} \in \mathbb{F}_2^{8n - z_b}$,

$$\mathbf{x} = (x_0, \cdots, x_{l-1}) = \texttt{IntSeq}_{(\log_2 w_0, \cdots, \log_2 w_{l-1})}(\mathbf{y}) \in \mathcal{C}_{\mathsf{s}} \left(\prod_{j=0}^{l-1} [w_j] \right),$$

where $\mathsf{s} = \lfloor ((w_0 - 1) + \cdots + (w_{l-1} - 1))/2 \rfloor$. The signature contains γ and the chain values at positions x_i. The verifier recomputes d, rejects unless the same predicate holds, and then performs the remaining $w_i - 1 - x_i$ chain steps. We use $\Phi = \mathsf{WOTS}{+}\mathsf{C}[\mathsf{H}_{\mathsf{Root}}, \mathbb{F}_2^{8n}, z_b, \mathbf{w}]$ with $\mathbf{w} = (w_0, \cdots, w_{l-1})$ to denote an inhomogeneous $\texttt{WOTS}^{\mathsf{C}}$ scheme. Unlike the deterministic encodings above, this preprocessing is not an injective map from messages to codewords. Its security is therefore covered by Theorem 6, rather than by Construction 1 alone. A finite counter-search limit can cause signing failure; this is a correctness and performance event and must be bounded separately from unforgeability. Let $\mathfrak{C}_{\mathsf{Sign}}(\Phi)$, $\mathfrak{C}_{\mathsf{Ver}}(\Phi)$, and $\mathfrak{C}_{\mathsf{Size}}(\Phi)$ denote the signing cost, verification cost, and signature size of Φ, respectively. Then, we have the following estimation

$$\begin{cases} \mathfrak{C}_{\mathsf{Size}}(\Phi) = l \cdot n + 4 \quad \text{(4 bytes for storing the counter)} \\ \mathfrak{C}_{\mathsf{Sign}}(\Phi) = \left\lfloor \frac{(w_0 - 1) + \cdots + (w_{l-1} - 1)}{2} \right\rfloor + l + \frac{2^{z_b} \cdot \prod_{j=0}^{l-1} w_j}{|\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1} [w_j])|} \\ \mathfrak{C}_{\mathsf{Ver}}(\Phi) = \left\lceil \frac{(w_0 - 1) + \cdots + (w_{l-1} - 1)}{2} \right\rceil \end{cases}, \tag{6}$$

where $s = \left\lfloor \frac{\sum_{i=0}^{l-1} (w_i - 1)}{2} \right\rfloor$, the unit for $\mathfrak{C}_{\mathsf{Sign}}(\Psi)$ and $\mathfrak{C}_{\mathsf{Ver}}(\Psi)$ is the number of hash function evaluations and the unit for $\mathfrak{C}_{\mathsf{Size}}(\Psi)$ is the number of bytes.

Searching Strategy. First of all, we assume that n is given. Our search needs to determine z_b, l and $(w_0, \cdots, w_{l-1})$. In the following, we give some lemmas and propositions based on which our search strategy is developed.

Lemma 11. *Let A and l be positive integers. Then*

$$\max_{\substack{(\log_2 w_0,\cdots,\log_2 w_{l-1})\in\mathbb{Z}_+^l \\ \log_2 w_0+\cdots+\log_2 w_{l-1}=A}} \left|\mathcal{C}_{\left\lfloor\frac{\sum_{i=0}^{l-1}(w_i-1)}{2}\right\rfloor}\left(\prod_{j=0}^{l-1}[w_j]\right)\right| = \left|\mathcal{C}_{\left\lfloor\frac{\sum_{i=0}^{l-1}(\hat{w}_i-1)}{2}\right\rfloor}\left(\prod_{j=0}^{l-1}[\hat{w}_j]\right)\right|,$$

where $\log_2 \hat{w}_0 = \cdots = \log_2 \hat{w}_{l-k-1} = \left\lfloor\frac{A}{l}\right\rfloor$, $\log_2 \hat{w}_{l-k} = \cdots = \log_2 \hat{w}_{l-1} = \left\lceil\frac{A}{l}\right\rceil$, *and* $k = A - l\cdot\left\lfloor\frac{A}{l}\right\rfloor$.

Proof. See Section M in Supplementary Material. □

Proposition 3. *Let $(w_0,\cdots,w_{l-1})$ and $(u_0,\cdots,u_{l-1})$ be two any given vectors in $\mathbb{Z}_+^l$ and z_b be any given integer in $\mathbb{N}$, such that $8n - z_b = \log_2 w_0 + \cdots + \log_2 w_{l-1} = \log_2 u_0 + \cdots + \log_2 u_{l-1}$. Let $\Phi = \mathsf{WOTS{+}C}[\mathsf{H}_{\mathsf{Root}}, \mathbb{F}_2^{8n}, z_b, (w_0,\cdots,w_{l-1})]$ be a one-time signature scheme. Let $\hat{\Phi} = \mathsf{WOTS{+}C}[\mathsf{H}_{\mathsf{Root}}, \mathbb{F}_2^{8n}, z_b, (u_0,\cdots,u_{l-1})]$ be another one-time signature scheme with $\log_2 u_0 = \cdots = \log_2 u_{l-k-1} = \left\lfloor\frac{8n-z_b}{l}\right\rfloor$, $\log_2 u_{l-k} = \cdots = \log_2 u_{l-1} = \left\lceil\frac{8n-z_b}{l}\right\rceil$, and $k = 8n - z_b - l\cdot\left\lfloor\frac{8n-z_b}{l}\right\rfloor$. Then $\mathfrak{C}_{\mathsf{Size}}(\hat{\Phi}) \le \mathfrak{C}_{\mathsf{Size}}(\Phi)$, $\mathfrak{C}_{\mathsf{Sign}}(\hat{\Phi}) \le \mathfrak{C}_{\mathsf{Sign}}(\Phi)$, and $\mathfrak{C}_{\mathsf{Ver}}(\hat{\Phi}) \le \mathfrak{C}_{\mathsf{Ver}}(\Phi)$.*

Proof. Obviously, $\mathfrak{C}_{\mathsf{Size}}(\hat{\Phi}) = \mathfrak{C}_{\mathsf{Size}}(\Phi) = l\cdot n + 4$. Since $\log_2 w_0 + \cdots + \log_2 w_{l-1} = 8n - z_b = \log_2 u_0 + \cdots + \log_2 u_{l-1}$, $2^{z_b}\cdot\prod_{j=0}^{l-1} w_j = 2^{z_b}\cdot\prod_{j=0}^{l-1} u_j$. According to Lemma 11, we have

$$\left|\mathcal{C}_{\left\lfloor\frac{\sum_{i=0}^{l-1}(w_i-1)}{2}\right\rfloor}\left(\prod_{j=0}^{l-1}[w_j]\right)\right| \le \left|\mathcal{C}_{\left\lfloor\frac{\sum_{i=0}^{l-1}(u_i-1)}{2}\right\rfloor}\left(\prod_{j=0}^{l-1}[u_j]\right)\right|.$$

According to Lemma 4, we have $\sum_{j=0}^{l-1} w_j \ge \sum_{j=0}^{l-1} u_j$ and thus

$$\left\lfloor\frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2}\right\rfloor + l \ge \left\lfloor\frac{(u_0-1)+\cdots+(u_{l-1}-1)}{2}\right\rfloor + l.$$

Consequently, $\mathfrak{C}_{\mathsf{Sign}}(\hat{\Phi}) \le \mathfrak{C}_{\mathsf{Sign}}(\Phi)$, and $\mathfrak{C}_{\mathsf{Ver}}(\hat{\Phi}) \le \mathfrak{C}_{\mathsf{Ver}}(\Phi)$ by Equation (6). □

Proposition 3 shows that for a given value of (n, l, z_b), we should always choose $(u_0,\cdots,u_{l-1})$ to be the value of $(w_0,\cdots,w_{l-1})$. Since there is no hope for other choices to outperform $(u_0,\cdots,u_{l-1})$.

Based on the above lemma and proposition, we come up with the following search procedure. First, we assume that for $0 \le i < l$, $4 \le w_i \le 256$, and $0 \le z_b \le 14$. Then, we have $\left\lceil\frac{8n-z_b}{\log_2 256}\right\rceil \le l \le \left\lfloor\frac{8n-z_b}{\log_2 4}\right\rfloor$. For each l and z_b, we can derive the optimal value of $(w_0,\cdots,w_{l-1})$ based on Proposition 3. With this procedure, we identify several inhomogeneous $\texttt{WOTS}^{\texttt{C}}$ schemes outperforming the regular $\texttt{WOTS}^{\texttt{C}}$ schemes used in the $\texttt{SPHINCS}^{+}\texttt{C}$ algorithms specified in [HKRY23] in signing cost with the same signature size and slightly increased verification cost. Some representative results are listed in Table 6 and more trade-offs can be found in Table 16, Table 17 and Table 18 in Supplementary Material.

Table 6: Representative regular and inhomogeneous $\texttt{WOTS}^{\texttt{C}}$ schemes, where the signature size is in bytes, and signing and verification are in terms of the number of hash function calls.

BitSec	Regular $\texttt{WOTS}^{\texttt{C}}$					Inhomogeneous $\texttt{WOTS}^{\texttt{C}}$				
	$\mathbf{w}$	z_b	Signature Size	Signing	Verification	$\mathbf{w}$	z_b	Signature Size	Signing	Verification
128	$[16]^{32}$	0	516	337	240	$[16]^{27} \times [32]^{4}$	0	500	371	265
	$[128]^{18}$	2	292	2746	1143	$[128]^{8} \times [256]^{9}$	0	276	2293	1656
192	$[16]^{48}$	0	1156	488	360	$[16]^{43} \times [32]^{4}$	0	1132	520	385
	$[128]^{27}$	3	652	5612	1715	$[128]^{16} \times [256]^{10}$	0	628	3016	2291
256	$[16]^{64}$	0	2052	636	480	$[16]^{59} \times [32]^{4}$	0	2020	667	505
	$[64]^{42}$	4	1348	6183	1323	$[64]^{31} \times [128]^{10}$	0	1316	2044	1612

Finally, to deepen our understanding of the trade-off curve, we show that reducing the signature size of $\texttt{WOTS}^{\texttt{C}}$ without changing z_b always decreases the performance of signing and verification by Lemma 12.

Lemma 12. *Let A and l,l' be positive integers with $l > l'$. Then*

$$\left| \mathcal{C}_{\left\lfloor \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rfloor} \left(\prod_{j=0}^{l-1} [w_j] \right) \right| > \left| \mathcal{C}_{\left\lfloor \frac{\sum_{i=0}^{l'-1}(w'_i-1)}{2} \right\rfloor} \left(\prod_{j=0}^{l'-1} [w'_j] \right) \right|,$$

where $\log_2 w_0 = \cdots = \log_2 w_{l-k-1} = \left\lfloor \frac{A}{l} \right\rfloor$, $\log_2 w_{l-k} = \cdots = \log_2 w_{l-1} = \left\lceil \frac{A}{l} \right\rceil$, $k = A - l \cdot \left\lfloor \frac{A}{l} \right\rfloor$, and $\log_2 w'_0 = \cdots = \log_2 w'_{l'-k'-1} = \left\lfloor \frac{A}{l'} \right\rfloor$, $\log_2 w'_{l'-k'} = \cdots = \log_2 w'_{l'-1} = \left\lceil \frac{A}{l'} \right\rceil$, $k' = A - l' \cdot \left\lfloor \frac{A}{l'} \right\rfloor$. Note that under these assignments, $\log_2 w_0 + \cdots + \log_2 w_{l-1} = \log_2 w'_0 + \cdots + \log_2 w'_{l'-1} = A$.

Proof. See Section N in Supplementary Material. □

Proposition 4. *Let l, l' be two positive integers with $l > l'$, $(w_0, \cdots, w_{l-1})$ be any given vector in $\mathbb{Z}_+^l$, $(w'_0, \cdots, w'_{l'-1})$ be any given vector in $\mathbb{Z}_+^{l'}$ and z_b be any given integer in $\mathbb{N}$, such that $8n - z_b = \log_2 w_0 + \cdots + \log_2 w_{l-1}$ and $8n - z_b = \log_2 w'_0 + \cdots + \log_2 w'_{l'-1}$ is fulfilled. Let*

$$\Phi = \mathsf{WOTS{+}C}[\mathsf{H}_{\mathsf{Root}}, \mathbb{F}_2^{8n}, z_b, (w_0, \cdots, w_{l-1})]$$

be a one-time signature scheme with $\log_2 w_0 = \cdots = \log_2 w_{l-k-1} = \left\lfloor \frac{8n-z_b}{l} \right\rfloor$, $\log_2 w_{l-k} = \cdots = \log_2 w_{l-1} = \left\lceil \frac{8n-z_b}{l} \right\rceil$, and $k = 8n - z_b - l \cdot \left\lfloor \frac{8n-z_b}{l} \right\rfloor$. Let

$$\hat{\Phi} = \mathsf{WOTS{+}C}[\mathsf{H}_{\mathsf{Root}}, \mathbb{F}_2^{8n}, z_b, (w'_0, \cdots, w'_{l'-1})]$$

be a one-time signature scheme with $\log_2 w'_0 = \cdots = \log_2 w'_{l'-k'-1} = \left\lfloor \frac{8n-z_b}{l'} \right\rfloor$, $\log_2 w'_{l'-k'} = \cdots = \log_2 w'_{l'-1} = \left\lceil \frac{8n-z_b}{l'} \right\rceil$, and $k' = 8n - z_b - l' \cdot \left\lfloor \frac{8n-z_b}{l'} \right\rfloor$. Then $\mathfrak{C}_{\mathsf{Size}}(\hat{\Phi}) < \mathfrak{C}_{\mathsf{Size}}(\Phi)$, $\mathfrak{C}_{\mathsf{Sign}}(\hat{\Phi}) > \mathfrak{C}_{\mathsf{Sign}}(\Phi)$, and $\mathfrak{C}_{\mathsf{Ver}}(\hat{\Phi}) > \mathfrak{C}_{\mathsf{Ver}}(\Phi)$.

Proof. Similar to the proof of Proposition 3. □

4 Extending the SPHINCS$^+$ Hypertree Structure

The SPHINCS$^+$ framework [BHK+19] organizes a large number of Merkle trees into layers, and the resulting virtual structure is called a hypertree. In the top layer there is a single Merkle tree, while the number of Merkle trees in a lower layer is equal to the number of leaves of all the Merkle trees in the layer right above. Each leaf of the Merkle trees corresponds to an instances of WOTS$^+$ one time signature. In addition, each leaf of the trees in the lowest layer (layer 0) is associated with a few-time signature instance called FORS.

When signing a message msg, one part of the digest of msg is used to identify a leaf ϕ_{msg} in the lowest layer. Then, the FORS instance associated to ϕ_{msg} is used to sign another part of the digest of msg. The public key of the FORS instance is signed by a WOTS$^+$ instance corresponding to ϕ_{msg}. The root of the tree to which ϕ_{msg} belongs is then signed by a leaf (an WOTS$^+$ instance) of the tree in the layer right above. This continues until we reach the root of the tree in the top layer. The SPHINCS$^+$ signature includes all of the resulting FORS and WOTS$^+$ signatures and the authentication paths of the Merkle trees required for the verifier to compute the root of the hypertree. In this work, we use $\Psi[n, h, d, k, a, l_0, l_1, (w_0, \cdots, w_{l_0+l_1-1})]$ to denote the scheme instantiating the extended SPHINCS$^+$ framework, where n is the size (in bytes) of the nodes of the trees and hash chains, $h = h_0 + \cdots + h_{d-1}$ is the total height with h_i being the height of the trees in layer $i \in \{0, \cdots, d-1\}$, d is the number of layers, each FORS instance in the extended hypertree contains k subtrees with $t = 2^a$ leaves, and the one-time signatures employed are instances of inhomogeneous WOTS$^+$ one-time signature $\mathsf{OTS}[\mathsf{H}, \mathtt{Encode}, \mathbb{F}_2^{8n}, (w_0, \cdots, w_{l_0+l_1-1})]$. If we allow $\Psi[n, h, d, k, a, l_0, l_1, (w_0, \cdots, w_{l_0+l_1-1})]$ to sign at most N messages, the structural forgery probability used in our parameter search is estimated by $-\log_2 \lambda(h, k, a)$-bit[NIS24b], where

$$\lambda(h,k,a) = \frac{1}{2^{8n}} + \sum_{i=0}^{N} \left(\frac{1}{2^{ih}} \binom{N}{i} \left(1 - \frac{1}{2^h}\right)^{N-i} \left(1 - \left(1 - \frac{1}{2^a}\right)^i\right)^k \right).$$

The full reduction is stated in Theorem 7 (Section R in Supplementary Material). In particular, the total number of inhomogeneous-WOTS instances is less than 2^{h+1} for every positive partition $h = h_0 + \cdots + h_{d-1}$. The numbers of Merkle internal nodes and FORS instances remain exactly $2^h - 1$ and 2^h, respectively. Thus, varying the layer heights preserves the game sequence of the SPHINCS$^+$ proof while changing only the exact multi-target counts.

For simplicity, we present the extended hypertree structure with FORS and inhomogeneous WOTS$^+$, and we name the resulting framework as CEDRUS$^+$ (to emphasize that it is a natural generalization of SPHINCS$^+$). Similarly, one can use inhomogeneous WOTS$^\alpha$ and FORC in the extended hypertree, and the resulting framework is called CEDRUS$^\alpha$ (to emphasize that it is a natural generalization of SPHINCS-α). Moreover, one can use inhomogeneous WOTS$^{\mathsf{C}}$ and FORS+C in the extended hypertree, and the resulting framework is named as CEDRUS$^+$C (to

emphasize that it is a natural generalization of SPHINCS$^+$C). In this section, we focus on CEDRUS$^+$. The security arguments for CEDRUS$^\alpha$ and CEDRUS$^+$C are not obtained by a verbatim substitution: the former uses the FORC-specific ITSR experiment, whereas the latter adds the S-TCR($\mathsf{Good}_{\mathbf{w},z_b}$) term of WOTS$^{\mathsf{C}}$; see Section R.3. Let $\Psi[n,h,d,k,a,l_0,l_1,(w_0,\cdots,w_{l_0+l_1-1})]$ be a CEDRUS$^+$ scheme. Let $\mathfrak{C}_{\mathsf{Sign}}(\Psi)$, $\mathfrak{C}_{\mathsf{Ver}}(\Psi)$, and $\mathfrak{C}_{\mathsf{Size}}(\Psi)$ denotes the signing cost, verification cost, and signature size of Ψ, respectively. Then, we have the following estimation

$$\begin{cases} \mathfrak{C}_{\mathsf{Size}}(\Psi) = (a+1)\cdot k + h + (l_0+l_1)\cdot d + 1 \\ \mathfrak{C}_{\mathsf{Sign}}(\Psi) = 3k\cdot 2^a - k + 1 + \left\langle \frac{h}{d} \right\rangle \cdot (2 + \sum_{i=0}^{(l_0+l_1)-1} w_i) - d \\ \mathfrak{C}_{\mathsf{Ver}}(\Psi) = k\cdot(a+1) + 1 + d\cdot(1 + \sum_{i=0}^{(l_0+l_1)-1} w_i - (l_0+l_1)) + h \end{cases}, \quad (7)$$

where the unit for $\mathfrak{C}_{\mathsf{Sign}}(\Psi)$ and $\mathfrak{C}_{\mathsf{Ver}}(\Psi)$ is the number of hash function evaluations and the unit for $\mathfrak{C}_{\mathsf{Size}}(\Psi)$ is the number of bytes.

Remark 2. In existing hash-based signature schemes, $h_0 = \cdots = h_{d-1}$ and $w_0 = \cdots = w_{l-1}$, while we allow trees in different layers to be of different heights (but trees in the same layer are of the same height). However, we do not specify $(h_0,\cdots,h_{d-1})$ in the parameter array $[n,h,d,k,a,l_0,l_1,(w_0,\cdots,w_{l_0+l_1-1})]$. This is because given the value of (n,h,d,k,a,l_0,l_1), according to Lemma 2 and Equation (7), there is in essence a unique optimal value for $(h_0,\cdots,h_{d-1})$.

Searching Strategy. First of all, we assume that n is given. Our search needs to determine $[h,d,k,a,l_0,l_1,(w_0,\cdots,w_{l_0+l_1-1})]$ and $(h_0,\cdots,h_{d-1})$ with $h_0+\cdots+h_{d-1}=h$. In the following, we give some lemmas based on which our search strategy is developed.

Lemma 13. *Let h, h' and d be positive integers with $h<h'$. Then, $\langle \frac{h}{d} \rangle < \langle \frac{h'}{d} \rangle$.*

Proof. See Section O in Supplementary Material. □

Lemma 13 shows that for two CEDRUS$^+$ schemes $\Psi[n,h,d,k,a,l_0,l_1,(w_0,\cdots,w_{l_0+l_1-1})]$ and $\Psi[n,h',d,k,a,l_0,l_1,(w_0,\cdots,w_{l_0+l_1-1})]$, we prefer the one with smaller total height. However, the total height cannot be too small due to the security constraint $-\log_2 \lambda(h,k,a) \geq 8n$.

Lemma 14. *Let h, d and d' be positive integers with $d<d'$. Then, $\langle \frac{h}{d} \rangle \geq \langle \frac{h}{d'} \rangle$.*

Proof. See Section P in Supplementary Material. □

Equation (7) and Lemma 14 shows that less layers leads to less efficient signers. So d cannot be too small if we would like to enforce a upper bound on the signing cost. At the same time, according to Equation (7), d cannot be too large if we would like to enforce upper bound on the signature size.

Based on the above lemmas, we come up with the searching strategy presented in Algorithm 3. In Algorithm 3, we assume that for $0 \leq i < l_0+l_1$, $4 \leq w_i \leq 256$. Then, we can use the searching strategy given in Section 3.1 to determine the candidates for $(l_0,l_1,(w_0,\cdots,w_{l_0+l_1-1}))$. Therefore,

the candidates mentioned in Line 1 of Algorithm 3 is well-defined. For each $(l_0, l_1, (w_0, \cdots, w_{l_0+l_1-1}))$ candidate, we enumerate h and a valuations. Then, we determine the smallest possible value of k fulfilling the security constraint given by Line 5. Afterwards, Line 6 enumerates the candidate values for d. Note that we break the search with Line 8 to stop checking any larger values of k, since Equation (7) indicates that larger k leads to inferior CEDRUS$^+$ schemes.

Algorithm 3: Exploring the design space of CEDRUS$^+$

Input: n, bounds for signing cost and signature size
Output: Candidate parameter sets for CEDRUS$^+$

```
1 for each candidate (l_0, l_1, (w_0, ··· , w_{l_0+l_1−1})) do
2     for h ∈ {56, 57, ··· , 75} do
3         for a ∈ {3, 4, ··· , 23} do
4             for k ∈ {1, 2, ··· , 63} do
5                 if −log_2 λ(h, k, a) ≥ 8n then
6                     for 3 ≤ d ≤ h − 1 do
7                         Check whether Ψ[n, h, d, a, t = 2^a, k, ℓ, (w_0, ··· , w_{ℓ−1})]
                            fulfills a specific design goal
8                 Break (the for-loop for h)
```

5 Applications and Implementations

In this Section, we apply the extended framework to SPHINCS$^+$, SPHINCS-α, and SPHINCS$^+$C, employing the inhomogeneous variants of their respective one-time-signature constructions. We explore the design spaces with the searching strategies given in Section 3 and Section 4. Some parameter sets leading to improved performance are reported. We implement several signature schemes instantiated with representative parameter sets by adapting the respective official implementations to our settings, where we reuse most of their basic modules whenever possible. We provide theoretical performance analysis and benchmark results obtained from running our codes compiled with `gcc-11.4.0-O3-march=native-fomit-frame-pointer -flto` on a `Ubuntu 22.04` machine with `Intel Core i5-1135G7` CPU and 16GB RAM. In addition, all cycle counts are the median of 10^3 runs. The source codes are submitted with the paper. To ensure a fair comparison, we provide both an AVX2-optimized and a non-optimized implementation of CEDRUS$^+$C, as the original implementations from [HKRY23] did not utilize the AVX2 instructions.

5.1 Extending and Improving SPHINCS$^+$

We search for parameter settings for the extended framework with the inhomogeneous WOTS$^+$ one-time signatures and FORS few-time signatures. The re-

sulting schemes are named as CEDRUS$^+$. Some representative parameter sets are listed in Table 7. Their security follows from the complete reduction in Theorem 7: the SPHINCS$^+$ games are retained, the inhomogeneous-WOTS transition uses W_Σ and $w_{\max}$, and unequal layer heights affect only the multi-target counts. We make some theoretical and experimental comparisons in Table 8 and Table 9. The NIST stateless hash-based digital signature standard specified in FIPS 205 [NIS24b] includes a fast (but larger) variant and a small (but slower) variant for each security level. For each variant, we provide a corresponding CEDRUS$^+$ variant that achieves smaller signature size while maintaining or improving the signing and verification speed. For example, CEDRUS$^+$-128f given in Table 7 is 560 bytes smaller than SPHINCS$^+$-128f. At the same time, it makes 406 less hash function calls in signing and 6469 less hash function calls in verification than SPHINCS$^+$-128f. The users may want to achieve different trade-offs depending on the use cases. For example, the CEDRUS$^+$ with ID = 0x01 (named as CEDRUS$^+$[0x01]) is 1472 bytes smaller than SPHINCS$^+$-128f, and it makes 6014 less hash function calls in verification with slightly increased signing cost. The CEDRUS$^+$ with ID = 0x02 (named as CEDRUS$^+$[0x02]) is 4992 bytes smaller than SPHINCS$^+$-128f, and it makes 3719 less hash function calls in verification with increased signing cost. Such variant may be employed in secure software updates where signing speed is not quite sensitive. In summary, our extension offers more comprehensive trade-offs, and Table 7 only lists of a few of them.

5.2 Extending and Improving SPHINCS-α

We search for parameter settings for the extended framework with the inhomogeneous WOTS$^\alpha$ one-time signatures and FORC few-time signatures used in SPHINCS-α [ZCY22]. The resulting schemes are named as CEDRUS$^\alpha$. Some representative parameter sets are listed in Table 20 (Section T in Supplementary Material). The security argument combines the corrected SPHINCS$^+$ game sequence of [HK22] with the FORC-specific ITSR experiment of [ZCY22]. It substitutes the inhomogeneous-WOTS$^\alpha$ chain bounds and unequal-layer target counts while retaining the FORC coverage relation and chain-length parameter; see Section R.3 in Supplementary Material. We make some theoretical and experimental comparisons in Table 21 (Section T in Supplementary Material) and Table 22 (Section T in Supplementary Material). In [ZCY23], the designers provide a fast (but larger) variant and a small (but slower) variant for each security level. For each variant, we provide a corresponding CEDRUS$^\alpha$ variant that achieves smaller signature size while maintaining or improving the signing and verification speed. For example, CEDRUS$^\alpha$-128f given in Table 20 is 480 bytes smaller than SPHINCS-α-128f. At the same time, it makes 105 less hash function calls in signing and 487 less hash function calls in verification than SPHINCS-α-128f. The users may want to achieve different trade-offs depending on the use cases. For example, the CEDRUS$^\alpha$ with ID = 0x01 (named as CEDRUS$^\alpha$[0x01]) is 1216 bytes smaller than SPHINCS-α-128f, and it makes 2088 less hash function calls in verification with slightly increased signing cost. The CEDRUS$^\alpha$ with ID = 0x02 (named as CEDRUS$^\alpha$[0x02]) is 5040 bytes smaller than SPHINCS-α-128f, and it makes 115

Table 7: Representative parameter sets for CEDRUS$^+$

ID	Name	n	h	d	k	a	l_0	l_1	$\mathbf{w}$	Bitsec
–	SPHINCS$^+$-128f	16	66	22	33	6	32	3	$[16]^{32} \times [16]^3$	128
0x00	CEDRUS$^+$-128f	16	64	16	29	7	43	3	$[4] \times [8]^{42} \times [8]^3$	128
0x01	–	16	64	16	23	8	41	3	$[8]^{36} \times [16]^5 \times [8]^3$	129
0x02	–	16	65	15	18	9	32	2	$[16]^{32} \times [16] \times [32]$	129
–	SPHINCS$^+$-128s	16	63	7	14	12	32	3	$[16]^{32} \times [16]^3$	133
0x03	CEDRUS$^+$-128s	16	62	7	13	13	32	3	$[16]^{32} \times [8]^3$	130
0x04	–	16	64	7	13	12	32	3	$[16]^{32} \times [8]^3$	131
0x05	–	16	66	7	9	15	32	2	$[16]^{32} \times [16] \times [32]$	128
–	SPHINCS$^+$-192f	24	66	22	33	8	48	3	$[16]^{48} \times [16]^3$	194
0x06	CEDRUS$^+$-192f	24	68	17	37	7	62	3	$[8]^{56} \times [16]^6 \times [8]^3$	195
0x07	–	24	64	16	38	8	61	3	$[8]^{52} \times [16]^9 \times [8]^3$	196
0x08	–	24	66	15	27	9	47	2	$[16]^{43} \times [32]^4 \times [32]^2$	192
–	SPHINCS$^+$-192s	24	63	7	17	14	48	3	$[16]^{48} \times [16]^3$	193
0x09	CEDRUS$^+$-192s	24	64	7	18	13	48	3	$[16]^{48} \times [8]^2 \times [16]$	194
0x0A	–	24	65	7	19	12	48	3	$[16]^{48} \times [8]^2 \times [16]$	192
0x0B	–	24	66	7	13	16	48	3	$[16]^{48} \times [8]^2 \times [16]$	193
–	SPHINCS$^+$-256f	32	68	17	35	9	64	3	$[16]^{64} \times [16]^3$	255
0x0C	CEDRUS$^+$-256f	32	64	16	43	9	64	2	$[16]^{64} \times [32]^2$	259
0x0D	–	32	65	16	40	9	64	2	$[16]^{64} \times [32]^2$	256
0x0E	–	32	65	13	34	10	58	3	$[16]^{34} \times [32]^{24} \times [8] \times [16]^2$	257
–	SPHINCS$^+$-256s	32	64	8	22	14	64	3	$[16]^{64} \times [16]^3$	255
0x0F	CEDRUS$^+$-256s	32	66	8	23	13	64	3	$[16]^{64} \times [8]^2 \times [16]$	259
0x10	–	32	68	8	24	12	64	3	$[16]^{64} \times [8]^2 \times [16]$	257
0x11	–	32	64	7	22	14	61	3	$[16]^{49} \times [32]^{12} \times [8] \times [16]^2$	256

less hash function calls in verification with 1.93 times increased signing cost. Such variant may be employed in secure software updates where signing speed is not quite sensitive. In summary, our extension offers more comprehensive trade-offs, and Table 20 only lists of a few of them.

5.3 Extending and Improving SPHINCS$^+$C

We search for parameter settings for the extended framework with the inhomogeneous WOTS^C one-time signatures and FORS+C few-time signatures used in SPHINCS$^+$C [HKRY23]. The resulting schemes are named as CEDRUS$^+$C. Some representative parameter sets are listed in Table 23 (Section U in Supplementary Material). Its reduction uses the additional S-TCR($\mathsf{Good}_{\mathbf{w},z_b}$) term required by the counter-based WOTS^C preprocessing. The FORS+C component remains inside the same ITSR step used by the SPHINCS$^+$C reduction and introduces no second counter-security term; see Section R.3 in Supplementary Material. We make some theoretical and experimental comparisons in Table 24 (Section U in Supplementary Material) Table 25 and Table 26 (Section U in Supplementary Material). Some

Table 8: A theoretical comparison of the signature schemes instantiated with the parameter sets listed in Table 7 with the SPHINCS$^+$ algorithms given in the NIST FIPS 205 standards [NIS24b]

Scheme	PK Size (Bytes)	SK Size (Bytes)	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
SPHINCS$^+$-128f	32	64	17088	105194	11870
CEDRUS$^+$-128f	32	64	16528	104788	5401
0x01	32	64	15616	118490	5856
0x02	32	64	12096	207456	8151
SPHINCS$^+$-128s	32	64	7856	2186220	3928
CEDRUS$^+$-128s	32	64	7840	2109933	3759
0x04	32	64	7664	2363373	3748
0x05	32	64	7184	3762161	3900
SPHINCS$^+$-192f	48	96	35664	169258	17216
CEDRUS$^+$-192f	48	96	35280	169195	8933
0x07	48	96	34344	179147	8743
0x08	48	96	25728	337783	12817
SPHINCS$^+$-192s	48	96	16224	3767273	5681
CEDRUS$^+$-192s	48	96	16176	3727336	5567
0x0A	48	96	16080	3929063	5563
0x0B	48	96	15480	6662125	5538
SPHINCS$^+$-256f	64	128	49856	345837	17521
CEDRUS$^+$-256f	64	128	49632	345030	16863
0x0D	64	128	48704	357865	16834
0x0E	64	128	39456	667666	17236
SPHINCS$^+$-256s	64	128	29792	3280867	8443
CEDRUS$^+$-256s	64	128	29600	3273698	8309
0x10	64	128	29344	3545057	8301
0x11	64	128	26976	6037476	8410

Table 9: Performance comparison between CEDRUS$^+$ and SPHINCS$^+$ simple variants, with tweakable hash functions instantiated with SHAKE-avx2. Key generation, signing and verification time are measured in 1000 CPU cycles. Key and signature sizes are in bytes.

	Key Generation		Signing		Verification		Size	
	SPHINCS$^+$	CEDRUS$^+$	SPHINCS$^+$	CEDRUS$^+$	SPHINCS$^+$	CEDRUS$^+$	SPHINCS$^+$	CEDRUS$^+$
SHAKE-128f	847.8	1119.8 (+32.1%)	20367.3	20308.4 (-0.3%)	1526.0	817.4 (-46.4%)	17088	16528 (-3.3%)
SHAKE-128s	55557.5	26350.6 (-52.6%)	422739.1	405516.1 (-4.1%)	608.4	565.5 (-7.1%)	7856	7840 (-0.2%)
SHAKE-192f	1254.9	1759.2 (+40.2%)	32963.6	33016.8 (+1.2%)	2150.8	1307.2 (-39.2%)	35664	35280 (-1.1%)
SHAKE-192s	81028.9	79192.8 (-2.3%)	727435.0	720957.7 (-1.0%)	862.3	832.5 (-3.5%)	16224	16176 (-0.3%)
SHAKE-256f	3331.1	3350.3 (+0.6%)	67099.8	66924.8 (-0.3%)	2232.2	2210.4 (-1.0%)	49856	49632 (-0.4%)
SHAKE-256s	53243.5	52540.6 (-1.3%)	636441.6	636352.1 (-0.0%)	1173.9	1141.0 (-2.8%)	29792	29600 (-0.6%)

experimental comparisons with the AVX2 instruction set are listed in Table 26 (Section U in Supplementary Material). The designers of [HKRY23] provide a fast (but larger) variant and a small (but slower) variant for each security level.

For each variant, we provide a corresponding CEDRUS$^+$C variant that achieves smaller signature size while maintaining or improving the signing and verification speed. For example, CEDRUS$^+$C-128f given in Table 23 is 564 bytes smaller than SPHINCS$^+$C-128f. At the same time, it makes 154 less hash function calls in signing and 323 less hash function calls in verification than SPHINCS$^+$C-128f. The users may want to achieve different trade-offs depending on the use cases. For example, the CEDRUS$^+$C with ID = 0x01 (named as CEDRUS$^+$C[0x01]) is 1080 bytes smaller than SPHINCS$^+$C-128f, and it makes 559 less hash function calls in verification with slightly increased signing cost. The CEDRUS$^+$C with ID = 0x02 (named as CEDRUS$^+$C[0x02]) is 4400 bytes smaller than SPHINCS$^+$C-128f, and it makes 2041 less hash function calls in verification with 2 times increased signing cost. Such variant may be employed in secure software updates where signing speed is not quite sensitive. In summary, our extension offers more comprehensive trade-offs, and Table 23 only lists a few of them.

5.4 Synergistic Optimization with PORS+FP

The PORS+FP [AK25] effectively reduce signature size by forced pruning. We observe that applying the PORS+FP technique to our CEDRUS$^+$ and CEDRUS$^+$C frameworks leads to greater efficiency gains than applying it to the baseline SPHINCS$^+$ and SPHINCS$^+$C. The resulting schemes are named as CEDRUS$^+$FP and CEDRUS$^+$C-FP. Similarly, we abbreviate the original algorithms as SPHINCS$^+$FP and SPHINCS$^+$C-FP. The parameter sets of CEDRUS$^+$FP are listed in Table 27 (Section V in Supplementary Material). We make some theoretical comparisons in Table 28 (Section V in Supplementary Material) and theoretical comparisons with percentage in Table 29 (Section V in Supplementary Material). The parameter sets of CEDRUS$^+$C-FP are listed in Table 30 (Section V in Supplementary Material), and theoretical comparisons are listed in Table 31 (Section V in Supplementary Material) and Table 32 (Section V in Supplementary Material). Note that the SPHINCS$^+$C-FP instances provided in [AK25] for 128f and 192f exhibit security levels of 125 bits and 189 bits rather than 128 and 192 bits (We have confirmed this with the authors), respectively, falling slightly short of the required 128-bit and 192-bit targets. Consequently, to fully meet these security requirements, we anticipate a tiny increase in signature size, as well as in signing and verification costs.

5.5 CEDRUS Supporting Less than 2^{64} Signatures

Recently, the Smaller SPHINCS$^+$ [FD24] has attracted significant attention within the cryptographic community. This variant introduces a refined parameter selection that deliberately sacrifices the total number of available signatures to achieve superior performance. Under identical constraints on security strength, maximum signing cost, and signing capacity, our CEDRUS$^+$ method offers flexible optimization strategies. Specifically, if we maintain an identical overuse safety to the baseline, we can identify parameter sets with strictly smaller signature sizes and reduced verification costs, e.g., compared with the A-1, E-2, and N-8

parameter sets in [FD24]. Conversely, we can also find parameters that achieve higher overuse safety, e.g., compared with the I-2, P-2, and Q-1 parameter sets in [FD24], while keeping the signature size and verification cost approximate to the original values. We present several representative results in Table 33 (Section W in Supplementary Material).

The application of SPHINCS$^+$FP to the Smaller SPHINCS$^+$ framework yields significant improvements, optimizing both signature size and verification cost with only a marginal penalty in signing overhead. However, our proposed CEDRUS$^+$FP identifies new parameter sets that achieve superior performance compared to the optimizations presented in Appendix C of [AK25]. Some results are presented in Table 34 (Section W in Supplementary Material).

There are three specific parameter sets (rlsxxxcs1) constrained by a maximum signing capacity of 2^{24} in [Dan25], which satisfy the 128-bit, 192-bit, and 256-bit security levels, respectively. The application of SPHINCS$^+$FP to them also yields improvements (Table 5 of [AK25]). We observed that alternative configurations were available but likely discarded due to excessive signing costs. CEDRUS$^+$FP addresses this limitation by reducing the signing cost to a level we consider acceptable - below that of SPHINCS$^+$FP, thereby lowering the verification cost penalty required to reduce signature size. This unlocks new possibilities for parameter selection. For example, we can accept a practically negligible increase in verification time in exchange for further signature compression. For instance, with the CEDRUS$^+$FP-rls256cs1-1 parameter set, we observe a 15.3% increase in verification cost. However, this is accompanied by a 10.5% reduction in signature size and a 9.8% decrease in signing cost. Crucially, in absolute terms, the reduction in hash function calls during the signing process is 2,638,942 times larger than the increase in hash calls required for verification. Some representative parameter sets are listed in Table 35 (Section W in Supplementary Material). We make some theoretical comparisons in Table 36 (Section W in Supplementary Material). To the best of our knowledge, we are the first to reduce the signature size to below 3000 bytes while supporting a signing capacity of 2^{24}, without increasing the signing cost compared to the original scheme.

6 Conclusion

We generalize the one-time signature constructions based on hash chains by allowing the hash chains to be of different lengths. Moreover, we extend the hypertree structure that underlies the SPHINCS$^+$ framework by allowing trees of different heights to appear on different layers of the hypertree. These generalizations lead to enlarged design space and finer-grained trade-offs. By performing a systematic search guided by a thorough theoretical cost analysis, we identify new stateless hash-based signature schemes that exceed SPHINCS$^+$, SPHINCS-α, SPHINCS$^+$C, SPHINCS$^+$FP, and smaller SPHINCS$^+$ variants in terms of signature size, signing and verification efficiency. One can apply our ideas to stateful hash-based signatures like XMSS and we expect the relative performance gains to carry over. Our deterministic constructions are limited to injective, incomparable en-

codings, while $\texttt{WOTS}^{\texttt{C}}$ is the counter-based case considered here. It is interesting to see how the trade-off curve behaves for other randomized, non-uniform, and non-injective encodings [KKW25].

Acknowledgement. This work is supported by the National Cryptologic Science Fund of China (2025NCSF01012), the National Natural Science Foundation of China (62532013, 62672474), and the Fundamental Research Funds for the Central Universities and the Scientific Research Innovation Capability Support Project for Young Faculty.

References

AE17. Jean-Philippe Aumasson and Guillaume Endignoux. Clarifying the subset-resilience problem. IACR ePrint 2017/909, 2017. http://eprint.iacr.org/2017/909.

AE18. Jean-Philippe Aumasson and Guillaume Endignoux. Improving stateless hash-based signatures. In Nigel P. Smart, editor, *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference 2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings*, volume 10808 of *Lecture Notes in Computer Science*, pages 219–242. Springer, 2018.

AK25. Mehdi Abri and Jonathan Katz. Shorter hash-based signatures using forced pruning. Cryptology ePrint Archive, Paper 2025/2069, 2025.

BBB+20. Marco Baldi, Alessandro Barenghi, Michele Battagliola, Sebastian Bitzer, Marco Gianvecchio, Patrick Karl, Felice Manganiello, Alessio Pavoni, Gerardo Pelosi, Federico Pintore, Paolo Santini, Jonas Schupp, Edoardo Signorini, Freeman Slaughter, Antonia Wachter-Zeh, and Violetta Weger. CROSS: Codes and Restricted Objects Signature Scheme, 2020. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/cross-spec-round2-web.pdf.

BBB+25. Carsten Baum, Lennart Braun, Ward Beullens, Cyprien Delpech de Saint Guilhem Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST v2: Algorithm Specifications (Version 2.0), 2025. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/faest-spec-round2-web.pdf.

BDH11. Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. `XMSS` - A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions. In Bo-Yin Yang, editor, *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29 - December 2, 2011. Proceedings*, volume 7071 of *Lecture Notes in Computer Science*, pages 117–129. Springer, 2011.

Beu21. Ward Beullens. MAYO: Practical Post-quantum Signatures from Oil-and-Vinegar Maps. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*, volume 13203 of *Lecture Notes in Computer Science*, pages 355–376. Springer, 2021.

BHH+15. 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, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 368–397. Springer, 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, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2129–2146. ACM, 2019.

CDG+17. Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1825–1842. ACM, 2017.

CFS01. Nicolas T. Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to Achieve a McEliece-Based Digital Signature Scheme. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer, 2001.

CGH+96. Robert M Corless, Gaston H Gonnet, David EG Hare, David J Jeffrey, and Donald E Knuth. On the Lambert W Function. *Advances in Computational mathematics*, 5(1):329–359, 1996.

Dan25. Quynh Dang. Smaller SLH-DSA. NIST Computer Security Resource Center (CSRC) Presentation, 2025.

DLL+17. Léo Ducas, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - Dilithium: Digital Signatures from Module Lattices. *IACR Cryptol. ePrint Arch.*, page 633, 2017.

DLRW24. Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQIsignHD: New Dimensions in Cryptography. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2024.

FD24. Scott Fluhrer and Quynh Dang. Smaller sphincs+. Cryptology ePrint Archive, Paper 2024/018, 2024.

FKL+20. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact Post-quantum Signatures from Quaternions and Isogenies. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 64–93. Springer, 2020.

Gol86. Oded Goldreich. Two Remarks Concerning the Goldwasser-Micali-Rivest Signature Scheme. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 104–110. Springer, 1986.

HBG+18. Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, and Aziz Mohaisen. XMSS: eXtended Merkle Signature Scheme. RFC 8391, 2018. https://datatracker.ietf.org/doc/html/rfc8391.

HK22. Andreas Hülsing and Mikhail Kudinov. Recovering the tight security proof of SPHINCS+. In *Advances in Cryptology – ASIACRYPT 2022*, volume 13794 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2022.

HKRY23. Andreas Hülsing, Mikhail A. Kudinov, Eyal Ronen, and Eylon Yogev. SPHINCS+C: Compressing SPHINCS+ With (Almost) No Cost. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*, pages 1435–1453. IEEE, 2023.

KKW25. Dmitry Khovratovich, Mikhail A. Kudinov, and Benedikt Wagner. At the Top of the Hypercube - Better Size-Time Tradeoffs for Hash-Based Signatures. *IACR Cryptol. ePrint Arch.*, page 889, 2025.

KP22. Stefan Kölbl and Jade Philipoom. A note on SPHINCS+ parameter sets. Cryptology ePrint Archive, Paper 2022/1725, 2022.

Lam79. Leslie Lamport. Constructing Digital Signatures from a One Way Function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

MCF19. David McGrew, Michael Curcio, and Scott Fluhrer. Leighton-Micali Hash-Based Signatures. RFC 8554, 2019. https://datatracker.ietf.org/doc/html/rfc8554.

Mer79. Ralph Charles Merkle. Secrecy, authentication, and public key systems. Stanford University, 1979.

NIS24a. NIST. Module-Lattice-Based Digital Signature Standard, FIPS 204, 2024. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf.

NIS24b. NIST. Stateless Hash-Based Digital Signature Standard. Federal Information Processing Standards Publication, FIPS 205, 2024. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf.

PFH+20. Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON, 2020. https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

SEMR24. Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Krijn Reijnders. AprèsSQI: Extra Fast Verification for SQIsign Using Extension-Field Signing. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 63–93. Springer, 2024.

Sho99. Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.*, 41(2):303–332, 1999.

Ste93. Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California,*

USA, August 22-26, 1993, Proceedings, volume 773 of *Lecture Notes in Computer Science*, pages 13–21. Springer, 1993.

WBK+24. Thom Wiggers, Kaveh Bashiri, Stefan Kölbl, Jim Goodman, and Stavros Kousidis. Hash-based Signatures: State and Backup Management (Internet-Draft). IETF, Network Working Group, 2024. https://www.ietf.org/archive/id/draft-wiggers-hbs-state-00.html.

ZCY22. Kaiyi Zhang, Hongrui Cui, and Yu Yu. SPHINCS-α: A compact stateless hash-based signature scheme. Cryptology ePrint Archive, Paper 2022/059, 2022. https://eprint.iacr.org/2022/059.

ZCY23. Kaiyi Zhang, Hongrui Cui, and Yu Yu. Revisiting the Constant-Sum Winternitz One-Time Signature with Applications to SPHINCS$^+$ and XMSS. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part V*, volume 14085 of *Lecture Notes in Computer Science*, pages 455–483. Springer, 2023.

Supplementary Material

A Proof of Lemma 1

Proof. Suppose among these d positive integers, there are $d-k$ x_i's take on the value $a \geq 1$, and k x_i's take on the value $a+1$ where $0 \leq k < d$. Then we have

$$h = (d-k) \cdot a + k \cdot (a+1) = k + da.$$

Let $a = \lfloor \frac{h}{d} \rfloor + \epsilon$, where ϵ is an integer. If $\epsilon \geq 1$, then

$$h = k + d \cdot \left\lfloor \frac{h}{d} \right\rfloor + d\epsilon > k + d \cdot \left(\frac{h}{d} - 1 \right) + d\epsilon = h + k + d \cdot (\epsilon - 1) \geq h,$$

which is a contradiction. If $\epsilon \leq -1$, then

$$h = k + d \cdot \left\lfloor \frac{h}{d} \right\rfloor + d\epsilon \leq k + d \cdot \frac{h}{d} + d\epsilon = h + k + d \cdot \epsilon \leq h + k - d < h,$$

which is also a contradiction. Therefore, ϵ must be 0 and

$$\begin{cases} a = \lfloor \frac{h}{d} \rfloor \\ k = h - d \cdot \lfloor \frac{h}{d} \rfloor \end{cases}.$$

□

B Proof of Lemma 2

Proof. Since $f(x) = 2^x$ is a convex function, for any $(x_0, \cdots, x_{l-1}) \in \mathbb{R}^l$, and $(\lambda_0, \cdots, \lambda_{l-1}) \in \mathbb{R}^l_+$ with $\sum_{i=0}^{l-1} \lambda_i = 1$. According to Theorem 1, we have

$$2^{\sum_{i=0}^{l-1} \lambda_i x_i} \leq \sum_{i=0}^{l-1} \lambda_i 2^{x_i}.$$

Setting $\lambda_i = \frac{1}{l}$ and $\sum_{i=0}^{l-1} x_i = A$ gives $l \cdot 2^{\frac{A}{l}} \leq \sum_{i=0}^{l-1} 2^{x_i}$. Moreover,

$$\left\langle \frac{A}{l} \right\rangle = \min_{\substack{(x_0,\cdots,x_{l-1}) \in \mathbb{Z}^l_+ \\ x_0+\cdots+x_{l-1}=A}} \sum_{i=0}^{l-1} 2^{x_i}$$

implies $\left\langle \frac{A}{l} \right\rangle \geq l \cdot 2^{\frac{A}{l}}$. For any positive integers $x_0, \cdots, x_{l-1}$ with $x_0 + \cdots + x_{l-1} = A$, if there exist x_i, x_j where $0 \leq i, j < l$ such that $x_i \leq x_j - 2$, we set $x'_i = x_i + 1$ and $x'_j = x_j - 1$. Then, $2^{x_i} + 2^{x_j} - 2^{x'_i} - 2^{x'_j} = 2^{x_i} + 2^{x_j} - 2^{x_i+1} - 2^{x_j-1} = 2^{x_j-1} - 2^{x_i} \geq 2^{x_i+1} - 2^{x_i} = 2^{x_i} \geq 2$. That is, the above operation strictly decreases $\sum_{i=0}^{l-1} 2^{x_i}$ by at least 2. For any positive integers $x_0, \cdots, x_{l-1}$ with $x_0 + \cdots + x_{l-1} = A$, we repeatedly apply the above operation. This process will eventually terminates in at most

$$\frac{\sum_{i=0}^{l-1} 2^{x_i} - \left\langle \frac{A}{l} \right\rangle}{2} \leq \frac{\sum_{i=0}^{l-1} 2^{x_i} - l \cdot 2^{\frac{A}{l}}}{2}$$

steps, where $(x_0, \cdots, x_{l-1})$ is transformed into $(\hat{x}_0, \cdots, \hat{x}_{l-1})$ such that the difference between any two entries is at most 1. At this point, applying Lemma 1 completes the proof. □

C Proof of Lemma 5

Proof. According to Lemma 2,

$$\left\langle \frac{A}{l_1} \right\rangle = \min_{\substack{(\log_2 w_{l_0},\cdots,\log_2 w_{l_0+l_1-1})\in\mathbb{Z}_+^{l_0} \\ \log_2 w_{l_0}+\cdots+\log_2 w_{l_0+l_1-1}=A}} \sum_{i=l_0}^{l_0+l_1-1} w_i.$$

We only need to show that for positive integers A and A' with $A' = A + 1$,

$$\left\langle \frac{A}{l_1} \right\rangle < \left\langle \frac{A'}{l_1} \right\rangle = \min_{\substack{(\log_2 w'_{l_0},\cdots,\log_2 w'_{l_0+l_1-1})\in\mathbb{Z}_+^{l_0} \\ \log_2 w'_{l_0}+\cdots+\log_2 w'_{l_0+l_1-1}=A'}} \sum_{i=l_0}^{l_0+l_1-1} w'_i.$$

Lemma 2 implies $\left\langle \frac{A}{l_1} \right\rangle = (l_1 - k) \cdot 2^{\lfloor \frac{A}{l_1} \rfloor} + k \cdot 2^{\lceil \frac{A}{l_1} \rceil}$, where $\log_2 w_0 = \cdots = \log_2 w_{l_1-k-1} = \left\lfloor \frac{A}{l_1} \right\rfloor$, $\log_2 w_{l_1-k} = \cdots = \log_2 w_{l_1-1} = \left\lceil \frac{A}{l_1} \right\rceil$, $k = A - l_1 \cdot \left\lfloor \frac{A}{l_1} \right\rfloor$, and $\left\langle \frac{A'}{l_1} \right\rangle = (l_1 - k') \cdot 2^{\lfloor \frac{A'}{l_1} \rfloor} + k' \cdot 2^{\lceil \frac{A'}{l_1} \rceil}$, where $\log_2 w'_0 = \cdots = \log_2 w'_{l_1-k'-1} = \left\lfloor \frac{A'}{l_1} \right\rfloor$, $\log_2 w'_{l_1-k'} = \cdots = \log_2 w'_{l_1-1} = \left\lceil \frac{A'}{l_1} \right\rceil$, $k' = A' - l_1 \cdot \left\lfloor \frac{A'}{l_1} \right\rfloor$. If l_1 divides A,

$$\left\langle \frac{A}{l_1} \right\rangle = l_1 \cdot 2^{\frac{A}{l_1}} < l_1 \cdot 2^{\frac{A+1}{l_1}} \leq \left\langle \frac{A'}{l_1} \right\rangle.$$

If l_1 does not divide A, we have $\left\lfloor \frac{A}{l_1} \right\rfloor + 1 = \left\lceil \frac{A}{l_1} \right\rceil$, then if l_1 does not divide A', we have $\left\lfloor \frac{A'}{l_1} \right\rfloor = \left\lfloor \frac{A}{l_1} \right\rfloor$ and $\left\lceil \frac{A'}{l_1} \right\rceil = \left\lceil \frac{A}{l_1} \right\rceil$, which implies $k' = k + 1$. Therefore,

$$\left\langle \frac{A'}{l_1} \right\rangle - \left\langle \frac{A}{l_1} \right\rangle = 2^{\left\lfloor \frac{A}{l_1} \right\rfloor},$$

that is $\left\langle \frac{A'}{l_1} \right\rangle > \left\langle \frac{A}{l_1} \right\rangle$. If l_1 divides A', then $k' = 0$ and $\left\lfloor \frac{A}{l_1} \right\rfloor + 1 = \left\lceil \frac{A'}{l_1} \right\rceil$, which implies that $\left\langle \frac{A'}{l_1} \right\rangle = l_1 \cdot 2^{\left\lfloor \frac{A}{l_1} \right\rfloor + 1}$ and $k = l - 1$. Therefore,

$$\left\langle \frac{A'}{l_1} \right\rangle - \left\langle \frac{A}{l_1} \right\rangle = 2^{\left\lfloor \frac{A}{l_1} \right\rfloor},$$

that is $\left\langle \frac{A'}{l_1} \right\rangle > \left\langle \frac{A}{l_1} \right\rangle$. □

D Additional Parameter Sets for Inhomogeneous WOTS$^+$

Table 10: More parameter setting for inhomogeneous WOTS$^+$ with $n = 16$

l_0	l_1	$(w_0, \cdots, w_{l_0-1})$	$(w_{l_0}, \cdots, w_{l_0+l_1-1})$	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
30	2	$[16]^{22} \times [32]^{8}$	$[32]^{2}$	512	672	640
31	2	$[16]^{27} \times [32]^{4}$	$[32]^{2}$	528	624	591
32	2	$[16]^{32}$	$[32]^{2}$	544	560	526
32	3	$[16]^{32}$	$[8]^{3}$	560	536	501
33	3	$[8]^{4} \times [16]^{29}$	$[8]^{3}$	576	520	484
34	3	$[8]^{8} \times [16]^{26}$	$[8]^{3}$	592	504	467
35	3	$[8]^{12} \times [16]^{23}$	$[8]^{3}$	608	488	450
36	3	$[8]^{16} \times [16]^{20}$	$[8]^{3}$	624	472	433
37	3	$[8]^{20} \times [16]^{17}$	$[8]^{3}$	640	456	416
38	3	$[8]^{24} \times [16]^{14}$	$[8]^{3}$	656	440	399
39	3	$[8]^{28} \times [16]^{11}$	$[8]^{3}$	672	424	382

Table 11: More parameter setting for inhomogeneous WOTS$^+$ with $n = 24$

l_0	l_1	$(w_0, \cdots, w_{l_0-1})$	$(w_{l_0}, \cdots, w_{l_0+l_1-1})$	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
45	2	$[16]^{33} \times [32]^{12}$	$[32]^{2}$	1128	976	929
46	2	$[16]^{38} \times [32]^{8}$	$[32]^{2}$	1152	928	880
47	2	$[16]^{43} \times [32]^{4}$	$[32]^{2}$	1176	880	831
48	2	$[16]^{48}$	$[32]^{2}$	1200	832	782
48	3	$[16]^{48}$	$[8]^{2} \times [16]$	1224	800	749
49	3	$[8]^{4} \times [16]^{45}$	$[8]^{2} \times [16]$	1248	784	732
50	3	$[8]^{8} \times [16]^{42}$	$[8]^{2} \times [16]$	1272	768	715
51	3	$[8]^{12} \times [16]^{39}$	$[8]^{2} \times [16]$	1296	752	698
52	3	$[8]^{16} \times [16]^{36}$	$[8]^{2} \times [16]$	1320	736	681
53	3	$[8]^{20} \times [16]^{33}$	$[8]^{2} \times [16]$	1344	720	664
54	3	$[8]^{24} \times [16]^{30}$	$[8]^{2} \times [16]$	1368	704	647
55	3	$[8]^{28} \times [16]^{27}$	$[8]^{2} \times [16]$	1392	688	630
56	3	$[8]^{32} \times [16]^{24}$	$[8]^{2} \times [16]$	1416	672	613
57	3	$[8]^{36} \times [16]^{21}$	$[8]^{2} \times [16]$	1440	656	596
58	3	$[8]^{40} \times [16]^{18}$	$[8]^{2} \times [16]$	1464	640	579

Table 12: More parameter setting for inhomogeneous $\texttt{WOTS}^+$ with $n = 32$

l_0	l_1	$(w_0, \cdots, w_{l_0-1})$	$(w_{l_0}, \cdots, w_{l_0+l_1-1})$	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
60	3	$[16]^{44} \times [32]^{16}$	$[8] \times [16]^2$	2016	1256	1193
61	3	$[16]^{49} \times [32]^{12}$	$[8] \times [16]^2$	2048	1208	1144
63	2	$[16]^{59} \times [32]^4$	$[32]^2$	2080	1136	1071
64	2	$[16]^{64}$	$[32]^2$	2112	1088	1022
64	3	$[16]^{64}$	$[8]^2 \times [16]$	2144	1056	989
65	3	$[8]^4 \times [16]^{61}$	$[8]^2 \times [16]$	2176	1040	972
66	3	$[8]^8 \times [16]^{58}$	$[8]^2 \times [16]$	2208	1024	955
67	3	$[8]^{12} \times [16]^{55}$	$[8]^2 \times [16]$	2240	1008	938
68	3	$[8]^{16} \times [16]^{52}$	$[8]^2 \times [16]$	2272	992	921
69	3	$[8]^{20} \times [16]^{49}$	$[8]^2 \times [16]$	2304	976	904
70	3	$[8]^{24} \times [16]^{46}$	$[8]^2 \times [16]$	2336	960	887
71	3	$[8]^{28} \times [16]^{43}$	$[8]^2 \times [16]$	2368	944	870
72	3	$[8]^{32} \times [16]^{40}$	$[8]^2 \times [16]$	2400	928	853
73	3	$[8]^{36} \times [16]^{37}$	$[8]^2 \times [16]$	2432	912	836
74	3	$[8]^{40} \times [16]^{34}$	$[8]^2 \times [16]$	2464	896	819
75	3	$[8]^{44} \times [16]^{31}$	$[8]^2 \times [16]$	2496	880	802
76	3	$[8]^{48} \times [16]^{28}$	$[8]^2 \times [16]$	2528	864	785
77	3	$[8]^{52} \times [16]^{25}$	$[8]^2 \times [16]$	2560	848	768

E Additional Parameter Sets for Inhomogeneous $\texttt{WOTS}^{\alpha}$

Table 13: More parameter setting for inhomogeneous $\texttt{WOTS}^{\alpha}$ with $n = 16$

l	$(w_0, \cdots, w_{l-1})$	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
45	$[7]^9 \times [8]^{36}$	720	198	153
43	$[8]^{17} \times [9]^{26}$	688	206	164
41	$[9]^{17} \times [10]^{24}$	656	217	176
39	$[10]^9 \times [11]^{30}$	624	229	191
38	$[11]^{20} \times [12]^{18}$	608	237	199
37	$[12]^{27} \times [13]^{10}$	592	245	209
36	$[13]^{30} \times [14]^{6}$	576	255	219
35	$[14]^{28} \times [15]^{7}$	560	266	231
34	$[15]^{21} \times [16]^{13}$	544	278	245
33	$[16]^{8} \times [17]^{25}$	528	293	260
32	$[18]^{21} \times [19]^{11}$	512	309	278
31	$[20]^{25} \times [21]^{6}$	496	328	298
30	$[22]^{19} \times [23]^{11}$	480	350	321
29	$[24] \times [25]^{28}$	464	376	348
28	$[28]^{25} \times [29]^{3}$	448	407	380
27	$[32] \times [32]^{26}$	432	445	418
26	$[36]^9 \times [37]^{17}$	416	489	464
25	$[42]^{11} \times [43]^{14}$	400	544	520

Table 14: More parameter setting for inhomogeneous $\texttt{WOTS}^{\alpha}$ with $n = 24$

l	$(w_0, \cdots, w_{l-1})$	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
66	$[7]^2 \times [8]^{64}$	1584	296	230
63	$[8]^{12} \times [9]^{51}$	1512	309	246
60	$[9]^{10} \times [10]^{50}$	1440	325	265
58	$[10]^{20} \times [11]^{38}$	1392	338	280
56	$[11]^{22} \times [12]^{34}$	1344	353	297
54	$[12]^{15} \times [13]^{39}$	1296	370	317
53	$[13]^{34} \times [14]^{19}$	1272	380	328
51	$[14]^{9} \times [15]^{42}$	1224	403	353
50	$[15]^{17} \times [16]^{33}$	1200	416	367
49	$[16]^{21} \times [17]^{28}$	1176	430	382
48	$[17]^{20} \times [18]^{28}$	1152	446	398
47	$[18]^{14} \times [19]^{33}$	1128	463	416
46	$[19]^{2} \times [20]^{44}$	1104	482	436
45	$[21]^{29} \times [22]^{16}$	1080	503	458
44	$[22]^{3} \times [23]^{41}$	1056	526	483
43	$[24]^{13} \times [25]^{30}$	1032	552	510
42	$[26]^{13} \times [27]^{29}$	1008	581	540
41	$[28]^{2} \times [29]^{39}$	984	614	573
40	$[31]^{17} \times [32]^{23}$	960	651	612
39	$[34]^{17} \times [35]^{22}$	936	693	655
38	$[38]^{38}$	912	741	703
37	$[42]^{36} \times [43]$	888	796	759
36	$[46]^{9} \times [47]^{27}$	864	859	824

Table 15: More parameter setting for inhomogeneous $\texttt{WOTS}^{\alpha}$ with $n = 32$

l	$(w_0, \cdots, w_{l-1})$	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
88	$[7]^{11} \times [8]^{77}$	2816	390	303
83	$[8]^{7} \times [9]^{76}$	2656	411	329
79	$[9]^{2} \times [10]^{77}$	2528	433	355
76	$[10]^{5} \times [11]^{71}$	2432	453	378
74	$[11]^{24} \times [12]^{50}$	2368	469	395
71	$[12]^{3} \times [13]^{68}$	2272	495	425
69	$[13]^{2} \times [14]^{67}$	2208	516	448
68	$[14]^{32} \times [15]^{36}$	2176	528	460
66	$[15]^{15} \times [16]^{51}$	2112	553	488
65	$[16]^{35} \times [17]^{30}$	2080	567	503
63	$[18]^{63}$	2016	598	536
62	$[18]^{7} \times [19]^{55}$	1984	616	555
61	$[19]^{11} \times [20]^{50}$	1952	635	574
60	$[20]^{9} \times [21]^{51}$	1920	655	596
59	$[21]^{2} \times [22]^{57}$	1888	677	619
58	$[23]^{47} \times [24]^{11}$	1856	701	644
57	$[24]^{27} \times [25]^{30}$	1824	727	671
56	$[25] \times [26]^{55}$	1792	755	700
55	$[27]^{22} \times [28]^{33}$	1760	786	732
54	$[29]^{34} \times [30]^{20}$	1728	820	766
53	$[31]^{35} \times [32]^{18}$	1696	857	804
52	$[33]^{25} \times [34]^{27}$	1664	897	846
51	$[35]^{3} \times [36]^{48}$	1632	942	891
50	$[38]^{17} \times [39]^{33}$	1600	991	942
49	$[41]^{14} \times [42]^{35}$	1568	1046	998
48	$[45]^{41} \times [46]^{7}$	1536	1107	1060

F Additional Parameter Sets for Inhomogeneous $\texttt{WOTS}^{\texttt{C}}$

Table 16: More parameter setting for inhomogeneous $\texttt{WOTS}^{\texttt{C}}$ with $n = 16$

l	$(w_0, \cdots, w_{l-1})$	z_b	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
17	$[128]^8 \times [256]^9$	0	276	2293	1656
18	$[128]^{16} \times [256]^2$	0	292	1748	1271
19	$[64]^5 \times [128]^{14}$	0	308	1430	1047
20	$[64]^{12} \times [128]^8$	0	324	1216	886
21	$[64]^{19} \times [128]^2$	0	340	989	726
22	$[32]^4 \times [64]^{18}$	0	356	854	629
23	$[32]^{10} \times [64]^{13}$	0	372	770	565
24	$[32]^{16} \times [64]^8$	0	388	685	500
25	$[32]^{22} \times [64]^3$	0	404	596	436
26	$[16]^2 \times [32]^{24}$	0	420	528	387
27	$[16]^7 \times [32]^{20}$	0	436	497	363
28	$[16]^{12} \times [32]^{16}$	0	452	467	338
29	$[16]^{17} \times [32]^{12}$	0	468	435	314
30	$[16]^{22} \times [32]^8$	0	484	404	289
31	$[16]^{27} \times [32]^4$	0	500	371	265
32	$[16]^{32}$	0	516	337	240
33	$[8]^4 \times [16]^{29}$	0	532	327	232

Table 17: More parameter setting for inhomogeneous $\texttt{WOTS}^{\texttt{C}}$ with $n = 24$

l	$(w_0, \cdots, w_{l-1})$	z_b	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
26	$[128]^{16} \times [256]^{10}$	0	628	3016	2291
27	$[128]^{24} \times [256]^{3}$	0	652	2493	1907
28	$[64]^{4} \times [128]^{24}$	0	676	2143	1650
29	$[64]^{11} \times [128]^{18}$	0	700	1942	1490
30	$[64]^{18} \times [128]^{12}$	0	724	1737	1329
31	$[64]^{25} \times [128]^{6}$	0	748	1525	1169
32	$[64]^{32}$	0	772	1303	1008
33	$[32]^{6} \times [64]^{27}$	0	796	1224	944
34	$[32]^{12} \times [64]^{22}$	0	820	1145	879
35	$[32]^{18} \times [64]^{17}$	0	844	1064	815
36	$[32]^{24} \times [64]^{12}$	0	868	983	750
37	$[32]^{30} \times [64]^{7}$	0	892	899	686
38	$[32]^{36} \times [64]^{2}$	0	916	813	621
39	$[16]^{3} \times [32]^{36}$	0	940	759	581
40	$[16]^{8} \times [32]^{32}$	0	964	731	556
41	$[16]^{13} \times [32]^{28}$	0	988	701	532
42	$[16]^{18} \times [32]^{24}$	0	1012	673	507
43	$[16]^{23} \times [32]^{20}$	0	1036	642	483
44	$[16]^{28} \times [32]^{16}$	0	1060	613	458
45	$[16]^{33} \times [32]^{12}$	0	1084	582	434
46	$[16]^{38} \times [32]^{8}$	0	1108	552	409
47	$[16]^{43} \times [32]^{4}$	0	1132	520	385
48	$[16]^{48}$	0	1156	488	360
49	$[8]^{4} \times [16]^{45}$	0	1180	478	352

Table 18: More parameter setting for inhomogeneous $\texttt{WOTS}^{\texttt{C}}$ with $n = 32$

l	$(w_0, \cdots, w_{l-1})$	z_b	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
41	$[64]^{31} \times [128]^{10}$	0	1316	2044	1612
42	$[64]^{38} \times [128]^{4}$	0	1348	1835	1451
43	$[32]^{2} \times [64]^{41}$	0	1380	1664	1323
44	$[32]^{8} \times [64]^{36}$	0	1412	1588	1258
45	$[32]^{14} \times [64]^{31}$	0	1444	1511	1194
46	$[32]^{20} \times [64]^{26}$	0	1476	1433	1129
47	$[32]^{26} \times [64]^{21}$	0	1508	1354	1065
48	$[32]^{32} \times [64]^{16}$	0	1540	1275	1000
49	$[32]^{38} \times [64]^{11}$	0	1572	1194	936
50	$[32]^{44} \times [64]^{6}$	0	1604	1112	871
51	$[32]^{50} \times [64]^{1}$	0	1636	1027	807
52	$[16]^{4} \times [32]^{48}$	0	1668	988	774
53	$[16]^{9} \times [32]^{44}$	0	1700	959	750
54	$[16]^{14} \times [32]^{40}$	0	1732	932	725
55	$[16]^{19} \times [32]^{36}$	0	1764	903	701
56	$[16]^{24} \times [32]^{32}$	0	1796	875	676
57	$[16]^{29} \times [32]^{28}$	0	1828	845	652
58	$[16]^{34} \times [32]^{24}$	0	1860	817	627
59	$[16]^{39} \times [32]^{20}$	0	1892	787	603
60	$[16]^{44} \times [32]^{16}$	0	1924	758	578
61	$[16]^{49} \times [32]^{12}$	0	1956	728	554
62	$[16]^{54} \times [32]^{8}$	0	1988	698	529
63	$[16]^{59} \times [32]^{4}$	0	2020	667	505
64	$[16]^{64}$	0	2052	636	480
65	$[8]^{4} \times [16]^{61}$	0	2084	627	472

G Proof of Lemma 7

Proof. According to the definition of $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$, we have the partition:

$$\mathcal{C}_s\left(\prod_{j=0}^{l-1}[w_j]\right) = \bigcup_{i=0}^{\min\{w_{l-1},s\}} \left\{ v \in \prod_{j=0}^{l-1}[w_j] : \sum_{j=0}^{l-2} v_j = s-i, v_{l-1} = i \right\}.$$

Therefore, we have

$$\begin{aligned}\left|\mathcal{C}_s\left(\prod_{j=0}^{l-1}[w_j]\right)\right| &= \sum_{i=0}^{\min\{w_{l-1},s\}} \left| \left\{ v \in \prod_{j=0}^{l-1}[w_j] : \sum_{j=0}^{l-2} v_j = s-i, v_{l-1} = i \right\} \right| \\ &= \sum_{i=0}^{\min\{w_{l-1},s\}} \left|\mathcal{C}_{s-i}\left(\prod_{j=0}^{l-2}[w_j]\right)\right|.\end{aligned}$$

□

H Proof of Theorem 2

Lemma 15. *For any positive integer s,*

$$\left|\left\{(x_0,\cdots,x_{\ell-1})\in\mathbb{N}^{\ell}:\sum_{j=0}^{\ell-1}x_i=s\right\}\right|=\binom{s+\ell-1}{\ell-1}.$$

Proof. We consider s identical balls and $\ell-1$ identical dividers to separate the balls into ℓ groups. Each distribution corresponds uniquely to a non-negative integer solution of the equation $x_0+x_1+\cdots+x_{l-1}=s$. Assuming there are $s+\ell-1$ positions, we choose $\ell-1$ positions to hold the dividers and s positions to hold the balls. There are $\binom{s+\ell-1}{\ell-1}$ distinct possibilities. □

Lemma 16 (Inclusion–exclusion principle). *Let $\mathbb{A}_i$ be finite sets for $i\in\{1,\cdots,n\}$. Then,*

$$\left|\bigcup_{i=1}^{n}\mathbb{A}_i\right|=\sum_{k=1}^{n}(-1)^{k+1}\left(\sum_{1\le i_1<\cdots<i_k\le n}\left|\mathbb{A}_{i_1}\cap\cdots\cap\mathbb{A}_{i_k}\right|\right).$$

Proof. Can be found in standard textbook on combinatorics. □

Now, we are ready to give the proof of Theorem 2

Proof. Let $\mathbb{X}=\{(x_0,\cdots,x_{l-1})\in\mathbb{N}^l:\sum_{i=0}^{l-1}x_i=s\}$. According to Lemma 15, we have $|\mathbb{X}|=\binom{s+l-1}{l-1}$. For $0\le j\le l-1$, we define

$$\mathbb{A}_j=\{(x_0,\cdots,x_{l-1})\in\mathbb{N}^l:\sum_{i=0}^{l-1}x_i=s,x_j>w_j\},$$

then it follows that

$$\overline{\mathbb{A}_j}=\{(x_0,\cdots,x_{l-1})\in\mathbb{N}^l:\sum_{i=0}^{l-1}x_i=s,x_j\le w_j-1\}.$$

Consequently, we have

$$\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])=\bigcap_{j=0}^{l-1}\overline{\mathbb{A}_j}=\overline{\bigcup_{j=0}^{l-1}\mathbb{A}_j}=\mathbb{X}-\bigcup_{j=0}^{l-1}\mathbb{A}_j,$$

and hence $\zeta_{\mathbf{w}}(l,s)=|\mathbb{X}|-\left|\bigcup_{j=0}^{l-1}\mathbb{A}_j\right|$. It follows from Lemma 16 that

$$\left|\bigcup_{j=0}^{l-1}\mathbb{A}_j\right|=\sum_{k=1}^{l}(-1)^{k-1}\sum_{0\le i_1<\cdots<i_k\le l-1}\left|\mathbb{A}_{i_1}\cap\cdots\cap\mathbb{A}_{i_k}\right|.$$

For $k \geq 1$, and any $w_{i_1}, \cdots, w_{i_k}$ such that $w_{i_1} + \cdots + w_{i_k} \leq s$, where $0 \leq i_1 < \cdots < i_k \leq l-1$, we have

$$\mathbb{A}_{i_1} \cap \cdots \cap \mathbb{A}_{i_k} = \{(x_0, \cdots, x_{l-1}) \in \mathbb{N}^l : \sum_{i=0}^{l-1} x_i = s, x_{i_1} > w_{i_1}, \cdots, x_{i_k} > w_{i_k}\}.$$

Lemma 15 implies that

$$|\mathbb{A}_{i_1} \cap \cdots \cap \mathbb{A}_{i_k}| = \binom{s+l-w_{i_1}-\cdots-w_{i_k}-1}{l-1}.$$

For $k \geq 1$, and any $w_{i_1}, \cdots, w_{i_k}$ such that $w_{i_1} + \cdots + w_{i_k} > s$, where $0 \leq i_1 < \cdots < i_k \leq l-1$, $|\mathbb{A}_{i_1} \cap \cdots \cap \mathbb{A}_{i_k}| = 0$. Therefore, we have

$$\begin{aligned}
\zeta_{\mathbf{w}}(l,s) = & \ |\mathbb{X}| - \left|\bigcup_{j=0}^{l-1} \mathbb{A}_j\right| \\
= & \ \binom{s+l-1}{l-1} - \left|\bigcup_{j=0}^{l-1} \mathbb{A}_j\right| \\
= & \ \binom{s+l-1}{l-1} - \sum_{k=1}^{l} (-1)^{k-1} \sum_{\substack{(i_1,\cdots,i_k)\in[l]^k \\ i_1<i_2\cdots<i_k \\ w_{i_1}+\cdots+w_{i_k}\leq s}} \binom{s+l-w_{i_1}-\cdots-w_{i_k}-1}{l-1}.
\end{aligned}$$

□

I Proof of Theorem 3

Lemma 17. *Let* $\mathbf{w} = (w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$, $l \geq 1$, *and* $0 \leq s \leq \lceil \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \rceil$. *Then,* $\zeta_{\mathbf{w}}(l,s) = \zeta_{\mathbf{w}}(l, \sum_{i=0}^{l-1}(w_i-1) - s)$.

Proof. We consider a mapping $\rho : \mathcal{C}_s(\prod_{j=0}^{l-1}[w_j]) \to \mathcal{C}_{\sum_{i=0}^{l-1}(w_i-1)-s}(\prod_{j=0}^{l-1}[w_j])$, such that for $\mathbf{x} = (x_0, \cdots, x_{l-1}) \in \mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$,

$$\rho(\mathbf{x}) = (w_0 - 1 - x_0, \cdots, w_{l-1} - 1 - x_{l-1}).$$

Obviously, the mapping ρ is bijective. Therefore,

$$\left| \mathcal{C}_s \left(\prod_{j=0}^{l-1}[w_j] \right) \right| = \left| \mathcal{C}_{\sum_{i=0}^{l-1}(w_i-1)-s} \left(\prod_{j=0}^{l-1}[w_j] \right) \right|,$$

or equivalently $\zeta_{\mathbf{w}}(l,s) = \zeta_{\mathbf{w}}(l, \sum_{i=0}^{l-1}(w_i-1) - s)$. □

Lemma 18. *Let* x *be a non-negative integer, then* $\lfloor \frac{x}{2} \rfloor + \lceil \frac{x}{2} \rceil = x$.

Proof. Let $x = 2y$ or $x = 2y+1$, where y is a non-negative integer. If $x = 2y$, $\lfloor \frac{x}{2} \rfloor + \lceil \frac{x}{2} \rceil = y + y = x$, If $x = 2y+1$, $\lfloor \frac{x}{2} \rfloor + \lceil \frac{x}{2} \rceil = y + (y+1) = x$. □

Now, we are ready to give the proof of Theorem 3

Proof. We prove it by induction. When $l = 1$, $\mathbf{w} = (w_0) \in \mathbb{Z}_+$ and $0 \leq s \leq \lceil \frac{w_0-1}{2} \rceil$, $1 = \zeta_{\mathbf{w}}(1,j) \leq \zeta_{\mathbf{w}}(1,s) = 1$ for any $j \in \{0, \cdots, s\}$. When $l \geq 2$ and $\mathbf{w} = (w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$, we assume that $\zeta_{\mathbf{w}}(l-1,j') \leq \zeta_{\mathbf{w}}(l-1,s')$ for any $0 \leq j' \leq s' \leq \lceil \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \rceil$. It remains to prove that $\zeta_{\mathbf{w}}(l,s-1) \leq \zeta_{\mathbf{w}}(l,s)$ with $0 \leq s \leq \lceil \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \rceil$. Proposition 2 implies

$$\begin{cases} \zeta_{\mathbf{w}}(l,s) = \sum_{i=0}^{\min\{w_{l-1}-1,s\}} \zeta_{\mathbf{w}}(l-1,s-i) \\ \zeta_{\mathbf{w}}(l,s-1) = \sum_{i=0}^{\min\{w_{l-1}-1,s-1\}} \zeta_{\mathbf{w}}(l-1,s-1-i) \end{cases}.$$

If $w_{l-1} - 1 \geq s$, we have

$$\begin{aligned} &\zeta_{\mathbf{w}}(l,s) - \zeta_{\mathbf{w}}(l,s-1) \\ &= \zeta_{\mathbf{w}}(l-1,s) + \cdots + \zeta_{\mathbf{w}}(l-1,0) - (\zeta_{\mathbf{w}}(l-1,s-1) + \cdots + \zeta_{\mathbf{w}}(l-1,0)) \\ &= \zeta_{\mathbf{w}}(l-1,s) \geq 0. \end{aligned}$$

That is, $\zeta_{\mathbf{w}}(l,s-1) \leq \zeta_{\mathbf{w}}(l,s)$. If $w_{l-1} - 1 < s$, we have

$$\begin{aligned} &\zeta_{\mathbf{w}}(l,s) - \zeta_{\mathbf{w}}(l,s-1) \\ &= \zeta_{\mathbf{w}}(l-1,s) + \cdots + \zeta_{\mathbf{w}}(l-1,s-(w_{l-1}-1)) \\ &\quad - (\zeta_{\mathbf{w}}(l-1,s-1) + \cdots + \zeta_{\mathbf{w}}(l-1,s-1-(w_{l-1}-1))) \\ &= \zeta_{\mathbf{w}}(l-1,s) - \zeta_{\mathbf{w}}(l-1,s-w_{l-1}). \end{aligned}$$

If $0 \leq s \leq \left\lceil \frac{\sum_{i=0}^{l-2}(w_i-1)}{2} \right\rceil$, the above assumption implies that

$$\zeta_{\mathbf{w}}(l-1,s) \geq \zeta_{\mathbf{w}}(l-1,s-w_{l-1}).$$

Therefore, $\zeta_{\mathbf{w}}(l,s-1) \leq \zeta_{\mathbf{w}}(l,s)$. If $\left\lceil \frac{\sum_{i=0}^{l-2}(w_i-1)}{2} \right\rceil \leq s \leq \left\lceil \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rceil$, it follows from Lemma 17 that $\zeta_{\mathbf{w}}(l-1,s) = \zeta_{\mathbf{w}}(l-1,\sum_{i=0}^{l-2}(w_i-1)-s)$, which implies

$$\zeta_{\mathbf{w}}(l,s) - \zeta_{\mathbf{w}}(l,s-1) = \zeta_{\mathbf{w}}\left(l-1,\sum_{i=0}^{l-2}(w_i-1)-s\right) - \zeta_{\mathbf{w}}(l-1,s-w_{l-1}).$$

Lemma 18 implies that

$$\begin{aligned}\sum_{i=0}^{l-2}(w_i-1)-s &\leq \sum_{i=0}^{l-2}(w_i-1) - \left\lceil \frac{\sum_{i=0}^{l-2}(w_i-1)}{2} \right\rceil \\ &= \left\lfloor \frac{\sum_{i=0}^{l-2}(w_i-1)}{2} \right\rfloor \leq \left\lceil \frac{\sum_{i=0}^{l-2}(w_i-1)}{2} \right\rceil\end{aligned}$$

Since $s \leq \left\lceil \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rceil \leq \frac{\sum_{i=0}^{l-1}(w_i-1)+1}{2} = \frac{\sum_{i=0}^{l-2}(w_i-1)+w_{l-1}}{2}$, then $s-w_{l-1} \leq \sum_{i=0}^{l-2}(w_i-1)-s$. Therefore, we have

$$\zeta_{\mathbf{w}}\left(l-1,\sum_{i=0}^{l-2}(w_i-1)-s\right) \geq \zeta_{\mathbf{w}}(l-1,s-w_{l-1}),$$

that is $\zeta_{\mathbf{w}}(l,s-1) \leq \zeta_{\mathbf{w}}(l,s)$. □

J Proof of Theorem 4

Lemma 19. *Let* $\mathbf{w} = (w_0, \cdots, w_{l-1}) \in \mathbb{Z}_+^l$, $l \geq 1$. *Then,*

$$\zeta_{\mathbf{w}}\left(l, \left\lceil \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rceil\right) = \zeta_{\mathbf{w}}\left(l, \left\lfloor \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rfloor\right).$$

Proof. If 2 divides $(w_0-1)+\cdots+(w_{l-1}-1)$, then

$$\left\lceil \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rceil = \left\lfloor \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rfloor.$$

If 2 does not divide $(w_0-1)+\cdots+(w_{l-1}-1)$, then

$$\zeta_{\mathbf{w}}\left(l, \left\lceil \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rceil\right) = \zeta_{\mathbf{w}}\left(l, \left\lfloor \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rfloor + 1\right).$$

According to Lemma 17 and Lemma 18, we have

$$\begin{aligned}
&\zeta_{\mathbf{w}}\left(l, \left\lceil \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rceil\right) \\
&= \zeta_{\mathbf{w}}\left(l, \left\lfloor \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rfloor + 1\right) \\
&= \zeta_{\mathbf{w}}\left(l, \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} - \left\lfloor \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rfloor - 1\right) \\
&= \zeta_{\mathbf{w}}\left(l, \left\lceil \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rceil - 1\right) \\
&= \zeta_{\mathbf{w}}\left(l, \left\lfloor \frac{(w_0-1)+\cdots+(w_{l-1}-1)}{2} \right\rfloor\right).
\end{aligned}$$

□

Lemma 19 tells us that, there is no contradiction between Theorem 3 and Theorem 4.

Definition 4 (Poset). *A poset* $(\mathcal{S}, \leq)$ *consists of a set* $\mathcal{S}$ *together with an antisymmetric, transitive and reflexive binary relation '*$\leq$*', where are certain pairs* $(x, y) \in \mathcal{S}$ *are comparable. We refer to the poset* $(\mathcal{S}, \leq)$ *simply as* $\mathcal{S}$*.*

Definition 5 ((Anti) chain and decomposition). *A chain (resp., antichain) refers to a subset of a poset, whose every pair of elements is comparable (resp., incomparable). A chain decomposition is a partition of a poset into disjoint chains. Note that a set consisting of a single element from the poset* $\mathcal{S}$ *is both a chain and an antichain.*

Lemma 20. *Let* $\mathbb{A}$ *is an antichain of a finite poset* $\mathcal{S}$*, if there exists a chain decomposition of* $\mathcal{S}$ *with exactly* $|\mathbb{A}|$ *chains, then* $\mathbb{A}$ *is the maximum antichain.*

Proof. Let $\mathcal{S}^{(1)}, \cdots, \mathcal{S}^{(|\mathbb{A}|)}$ be pairwise disjoint chains in $\mathcal{S}$ such that $\mathcal{S}^{(1)} \cup \cdots \cup \mathcal{S}^{(|\mathbb{A}|)} = \mathcal{S}$. Suppose there exists an antichain $\mathbb{B} \in \mathcal{S}$ with $|\mathbb{B}| \geq |\mathbb{A}| + 1$, then we can find an integer $j \in \{1, \cdots, |\mathbb{A}|\}$ such that $\mathcal{S}^{(j)}$ contains at least two incomparable elements of $\mathbb{B}$. This contradicts the fact that $\mathcal{S}^{(j)}$ is a chain. □

Now, we are ready to give the proof of Theorem 4

Proof. Let $\mathcal{S}_l = (\prod_{j=0}^{l-1}[w_j], \leq)$ be a finite poset. According to Lemma 20, we can prove that $\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])$ is the maximum antichain of $\mathcal{S}_l$ by arguing that (1) $\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])$ is an antichain and (2) there exists a chain decomposition for $\mathcal{S}_l$ which contains $|\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])|$ chains.

We have proved that $\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])$ is an antichain in Lemma 6. It remains to prove that there exists a chain decomposition for $\mathcal{S}_l$ which contains $|\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])|$ chains, we could construct it by induction as follows.

We denote an element of $\mathcal{S}_l$ by $\alpha_i = (a_1, \cdots, a_l) \in \prod_{j=0}^{l-1}[w_j]$ and define $\parallel \alpha_i \parallel \stackrel{\text{def}}{=} \sum_{i=1}^{l} a_i$. We could construct the chain decomposition for $\mathcal{S}_l$, where every chain $\langle \alpha_1, \cdots, \alpha_h \rangle$ satisfies the following two properties:

- $\parallel \alpha_{i+1} \parallel = \parallel \alpha_i \parallel +1, \forall i \in \{1, 2, \cdots, h-1\}$,
- $\parallel \alpha_1 \parallel + \parallel \alpha_h \parallel = \sum_{j=0}^{l-1}(w_j - 1)$.

When $l = 1$, $\mathcal{S}_1 = ([w_0], \leq)$ and $\langle (0), \cdots, (w_0 - 1) \rangle$ is a chain of $\mathcal{S}_1$ that satisfies the two stated properties. Therefore we find a chain decomposition whose size equals $|\mathcal{C}_{\lfloor \frac{w_0-1}{2} \rfloor}([w_0])| = 1$.

Assume that we already have a chain decomposition for $\mathcal{S}_{l-1}$ satisfying the two stated properties, then we could extend it to construct a chain decomposition for $\mathcal{S}_l$ as follows. For each chain $\alpha = \langle \alpha_1, \alpha_2, \cdots, \alpha_h \rangle$ from the chain decomposition for $\mathcal{S}_{l-1}$ satisfying the two stated properties, we could construct $k+1$ chains for $\mathcal{S}_l$, where $k = \min\{w_{l-1} - 1, h - 1\}$. For every $j \in \{0, \cdots, k\}$ the j-th chain is defined as follows:

$$\langle (\alpha_1, j), \cdots, (\alpha_{h-j}, j), (\alpha_{h-j}, j+1), \cdots, (\alpha_{h-j}, w_{l-1} - 1) \rangle .$$

The length of the new chain equals

$$(h - j - 1 + 1) + (w_{l-1} - 1 - (j+1) + 1) = h + w_{l-1} - 1 - 2j,$$

we let the new chain be $\beta = \langle \beta_1, \beta_2, \cdots, \beta_{h+w_{l-1}-1-2j} \rangle$, which holds $\parallel \beta_{i+1} \parallel = \parallel \beta_i \parallel +1$ and

$$\begin{aligned} \parallel \beta_1 \parallel + \parallel \beta_{h+w_{l-1}-1-2j} \parallel &= \parallel \alpha_1 \parallel + j + \parallel \alpha_{h-j} + w_{l-1} - 1 \\ &= \parallel \alpha_1 \parallel + j + \parallel \alpha_h \parallel - j + w_{l-1} - 1 \\ &= \parallel \alpha_1 \parallel + \parallel \alpha_h \parallel + w_{l-1} - 1 \\ &= \sum_{j=0}^{l-1}(w_j - 1). \end{aligned}$$

To provide a more intuitive understanding of the construction of such chain decomposition, we present a concrete example. Let $w_0 = 2, w_1 = 3, w_2 = 4$, when $l = 1$, $\mathcal{S}_1 = [w_0]$, the chain decomposition for $\mathcal{S}_1$ is $\langle (0), (1) \rangle$. When $l = 2$, we

could extend the chain decomposition to construct a new chain decomposition for $\mathcal{S}_2 = [w_0] \times [w_1]$:

$$\begin{cases} \langle (0,0),(1,0),(1,1),(1,2) \rangle \\ \langle (0,1),(0,2) \rangle \end{cases}.$$

When $l = 3$, we could extend the chain decomposition to construct a new chain decomposition for $\mathcal{S}_3 = [w_0] \times [w_1] \times [w_2]$:

$$\begin{cases} \langle (0,0,0),(1,0,0),(1,1,0),(1,2,0),(1,2,1)(1,2,2),(1,2,3) \rangle \\ \langle (0,0,1),(1,0,1),(1,1,1),(1,1,2),(1,1,3) \rangle \\ \langle (0,0,2),(1,0,2),(1,0,3) \rangle \\ \langle (0,1,0),(0,2,0),(0,2,1),(0,2,2),(0,2,3) \rangle \\ \langle (0,1,1),(0,1,2),(0,1,3) \rangle \end{cases}.$$

These $k+1$ chains constructed are pairwise disjoint. We now argue that $k+1$ chains constructed constitute a decomposition of $\{(\alpha_i, z) : 1 \leq i \leq h, z \in [w_{l-1}]\}$. The number of elements contained in these $k+1$ chains is

$$\frac{(h + w_{l-1} - 1) + (h + w_{l-1} - 1 - 2k)}{2} \cdot (k+1) = (h + w_{l-1} - k - 1) \cdot (k+1) = h \cdot w_{l-1},$$

which equals $|\{(\alpha_i, z) : 1 \leq i \leq h, z \in [w_{l-1}]\}|$. Therefore, the union of these $k+1$ chains must be $\{(\alpha_i, z) : 1 \leq i \leq h, z \in [w_{l-1}]\}$. If all chains in a chain decomposition for $\mathcal{S}_{l-1}$ satisfying the two stated properties are extended by the above construction, then we could construct a chain decomposition for $\mathcal{S}_l$.

It remains to prove that the size of the chain decomposition for $\mathcal{S}_l$ equals $|\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])|$. Assume the chain decomposition consists of d chains, each element of the antichain $\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])$ must belong to a chain in the chain decomposition, and no chain contains more than one element belong to $\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])$. Therefore, we have

$$d \geq \left| \mathcal{C}_{\mathsf{s}} \left(\prod_{j=0}^{l-1} [w_j] \right) \right|.$$

The two properties guarantee that every chain contains at least one element α_{mid} such that $\alpha_{mid} = \left\lfloor \frac{\sum_{j=0}^{l-1} w_j - 1}{2} \right\rfloor$. Therefore, we have

$$d \leq \left| \mathcal{C}_{\mathsf{s}} \left(\prod_{j=0}^{l-1} [w_j] \right) \right|.$$

Consequently, $d = |\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])|$, this completes the proof. □

K Proof of Lemma 8

First, we equivalently reformulate Algorithm 1 as Algorithm 4, and present several lemmas regarding Algorithm 4.

Algorithm 4: $\rho^s_{\mathbf{w}} : [|\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])|] \to \mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$

Input: $\mathsf{x} \in [|\mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])|]$
Output: $y = \rho^s_{\mathbf{w}}(\mathsf{x}) \in \mathcal{C}_{\mathsf{s}}(\prod_{j=0}^{l-1}[w_j])$

$x_l \leftarrow \mathsf{x}$
$s_l \leftarrow \mathsf{s}$
for i from l to 1 **do**
  for u from 0 to $\min(w_{i-1} - 1, s)$ **do**
    if $x_i \geq \zeta_{\mathbf{w}}(i-1, s-u)$ **then**
      $x_i \leftarrow x_i - \zeta_{\mathbf{w}}(i-1, s-u)$
    else
      $x_{i-1} \leftarrow x_i$
      $y_{l-i} \leftarrow u$
      break
  $s_{i-1} \leftarrow s_i - y_{l-i}$
return $(y_0, \cdots, y_{l-1})$

Lemma 21 (Algorithm 4). *For $1 \leq i \leq l$,we denote by $\dot{x}_i$ the initial value assigned to x_i, then we have*

$$\dot{x}_i < \zeta_{\mathbf{w}}(i-1, s_i) + \zeta_{\mathbf{w}}(i-1, s_i - 1) + \cdots + \zeta_{\mathbf{w}}(i-1, s_i - \min\{w_{i-1,s_i}\}). \quad (8)$$

Proof. We consider $\dot{x}_{l-j}$ $(0 \leq j < l)$ and prove it by induction. When $j = 0$,

$$\dot{x}_l < \sum_{t=0}^{\min\{w_{l-1}-1, s_l\}} \zeta_{\mathbf{w}}(l-1, s_l - t).$$

When $j = k$, assume that $\dot{x}_{l-k} < \sum_{t=0}^{\min\{w_{l-k-1}-1, s_{l-k}\}} \zeta_{\mathbf{w}}(l-k-1, s_{l-k} - t)$, then there exists a $j^\dagger \in \{0, \cdots, \min\{w_{l-k-1} - 1, s_{l-k}\}\}$ fulfilling

$$\dot{x}_{l-k} - \sum_{t=0}^{j^\dagger - 1} \zeta_{\mathbf{w}}(l-k-1, s_{l-k} - t) < \zeta_{\mathbf{w}}(l-k-1, s_{l-k} - j^\dagger).$$

When $j = k+1$, $s_{l-k-1} = s_{l-k} - j^\dagger$, we have

$$\begin{aligned}\dot{x}_{l-k-1} &= \dot{x}_{l-k} - \sum_{t=0}^{j^\dagger-1} \zeta_{\mathbf{w}}(l-k-1, s_{l-k}-t) \\ &< \zeta_{\mathbf{w}}(l-k-1, s_{l-k}-j^\dagger) \\ &= \sum_{t=0}^{\min\{w_{l-k-2}, s_{l-k-1}\}} \zeta_{\mathbf{w}}(l-k-2, s_{l-k-1}-t).\end{aligned}$$

□

Lemma 22 (Algorithm 4). *For $1 \le i \le l$, if $\dot{x}_i$ satisfies the* Inequality (8)*, then there exists a $j \in \{0, \cdots, \min\{w_{i-1}-1, s_i\}\}$ satisfies*

$$\dot{x}_i - \sum_{k=0}^{j-1} \zeta_{\mathbf{w}}(i-1, s_i-k) < \zeta_{\mathbf{w}}(i-1, s_i-j).$$

Proof. If this lemma does not hold, then

$$\dot{x}_i - \sum_{k=0}^{\min\{w_{i-1}-1, s_i\}-1} \zeta_{\mathbf{w}}(i-1, s_i-k) \ge \zeta_{\mathbf{w}}(i-1, s_i-j),$$

that is, $\dot{x}_i \ge \sum_{k=0}^{\min\{w_{i-1}, s_i\}-1} \zeta_{\mathbf{w}}(i-1, s_i-k)$, which contradicts Lemma 21. □

Lemma 23 (Algorithm 4). $\dot{x}_0 = \dot{x}_1 = s_0 = 0, y_{l-1} = s_1$.

Proof. According to Lemma 21,

$$\dot{x}_1 < \sum_{t=0}^{\min\{w_0-1, s_1\}} \zeta_{\mathbf{w}}(0, s_1-t) \le 1,$$

that is $\dot{x}_1 = 0$. When $0 \le t \le s_1 - 1$, $\dot{x}_1 = 0 \le \zeta_{\mathbf{w}}(0, s_1-t) = 0$, when $t = s_1$, $\dot{x}_1 = 0 < \zeta_{\mathbf{w}}(0, s_1-s_1) = \zeta_{\mathbf{w}}(0,0) = 1$, therefore $\dot{x}_0 = 0, y_{l-1} = s_1$, and $s_0 = s_1 - y_{l-1} = 0$. □

Now, we are ready to give the proof of Theorem 8

Proof. According to Algorithm 4, we have

$$\begin{cases} s_l = s \\ s_{l-1} = s_l - y_0 \\ s_{l-2} = s_{l-1} - y_1 \\ \cdots\cdots \\ s_1 = s_2 - y_{l-2} \\ s_0 = s_1 - y_{l-1} \end{cases}.$$

Therefore, we have $y_0 + \cdots + y_{l-1} = s$ by Lemma 23, that is every image of the mapping $\rho^s_{\mathbf{w}}$ lies in $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$. It remains to prove that $\rho^s_{\mathbf{w}}$ is injective. According to Algorithm 4, we have

$$\begin{cases} \dot{x}_l - \dot{x}_{l-1} = \sum\limits_{t=0}^{y_0-1} \zeta_{\mathbf{w}}(l-1, s_l - t) \\ \dot{x}_{l-1} - \dot{x}_{l-2} = \sum\limits_{t=0}^{y_1-1} \zeta_{\mathbf{w}}(l-2, s_{l-1} - t) \\ \dot{x}_{l-2} - \dot{x}_{l-3} = \sum\limits_{t=0}^{y_2-1} \zeta_{\mathbf{w}}(l-3, s_{l-2} - t) \\ \cdots\cdots \\ \dot{x}_2 - \dot{x}_1 = \sum\limits_{t=0}^{y_{l-2}-1} \zeta_{\mathbf{w}}(1, s_2 - t) \\ \dot{x}_1 - \dot{x}_0 = \sum\limits_{t=0}^{y_{l-1}-1} \zeta_{\mathbf{w}}(0, s_1 - t) \end{cases}.$$

According to Lemma 23, we have

$$x = \dot{x}_l = \dot{x}_l - \dot{x}_0 = \sum_{t=0}^{y_0-1} \zeta_{\mathbf{w}}(l-1, s_l - t) + \cdots + \sum_{t=0}^{y_{l-1}-1} \zeta_{\mathbf{w}}(0, s_1 - t),$$

that is $(y_0, y_1, \cdots, y_{l-1})$ uniquely determine x. □

L Proof of Lemma 10

Lemma 24. *For any two positive integers $a \le b$ and an integer t, we have*

$$|\mathcal{C}_t([a]\times[b])| = \begin{cases} t+1, & 0 \le t \le a-1 \\ a, & a \le t \le b-1 \\ a+b-t-1, & b \le t \le a+b-2 \\ 0, & \text{others} \end{cases}.$$

Proof. It follows from the fact that

$$\mathcal{C}_t([a]\times[b]) = \{(v_a, v_b) \in [a]\times[b] : v_a + v_b = t\}.$$

□

Now we are ready to give the proof of Lemma 10.

Proof. For any positive integers $w_0, \cdots, w_{l-1}$ with $w_0 + \cdots + w_{l-1} = A$, if there exist w_i, w_j where $0 \le i, j < l$ such that $w_i \le w_j - 2$, then we set $w'_i = w_i + 1$ and $w'_j = w_j - 1$.

According to Lemma 24, since $w_i \le w_j$ and $w'_i \le w'_j$, then for any integer t, we have Equation (9) and (10).

$$|\mathcal{C}_t([w_i]\times[w_j])| = \begin{cases} t+1, & 0 \le t \le w_i - 1 \\ w_i, & w_i \le t \le w_j - 1 \\ w_j + w_i - t - 1, & w_j \le t \le w_j + w_i - 2 \\ 0, & \text{others} \end{cases}. \tag{9}$$

$$|\mathcal{C}_t([w'_i]\times[w'_j])| = \begin{cases} t+1, & 0 \le t \le w'_i - 1 \\ w'_i, & w'_i \le t \le w'_j - 1 \\ w'_j + w'_i - t - 1, & w'_j \le t \le w'_j + w'_i - 2 \\ 0, & \text{others} \end{cases}. \tag{10}$$

By setting $w'_i = w_i + 1$ and $w'_j = w_j - 1$ in Equation (10), we have Equation (11).

$$|\mathcal{C}_t([w'_i]\times[w'_j])| = \begin{cases} t+1, & 0 \le t \le w_i \\ w_i + 1, & w_i + 1 \le t \le w_j - 2 \\ w_j + w_i - t - 1, & w_j - 1 \le t \le w_j + w_i - 2 \\ 0, & \text{others} \end{cases}. \tag{11}$$

From Equation (9) and (11), we have

$$\begin{cases} |\mathcal{C}_t([w'_i]\times[w'_j])| > |\mathcal{C}_t([w_i]\times[w_j])|, & w_i \le t \le w_j - 2 \\ |\mathcal{C}_t([w'_i]\times[w'_j])| = |\mathcal{C}_t([w_i]\times[w_j])|, & \text{others} \end{cases}. \tag{12}$$

Let $s = \left\lfloor \frac{\sum_{k=0}^{l-1} w_k - 1}{2} \right\rfloor$, since $s = \left\lfloor \frac{\sum_{k=0,k\neq i,j}^{l-1} w_k - 1}{2} + \frac{w_j - 1 + w_i - 1}{2} \right\rfloor \geq w_i$, we have

$$
\begin{aligned}
&\left| \mathcal{C}_s \left([w_i'] \times [w_j'] \times \prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right| - \left| \mathcal{C}_s \left(\prod_{k=0}^{l-1} [w_k] \right) \right| \\
= &\sum_{t=0,t\neq w_i}^{s} \left(\left| \mathcal{C}_t \left([w_i'] \times [w_j'] \right) \right| \cdot \left| \mathcal{C}_{s-t} \left(\prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right| \right) \\
&- \sum_{t=0,t\neq w_i}^{s} \left(\left| \mathcal{C}_t \left([w_i] \times [w_j] \right) \right| \cdot \left| \mathcal{C}_{s-t} \left(\prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right| \right) \\
&+ \left| \mathcal{C}_{w_i} \left([w_i'] \times [w_j'] \right) \right| \cdot \left| \mathcal{C}_{s-w_i} \left(\prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right| \\
&- \left| \mathcal{C}_{w_i} \left([w_i] \times [w_j] \right) \right| \cdot \left| \mathcal{C}_{s-w_i} \left(\prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right|.
\end{aligned}
$$

According to Equation (12), we have $|\mathcal{C}_t([w_i'] \times [w_j'])| \geq |\mathcal{C}_t([w_i] \times [w_j])|$, which implies that

$$
\begin{aligned}
&\sum_{t=0,t\neq w_i}^{s} \left(\left| \mathcal{C}_t \left([w_i'] \times [w_j'] \right) \right| \cdot \left| \mathcal{C}_{s-t} \left(\prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right| \right) \geq \\
&\qquad \sum_{t=0,t\neq w_i}^{s} \left(\left| \mathcal{C}_t \left([w_i] \times [w_j] \right) \right| \cdot \left| \mathcal{C}_{s-t} \left(\prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right| \right).
\end{aligned}
$$

Since $|\mathcal{C}_t([w_i'] \times [w_j'])| > |\mathcal{C}_t([w_i] \times [w_j])|$ where $w_i \leq t \leq w_j - 2$, we have

$$
\begin{aligned}
&\left| \mathcal{C}_{w_i} \left([w_i'] \times [w_j'] \right) \right| \cdot \left| \mathcal{C}_{s-w_i} \left(\prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right| > \\
&\qquad \left| \mathcal{C}_{w_i} \left([w_i] \times [w_j] \right) \right| \cdot \left| \mathcal{C}_{s-w_i} \left(\prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right|.
\end{aligned}
$$

Therefore, we have

$$
\left| \mathcal{C}_s \left([w_i'] \times [w_j'] \times \prod_{k=0,k\neq i,j}^{l-1} [w_k] \right) \right| > \left| \mathcal{C}_s \left(\prod_{k=0}^{l-1} [w_k] \right) \right|.
$$

That is, the above operation strictly increases $|\mathcal{C}_s(\prod_{k=0}^{l-1}[w_k])|$ by at least 1. For any positive integers $w_0, \cdots, w_{l-1}$ with $w_0 + \cdots + w_{l-1} = A$, we repeatedly apply the above operation. This process will eventually terminates in at most

$$
\max_{\substack{(w_0,\cdots,w_{l-1})\in\mathbb{Z}_+ \\ w_0+\cdots+w_{l-1}=A}} \left\{ \left| \mathcal{C}_s \left(\prod_{k=0}^{l-1} [w_k] \right) \right| \right\} - \left| \mathcal{C}_s \left(\prod_{k=0}^{l-1} [w_k] \right) \right|
$$

steps, where $(w_0, \cdots, w_{l-1})$ is transformed into $(\hat{w}_0, \cdots, \hat{w}_{l-1})$ such that the difference between any two entries is at most 1. At this point, applying Lemma 1 completes the proof. □

M Proof of Lemma 11

Proof. For any positive integers $\log_2 w_0, \cdots, \log_2 w_{l-1}$ with $\sum_{i=0}^{l-1} \log_2 w_i = A$, if there exist $\log_2 w_i$, $\log_2 w_j$ where $0 \le i, j < l$ such that $\log_2 w_i \le \log_2 w_j - 2$, i.e., $w_i \le 4w_j$. Then we set $\log_2 w_i' = \log_2 w_i + 1$ and $\log_2 w_j' = \log_2 w_j - 1$, that is $w_i' = 2w_i$ and $w_j' = \frac{w_j}{2}$, i.e., $w_i' \le w_j'$. Let $M = \sum_{k=0,k\neq i,j}^{l-1}(w_k - 1)$, $s = \left\lfloor \frac{M+w_i-1+w_j-1}{2} \right\rfloor$ and $s' = \left\lfloor \frac{M+2w_i-1+\frac{w_j}{2}-1}{2} \right\rfloor$, we have $s = \lfloor \frac{M}{2} \rfloor + \frac{w_i}{2} + \frac{w_j}{2} - 1$, $s' = \lfloor \frac{M}{2} \rfloor + w_i + \frac{w_j}{4} - 1$.

According to Lemma 24, since $w_i \le w_j$ and $w_i' \le w_j'$, then for any integer t, we have Equation (9) and (10) . By setting $w_i' = 2w_i$ and $w_j' = \frac{w_j}{2}$ in Equation (10), we have Equation (13).

$$|\mathcal{C}_t([w_i'] \times [w_j'])| = \begin{cases} t+1, & 0 \le t \le 2w_i - 1 \\ 2w_i, & 2w_i \le t \le \frac{w_j}{2} - 1 \\ \frac{w_j}{2} + 2w_i - t - 1, & \frac{w_j}{2} - 1 \le t \le \frac{w_j}{2} + 2w_i - 2 \\ 0, & \text{others} \end{cases}. \tag{13}$$

According to the definition of $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$, we have

$$\begin{aligned} &\left| \mathcal{C}_{s'} \left([w_i'] \times [w_j'] \times \prod_{\substack{k=0 \\ k\neq i,j}}^{l-1} [w_k] \right) \right| - \left| \mathcal{C}_s \left(\prod_{k=0}^{l-1} [w_k] \right) \right| \\ = &\sum_{t=0}^{M} \left(\left| \mathcal{C}_{s'-t} \left([w_i'] \times [w_j'] \right) \right| \cdot \left| \mathcal{C}_t \left(\prod_{\substack{k=0 \\ k\neq i,j}}^{l-1} [w_k] \right) \right| \right) \\ &- \sum_{t=0}^{M} \left(\left| \mathcal{C}_{s-t} \left([w_i] \times [w_j] \right) \right| \cdot \left| \mathcal{C}_t \left(\prod_{\substack{k=0 \\ k\neq i,j}}^{l-1} [w_k] \right) \right| \right) \\ = &\sum_{t=0}^{M} \left| \mathcal{C}_t \left(\prod_{\substack{k=0 \\ k\neq i,j}}^{l-1} [w_k] \right) \right| \cdot \left(\left| \mathcal{C}_{s'-t} \left([w_i'] \times [w_j'] \right) \right| - \left| \mathcal{C}_{s-t} \left([w_i] \times [w_j] \right) \right| \right). \end{aligned}$$

Let $y = s - t$, we have

$$\begin{aligned} &\left| \mathcal{C}_{s'} \left([w_i'] \times [w_j'] \times \prod_{\substack{k=0 \\ k\neq i,j}}^{l-1} [w_k] \right) \right| - \left| \mathcal{C}_s \left(\prod_{k=0}^{l-1} [w_k] \right) \right| \\ = &\sum_{y} \left| \mathcal{C}_{s-y} \left(\prod_{\substack{k=0 \\ k\neq i,j}}^{l-1} [w_k] \right) \right| \cdot \left(\left| \mathcal{C}_{y-(\frac{w_j}{4} - \frac{w_i}{2})} \left([w_i'] \times [w_j'] \right) \right| - \left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| \right). \end{aligned}$$

According to Equation (13), for any integer t, we have Equation (14).

$$\left|\mathcal{C}_{t-(\frac{w_j}{4}-\frac{w_i}{2})}([w_i'] \times [w_j'])\right| = \begin{cases} t+1-(\frac{w_j}{4}-\frac{w_i}{2}), & -\frac{w_i}{2}+\frac{w_j}{4} \le t \le \frac{3w_i}{2}+\frac{w_j}{4}-1 \\ 2w_i, & \frac{3w_i}{2}+\frac{w_j}{4} \le t \le -\frac{w_i}{2}+\frac{3w_j}{4}-1 \\ \frac{3w_i}{2}+\frac{3w_j}{4}-t-1, & -\frac{w_i}{2}+\frac{w_j}{4} \le t \le \frac{3w_i}{2}+\frac{w_j}{4}-2 \\ 0, & \text{others} \end{cases}. \tag{14}$$

From Equation (9) and (14), we have

$$\begin{cases} \left|\mathcal{C}_{t-(\frac{w_j}{4}-\frac{w_i}{2})}([w_i'] \times [w_j'])\right| = \left|\mathcal{C}_t([w_i] \times [w_j])\right|, t = \frac{w_i}{2}+\frac{w_j}{4}-1 \text{ and } \frac{w_i}{2}+\frac{3w_j}{4}-1 \\ \left|\mathcal{C}_{t-(\frac{w_j}{4}-\frac{w_i}{2})}([w_i'] \times [w_j'])\right| > \left|\mathcal{C}_t([w_i] \times [w_j])\right|, \frac{w_i}{2}+\frac{w_j}{4} \le t \le \frac{w_i}{2}+\frac{3w_j}{4}-2 \\ \left|\mathcal{C}_{t-(\frac{w_j}{4}-\frac{w_i}{2})}([w_i'] \times [w_j'])\right| < \left|\mathcal{C}_t([w_i] \times [w_j])\right|, \\ \qquad 0 \le t \le \frac{w_i}{2}+\frac{w_j}{4}-2 \text{ and } \frac{w_i}{2}+\frac{3w_j}{4} \le t \le w_i+w_j-2 \\ \left|\mathcal{C}_{t-(\frac{w_j}{4}-\frac{w_i}{2})}([w_i'] \times [w_j'])\right| = \left|\mathcal{C}_t([w_i] \times [w_j])\right| = 0, \text{others} \end{cases}. \tag{15}$$

It can be computed by Equation (15) that

$$\begin{aligned} &\sum_t \left|\mathcal{C}_t\left([w_i] \times [w_j]\right)\right| \\ &= \frac{(w_i+1)\cdot w_i}{2} + w_i \cdot (w_j - w_i) + \frac{(w_i-1+1)\cdot(w_i-1)}{2} \\ &= \frac{w_i^2}{2} + \frac{w_i}{2} + w_j w_i - w_i^2 + \frac{w_i^2}{2} - \frac{w_i}{2} = w_j w_i, \end{aligned}$$

$$\begin{aligned} &\sum_t \left|\mathcal{C}_{t-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)}\left([w_i'] \times [w_j']\right)\right| \\ &= \frac{(2w_i+1)\cdot 2w_i}{2} + 2w_i \cdot \left(\frac{w_j}{2} - 2w_i\right) + \frac{(2w_i-1+1)\cdot(2w_i-1)}{2} \\ &= 2w_i^2 + w_i + w_j w_i - 4w_i^2 + 2w_i^2 - w_i = w_j w_i, \end{aligned}$$

which implies $\sum_t \left|\mathcal{C}_t\left([w_i] \times [w_j]\right)\right| = \sum_t \left|\mathcal{C}_{t-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)}\left([w_i'] \times [w_j']\right)\right|$. Therefore,

$$\begin{aligned} &\sum_{t=\frac{w_j}{4}+\frac{w_i}{2}-1}^{\frac{w_i}{2}+\frac{3w_j}{4}-1} \left(\left|\mathcal{C}_{t-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)}\left([w_i'] \times [w_j']\right)\right| - \left|\mathcal{C}_t\left([w_i] \times [w_j]\right)\right|\right) \\ = &\sum_{t=0}^{\frac{w_i}{2}+\frac{w_j}{4}-2} \left(\left|\mathcal{C}_t\left([w_i] \times [w_j]\right)\right| - \left|\mathcal{C}_{t-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)}\left([w_i'] \times [w_j']\right)\right|\right) \\ + &\sum_{t=\frac{w_i}{2}+\frac{3w_j}{4}}^{w_i+w_j-2} \left(\left|\mathcal{C}_t\left([w_i] \times [w_j]\right)\right| - \left|\mathcal{C}_{t-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)}\left([w_i'] \times [w_j']\right)\right|\right). \end{aligned} \tag{16}$$

According to Equation (15), we have

$$
\begin{aligned}
&\left|\mathcal{C}_{s'}\left([w_i'] \times [w_j'] \times \prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right| - \left|\mathcal{C}_s\left(\prod_{k=0}^{l-1}[w_k]\right)\right| \\
=& \sum_{y=0}^{\frac{w_i}{2}+\frac{w_j}{4}-2}\left|\mathcal{C}_{s-y}\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right| \cdot \left(\left|\mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)}\left([w_i'] \times [w_j']\right)\right| - \left|\mathcal{C}_y\left([w_i] \times [w_j]\right)\right|\right) \\
+& \sum_{y=\frac{w_i}{2}+\frac{w_j}{4}-1}^{\frac{w_i}{2}+\frac{3w_j}{4}-1}\left|\mathcal{C}_{s-y}\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right| \cdot \left(\left|\mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)}\left([w_i'] \times [w_j']\right)\right| - \left|\mathcal{C}_y\left([w_i] \times [w_j]\right)\right|\right) \\
+& \sum_{y=\frac{w_i}{2}+\frac{3w_j}{4}}^{w_i+w_j-2}\left|\mathcal{C}_{s-y}\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right| \cdot \left(\left|\mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)}\left([w_i'] \times [w_j']\right)\right| - \left|\mathcal{C}_y\left([w_i] \times [w_j]\right)\right|\right).
\end{aligned}
$$

We denote $\mathcal{C}_{left}$, $\mathcal{C}_{mid}$ and $\mathcal{C}_{right}$ as follows:

$$
\begin{cases}
\mathcal{C}_{left} = \max_{y\in\left[0,\frac{w_i}{2}+\frac{w_j}{4}-2\right]}\left\{\left|\mathcal{C}_{s-y}\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right|\right\} \\
\mathcal{C}_{mid} = \min_{y\in\left[\frac{w_i}{2}+\frac{w_j}{4}-1,\frac{3w_j}{4}+\frac{w_i}{2}-1\right]}\left\{\left|\mathcal{C}_{s-y}\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right|\right\} \\
\mathcal{C}_{right} = \max_{y\in\left[\frac{w_i}{2}+\frac{3w_j}{4},w_i+w_j-2\right]}\left\{\left|\mathcal{C}_{s-y}\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right|\right\}
\end{cases}.
$$

Let $t = s - y$, then we have

$$
\begin{cases}
\mathcal{C}_{left} = \max_{t\in\left[\lfloor\frac{M}{2}\rfloor+\frac{w_j}{4}+1,\lfloor\frac{M}{2}\rfloor+\frac{w_i}{2}+\frac{w_j}{2}-1\right]}\left\{\left|\mathcal{C}_t\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right|\right\} \\
\mathcal{C}_{mid} = \min_{t\in\left[\lfloor\frac{M}{2}\rfloor-\frac{w_j}{4},\lfloor\frac{M}{2}\rfloor+\frac{w_j}{4}\right]}\left\{\left|\mathcal{C}_t\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right|\right\} \\
\mathcal{C}_{right} = \max_{t\in\left[\lfloor\frac{M}{2}\rfloor-\frac{w_i}{2}-\frac{w_j}{2}+1,\lfloor\frac{M}{2}\rfloor-\frac{w_j}{4}-1\right]}\left\{\left|\mathcal{C}_t\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right|\right\}
\end{cases}.
$$

According to Lemma 17 and Theorem 3, we have

$$
\begin{cases}
\mathcal{C}_{mid} \geq \left|\mathcal{C}_{\lfloor\frac{M}{2}\rfloor-\frac{w_j}{4}}\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right| \\
\mathcal{C}_{left} \leq \left|\mathcal{C}_{\lceil\frac{M}{2}\rceil-\frac{w_j}{4}-1}\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right| \\
\mathcal{C}_{right} \leq \left|\mathcal{C}_{\lfloor\frac{M}{2}\rfloor-\frac{w_j}{4}-1}\left(\prod_{\substack{k=0\\k\neq i,j}}^{l-1}[w_k]\right)\right|
\end{cases}.
$$

Since $\lceil \frac{M}{2} \rceil - 1 \leq \lfloor \frac{M}{2} \rfloor$, $\mathcal{C}_{mid} \geq \mathcal{C}_{left}$ and $\mathcal{C}_{mid} \geq \mathcal{C}_{right}$. Therefore, according to Equation (16), we have

$$
\begin{aligned}
&\sum_{y=\frac{w_i}{2}+\frac{w_j}{4}-1}^{\frac{w_i}{2}+\frac{3w_j}{4}-1} \left| \mathcal{C}_{s-y} \left(\prod_{\substack{k=0 \\ k\neq i,j}}^{l-1} [w_k] \right) \right| \cdot \left(\left| \mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)} \left([w_i'] \times [w_j'] \right) \right| - \left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| \right) \\
> &\sum_{y=\frac{w_i}{2}+\frac{w_j}{4}-1}^{\frac{w_i}{2}+\frac{3w_j}{4}-1} \mathcal{C}_{mid} \cdot \left(\left| \mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)} \left([w_i'] \times [w_j'] \right) \right| - \left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| \right) \\
= \mathcal{C}_{mid} \cdot &\sum_{y=\frac{w_i}{2}+\frac{w_j}{4}-1}^{\frac{w_i}{2}+\frac{3w_j}{4}-1} \left(\left| \mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)} \left([w_i'] \times [w_j'] \right) \right| - \left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| \right) \\
= \mathcal{C}_{mid} \cdot &\sum_{y=0}^{\frac{w_i}{2}+\frac{w_j}{4}-2} \left(\left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| - \left| \mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)} \left([w_i'] \times [w_j'] \right) \right| \right) \\
+ \mathcal{C}_{mid} \cdot &\sum_{y=\frac{w_i}{2}+\frac{3w_j}{4}}^{w_i+w_j-2} \left(\left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| - \left| \mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)} \left([w_i'] \times [w_j'] \right) \right| \right) \\
\geq \mathcal{C}_{left} \cdot &\sum_{y=0}^{\frac{w_i}{2}+\frac{w_j}{4}-2} \left(\left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| - \left| \mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)} \left([w_i'] \times [w_j'] \right) \right| \right) \\
+ \mathcal{C}_{right} \cdot &\sum_{y=\frac{w_i}{2}+\frac{3w_j}{4}}^{w_i+w_j-2} \left(\left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| - \left| \mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)} \left([w_i'] \times [w_j'] \right) \right| \right) \\
> &\sum_{y=0}^{\frac{w_i}{2}+\frac{w_j}{4}-2} \left| \mathcal{C}_{s-y} \left(\prod_{\substack{k=0 \\ k\neq i,j}}^{l-1} [w_k] \right) \right| \cdot \left(\left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| - \left| \mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)} \left([w_i'] \times [w_j'] \right) \right| \right) \\
+ &\sum_{y=\frac{w_i}{2}+\frac{3w_j}{4}}^{w_i+w_j-2} \mathcal{C}_{s-y} \left(\prod_{\substack{k=0 \\ k\neq i,j}}^{l-1} [w_k] \right) \cdot \left(\left| \mathcal{C}_y \left([w_i] \times [w_j] \right) \right| - \left| \mathcal{C}_{y-\left(\frac{w_j}{4}-\frac{w_i}{2}\right)} \left([w_i'] \times [w_j'] \right) \right| \right),
\end{aligned}
$$

which implies $|\mathcal{C}_{s'}([w_i'] \times [w_j'] \times \prod_{k=0,k\neq i,j}^{l-1} [w_k])| > |\mathcal{C}_s(\prod_{k=0}^{l-1}[w_k])|$. That is, the above operation strictly increases $|\mathcal{C}_s(\prod_{k=0}^{l-1}[w_k])|$ by at least 1. For any positive integers $\log_2 w_0, \cdots, \log_2 w_{l-1}$ with $\log_2 w_0 + \cdots + \log_2 w_{l-1} = A$, we repeatedly apply the above operation. This process will eventually terminates in at most

$$
\max_{\substack{(\log_2 w_0, \cdots, \log_2 w_{l-1}) \in \mathbb{Z}_+^l \\ \log_2 w_0 + \cdots + \log_2 w_{l-1} = A}} \left| \mathcal{C}_{\left\lfloor \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rfloor} \left(\prod_{j=0}^{l-1} [w_j] \right) \right| - \left| \mathcal{C}_s \left(\prod_{k=0}^{l-1} [w_k] \right) \right|
$$

steps, where $(\log_2 w_0, \cdots, \log_2 w_{l-1})$ is transformed into $(\log_2 \hat{w}_0, \cdots, \log_2 \hat{w}_{l-1})$ such that the difference between any two entries is at most 1. At this point, applying Lemma 1 completes the proof. □

N Proof of Lemma 12

Proof. We only need to prove it in the case where $l = l' + 1$. Since $l = l' + 1$, there exist an integer $\log_2 w'_q \in \{\log_2 w'_0, \cdots, \log_2 w'_{l'-1}\}$, which can be split into $\log_2 w'_i$ and $\log_2 w'_j$ such that $\log_2 w'_i + \log_2 w'_j = \log_2 w'_q$. Therefore $w'_q = 2^{\log_2 w'_q} = 2^{\log_2 w'_i + \log_2 w'_j} = w'_i \cdot w'_j$, and $w'_q \geq w'_i + w'_j$. Let $M = \sum_{k=0,k\neq q}^{l'-1}(w'_k - 1)$, $s' = \left\lfloor \frac{M+w'_q-1}{2} \right\rfloor = \lfloor \frac{M-1}{2} \rfloor + \frac{w'_q}{2}$, $s'_2 = \left\lfloor \frac{M+w'_i-1+w'_j-1}{2} \right\rfloor = \lfloor \frac{M}{2} \rfloor + \frac{w'_i}{2} + \frac{w'_j}{2} - 1$, and $\delta = s' - s'_2$.

Let $w'_i \leq w'_j$, then for any integer t, according to Lemma 24, we have Equation (10). For any integer t, the definition of $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$ implies that,

$$|\mathcal{C}_t([w'_q])| = \begin{cases} 1, & 0 \leq t \leq w'_q - 1 \\ 0, & \text{others} \end{cases}. \tag{17}$$

According to the definition of $\mathcal{C}_s(\prod_{j=0}^{l-1}[w_j])$, we have

$$\begin{aligned}
&\left|\mathcal{C}_{s'_2}\left([w'_i]\times[w'_j]\times\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right| - \left|\mathcal{C}_{s'}\left(\prod_{k=0}^{l'-1}[w'_k]\right)\right| \\
=& \sum_{t=0}^{M}\left(\left|\mathcal{C}_{s'_2-t}\left([w'_i]\times[w'_j]\right)\right|\cdot\left|\mathcal{C}_t\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\right) \\
&\qquad - \sum_{t=0}^{M}\left(\left|\mathcal{C}_{s'-t}\left([w'_q]\right)\right|\cdot\left|\mathcal{C}_t\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\right) \\
=& \sum_{t=0}^{M}\left|\mathcal{C}_t\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\cdot\left(\left|\mathcal{C}_{s'_2-t}\left([w'_i]\times[w'_j]\right)\right| - \left|\mathcal{C}_{s'-t}\left([w'_q]\right)\right|\right) \\
=& \sum_{t=0}^{M}\left|\mathcal{C}_t\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\cdot\left(\left|\mathcal{C}_{s'-\delta-t}\left([w'_i]\times[w'_j]\right)\right| - \left|\mathcal{C}_{s'-t}\left([w'_q]\right)\right|\right).
\end{aligned}$$

Let $y = s' - t$, we have

$$\begin{aligned}
&\left|\mathcal{C}_{s'_2}\left([w'_i]\times[w'_j]\times\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right| - \left|\mathcal{C}_{s'}\left(\prod_{k=0}^{l'-1}[w'_k]\right)\right| \\
=& \sum_{y}\left|\mathcal{C}_{s'-y}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\cdot\left(\left|\mathcal{C}_{y-\delta}\left([w'_i]\times[w'_j]\right)\right| - \left|\mathcal{C}_{y}\left([w'_q]\right)\right|\right).
\end{aligned}$$

According to Equation (10), for any integer t, we have Equation (18).

$$|\mathcal{C}_{t-\delta}([w_i'] \times [w_j'])| = \begin{cases} t+1-\delta, & \delta \le t \le w_i' - 1 + \delta \\ w_i', & w_i' + \delta \le t \le w_j' - 1 + \delta \\ w_i' + w_j' - t - 1 + \delta, & w_j' + \delta \le t \le w_i' + w_j' - 2 + \delta \\ 0, & \text{others} \end{cases}. \tag{18}$$

From Equation (17) and (18), we have

$$\begin{cases} |\mathcal{C}_{t-\delta}([w_i'] \times [w_j'])| = |\mathcal{C}_t([w_q'])|, t = \delta \text{ and } w_i' + w_j' - 2 + \delta \\ |\mathcal{C}_{t-\delta}([w_i'] \times [w_j'])| > |\mathcal{C}_t([w_q'])|, 1 + \delta \le t \le w_i' + w_j' - 3 + \delta \\ |\mathcal{C}_{t-\delta}([w_i'] \times [w_j'])| < |\mathcal{C}_t([w_q'])|, 0 \le t \le \delta - 1 \text{ and } w_i' + w_j' - 1 + \delta \le t \le w_q' - 1 \\ |\mathcal{C}_{t-\delta}([w_i'] \times [w_j'])| = |\mathcal{C}_t([w_q'])| = 0, \text{others} \end{cases}. \tag{19}$$

It can be computed by Equation (19) that $\sum_t \left|\mathcal{C}_t\left([w_q']\right)\right| = w_q'$ and

$$\begin{aligned} & \sum_t \left|\mathcal{C}_{t-\delta}\left([w_i'] \times [w_j']\right)\right| \\ = & \frac{w_i' \cdot (w_i' + 1)}{2} + w_i' \cdot \left(w_j' - w_i'\right) + \frac{w_i' \cdot (w_i' - 1)}{2} \\ = & \frac{{w_i'}^2}{2} + \frac{w_i'}{2} + w_i' w_j' - {w_i'}^2 + \frac{{w_i'}^2}{2} - \frac{w_i'}{2} \\ = & w_i' w_j' = w_q', \end{aligned}$$

which implies $\sum_t \left|\mathcal{C}_t\left([w_q']\right)\right| = \sum_t \left|\mathcal{C}_{t-\delta}\left([w_i'] \times [w_j']\right)\right|$. Therefore, we have

$$\begin{aligned} & \sum_{t=1+\delta}^{w_i'+w_j'-3+\delta} \left(\left|\mathcal{C}_{t-\delta}\left([w_i'] \times [w_j']\right)\right| - \left|\mathcal{C}_t\left([w_q']\right)\right|\right) \\ = & \sum_{t=0}^{\delta} \left(\left|\mathcal{C}_t\left([w_q']\right)\right| - \left|\mathcal{C}_{t-\delta}\left([w_i'] \times [w_j']\right)\right|\right) \\ + & \sum_{t=w_i'+w_j'-2+\delta}^{w_q'-1} \left(\left|\mathcal{C}_t\left([w_q']\right)\right| - \left|\mathcal{C}_{t-\delta}\left([w_i'] \times [w_j']\right)\right|\right). \end{aligned} \tag{20}$$

According to Equation (19), we have

$$
\begin{aligned}
&\left|\mathcal{C}_{s'_2}\left([w'_i]\times[w'_j]\times\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|-\left|\mathcal{C}_{s'}\left(\prod_{k=0}^{l'-1}[w'_k]\right)\right| \\
=&\sum_{y=0}^{\delta}\left|\mathcal{C}_{s'-y}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\cdot\left(\left|\mathcal{C}_{y-\delta}\left([w'_i]\times[w'_j]\right)\right|-\left|\mathcal{C}_{y}\left([w'_q]\right)\right|\right) \\
+&\sum_{y=1+\delta}^{w'_i+w'_j-3+\delta}\left|\mathcal{C}_{s'-y}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\cdot\left(\left|\mathcal{C}_{y-\delta}\left([w'_i]\times[w'_j]\right)\right|-\left|\mathcal{C}_{y}\left([w'_q]\right)\right|\right) \\
+&\sum_{y=w'_i+w'_j-2+\delta}^{w'_q-1}\left|\mathcal{C}_{s'-y}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\cdot\left(\left|\mathcal{C}_{y-\delta}\left([w'_i]\times[w'_j]\right)\right|-\left|\mathcal{C}_{y}\left([w'_q]\right)\right|\right).
\end{aligned}
$$

We denote $\mathcal{C}_{left}$, $\mathcal{C}_{mid}$ and $\mathcal{C}_{right}$ as follows:

$$
\begin{cases}
\mathcal{C}_{left}=\max_{y\in[0,\delta]}\left\{\left|\mathcal{C}_{s'-y}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\right\}\\
\mathcal{C}_{mid}=\min_{y\in\left[1+\delta,w'_i+w'_j-3+\delta\right]}\left\{\left|\mathcal{C}_{s'-y}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\right\}\\
\mathcal{C}_{right}=\max_{y\in\left[w'_i+w'_j-2+\delta,w'_q-1\right]}\left\{\left|\mathcal{C}_{s'-y}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\right\}
\end{cases}.
$$

Let $t=s'-y$, then we have

$$
\begin{cases}
\mathcal{C}_{left}=\max\limits_{t\in\left[\lfloor\frac{M}{2}\rfloor+\frac{w'_i}{2}+\frac{w'_j}{2}-1,\lfloor\frac{M-1}{2}\rfloor+\frac{w'_q}{2}\right]}\left\{\left|\mathcal{C}_{t}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\right\}\\
\mathcal{C}_{mid}=\min\limits_{t\in\left[\lfloor\frac{M}{2}\rfloor-\frac{w'_i}{2}-\frac{w'_j}{2}+2,\lfloor\frac{M}{2}\rfloor+\frac{w'_i}{2}+\frac{w'_j}{2}-2\right]}\left\{\left|\mathcal{C}_{t}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\right\}\\
\mathcal{C}_{right}=\max\limits_{t\in\left[\lfloor\frac{M-1}{2}\rfloor-\frac{w'_q}{2}+1,\lfloor\frac{M}{2}\rfloor-\frac{w'_i}{2}-\frac{w'_j}{2}+1\right]}\left\{\left|\mathcal{C}_{t}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\right\}
\end{cases}.
$$

According to Lemma 17 and Theorem 3, we have

$$
\begin{cases}
\mathcal{C}_{mid}\geq\left|\mathcal{C}_{\lfloor\frac{M}{2}\rfloor-\frac{w'_i}{2}-\frac{w'_j}{2}+2}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\\
\mathcal{C}_{left}\leq\left|\mathcal{C}_{\lceil\frac{M}{2}\rceil-\frac{w'_i}{2}\frac{w'_j}{2}+1}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|\\
\mathcal{C}_{right}\leq\left|\mathcal{C}_{\lfloor\frac{M}{2}\rfloor-\frac{w'_i}{2}-\frac{w'_j}{2}+1}\left(\prod_{\substack{k=0\\k\neq q}}^{l'-1}[w'_k]\right)\right|
\end{cases}.
$$

Since $\lceil \frac{M}{2} \rceil - 1 \leq \lfloor \frac{M}{2} \rfloor$, $\mathcal{C}_{mid} \geq \mathcal{C}_{left}$ and $\mathcal{C}_{mid} \geq \mathcal{C}_{right}$. Therefore, according to Equation (20), we have

$$
\begin{aligned}
&\sum_{y=1+\delta}^{w_i'+w_j'-3+\delta} \left| \mathcal{C}_{s'-y} \left(\prod_{\substack{k=0 \\ k\neq q}}^{l'-1} [w_k'] \right) \right| \cdot \left(\left| \mathcal{C}_{y-\delta} \left([w_i'] \times [w_j'] \right) \right| - \left| \mathcal{C}_y \left([w_q'] \right) \right| \right) \\
> &\sum_{y=1+\delta}^{w_i'+w_j'-3+\delta} \mathcal{C}_{mid} \cdot \left(\left| \mathcal{C}_{y-\delta} \left([w_i'] \times [w_j'] \right) \right| - \left| \mathcal{C}_y \left([w_q'] \right) \right| \right) \\
= &\, \mathcal{C}_{mid} \cdot \sum_{y=1+\delta}^{w_i'+w_j'-3+\delta} \left(\left| \mathcal{C}_{y-\delta} \left([w_i'] \times [w_j'] \right) \right| - \left| \mathcal{C}_y \left([w_q'] \right) \right| \right) \\
= &\, \mathcal{C}_{mid} \cdot \sum_{y=0}^{\delta} \left(\left| \mathcal{C}_y \left([w_q'] \right) \right| - \left| \mathcal{C}_{y-\delta} \left([w_i'] \times [w_j'] \right) \right| \right) \\
&+ \mathcal{C}_{mid} \cdot \sum_{y=w_i'+w_j'-2+\delta}^{w_q'-1} \left(\left| \mathcal{C}_y \left([w_q'] \right) \right| - \left| \mathcal{C}_{y-\delta} \left([w_i'] \times [w_j'] \right) \right| \right) \\
\geq &\, \mathcal{C}_{left} \cdot \sum_{y=0}^{\delta} \left(\left| \mathcal{C}_y \left([w_q'] \right) \right| - \left| \mathcal{C}_{y-\delta} \left([w_i'] \times [w_j'] \right) \right| \right) \\
&+ \mathcal{C}_{right} \cdot \sum_{y=w_i'+w_j'-2+\delta}^{w_q'-1} \left(\left| \mathcal{C}_y \left([w_q'] \right) \right| - \left| \mathcal{C}_{y-\delta} \left([w_i'] \times [w_j'] \right) \right| \right) \\
> &\sum_{y=0}^{\delta} \left| \mathcal{C}_{s'-y} \left(\prod_{\substack{k=0 \\ k\neq q}}^{l'-1} [w_k'] \right) \right| \cdot \left(\left| \mathcal{C}_y \left([w_q'] \right) \right| - \left| \mathcal{C}_{y-\delta} \left([w_i'] \times [w_j'] \right) \right| \right) \\
&+ \sum_{y=w_i'+w_j'-2+\delta}^{w_q'-1} \mathcal{C}_{s'-y} \left(\prod_{\substack{k=0 \\ k\neq q}}^{l'-1} [w_k'] \right) \cdot \left(\left| \mathcal{C}_y \left([w_q'] \right) \right| - \left| \mathcal{C}_{y-\delta} \left([w_i'] \times [w_j'] \right) \right| \right),
\end{aligned}
$$

which implies $|\mathcal{C}_{s_2'}([w_i'] \times [w_j'] \times \prod_{k=0,k\neq q}^{l'-1} [w_k'])| > |\mathcal{C}_{s'}(\prod_{k=0}^{l'-1} [w_k'])|$. Since $l = l'+1$, according to Lemma 11, we have

$$
\left| \mathcal{C}_{\left\lfloor \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rfloor} \left(\prod_{j=0}^{l-1} [w_j] \right) \right| \geq \left| \mathcal{C}_{s_2'} \left([w_i'] \times [w_j'] \times \prod_{\substack{k=0 \\ k\neq q}}^{l'-1} [w_k'] \right) \right|,
$$

where $\log_2 w_0 = \cdots = \log_2 w_{l-k-1} = \lfloor \frac{A}{l} \rfloor$, $\log_2 w_{l-k} = \cdots = \log_2 w_{l-1} = \lceil \frac{A}{l} \rceil$, $k = A - l \cdot \lfloor \frac{A}{l} \rfloor$. Therefore, we have

$$
\left| \mathcal{C}_{\left\lfloor \frac{\sum_{i=0}^{l-1}(w_i-1)}{2} \right\rfloor} \left(\prod_{j=0}^{l-1} [w_j] \right) \right| > \left| \mathcal{C}_{\left\lfloor \frac{\sum_{i=0}^{l'-1}(w_i'-1)}{2} \right\rfloor} \left(\prod_{j=0}^{l'-1} [w_j'] \right) \right|.
$$

□

O Proof of Lemma 13

Proof. According to Lemma 2,

$$\left\langle \frac{h}{d} \right\rangle = \min_{\substack{(x_0,\cdots,x_{d-1})\in\mathbb{Z}_+^{l_0} \\ x_0+\cdots+x_{d-1}=h}} \sum_{i=0}^{d-1} x_i.$$

We only need to show that for positive integers h and h' with $h' = h+1$,

$$\left\langle \frac{h}{d} \right\rangle < \left\langle \frac{h'}{d} \right\rangle = \min_{\substack{(x'_0,\cdots,x'_{d-1})\in\mathbb{Z}_+^{l_0} \\ x'_0+\cdots+x'_{d-1}=h'}} \sum_{i=0}^{d-1} x'_i.$$

Lemma 2 implies $\left\langle \frac{h}{d} \right\rangle = (d-k)\cdot 2^{\lfloor \frac{h}{d} \rfloor} + k\cdot 2^{\lceil \frac{h}{d} \rceil}$, where $x_0 = \cdots = x_{d-k-1} = \left\lfloor \frac{h}{d} \right\rfloor$, $x_{d-k} = \cdots = x_{d-1} = \left\lceil \frac{h}{d} \right\rceil$, $k = h - d\cdot\left\lfloor \frac{h}{d} \right\rfloor$, and $\left\langle \frac{h'}{d} \right\rangle = (d-k')\cdot 2^{\lfloor \frac{h'}{d} \rfloor} + k'\cdot 2^{\lceil \frac{h'}{d} \rceil}$, where $x'_0 = \cdots = x'_{d-k'-1} = \left\lfloor \frac{h'}{d} \right\rfloor$, $x'_{d-k'} = \cdots = x'_{d-1} = \left\lceil \frac{h'}{d} \right\rceil$, $k' = h' - d\cdot\left\lfloor \frac{h'}{d} \right\rfloor$. If d divides h,

$$\left\langle \frac{h}{d} \right\rangle = d\cdot 2^{\frac{h}{d}} < d\cdot 2^{\frac{h+1}{d}} \le \left\langle \frac{h'}{d} \right\rangle.$$

If d does not divide h, we have $\left\lfloor \frac{h}{d} \right\rfloor + 1 = \left\lceil \frac{h}{d} \right\rceil$, then if d does not divide h', we have $\left\lfloor \frac{h'}{d} \right\rfloor = \left\lfloor \frac{h}{d} \right\rfloor$ and $\left\lceil \frac{h'}{d} \right\rceil = \left\lceil \frac{h}{d} \right\rceil$, which implies $k' = k+1$. Therefore,

$$\left\langle \frac{h'}{d} \right\rangle - \left\langle \frac{h}{d} \right\rangle = 2^{\lfloor \frac{h}{d} \rfloor},$$

that is $\left\langle \frac{h'}{d} \right\rangle > \left\langle \frac{h}{d} \right\rangle$. If d divides h', then $k' = 0$ and $\left\lfloor \frac{h}{d} \right\rfloor + 1 = \left\lceil \frac{h'}{d} \right\rceil$, which implies $\left\langle \frac{h'}{d} \right\rangle = d\cdot 2^{\lfloor \frac{h}{d} \rfloor + 1}$ and $k = l-1$. Therefore,

$$\left\langle \frac{h'}{d} \right\rangle - \left\langle \frac{h}{d} \right\rangle = 2^{\lfloor \frac{h}{d} \rfloor},$$

that is $\left\langle \frac{h'}{d} \right\rangle > \left\langle \frac{h}{d} \right\rangle$. □

P Proof of Lemma 14

Proof. We only need to show that for any positive integer d and $d' = d+1$, $\left\langle \frac{h}{d} \right\rangle \geq \left\langle \frac{h}{d'} \right\rangle$. Lemma 2 implies $\left\langle \frac{h}{d} \right\rangle = (d-k) \cdot 2^{\lfloor \frac{h}{d} \rfloor} + k \cdot 2^{\lceil \frac{h}{d} \rceil}$ and $\left\langle \frac{h}{d'} \right\rangle = (d'-k') \cdot 2^{\lfloor \frac{h}{d'} \rfloor} + k' \cdot 2^{\lceil \frac{h}{d'} \rceil}$, where $k = h - d \cdot \left\lfloor \frac{h}{d} \right\rfloor$ and $k' = h - d' \cdot \left\lfloor \frac{h}{d'} \right\rfloor$. Therefore, we have $\left\langle \frac{h}{d} \right\rangle = (d+k) \cdot 2^{\lfloor \frac{h}{d} \rfloor}$ and $\left\langle \frac{h}{d'} \right\rangle = (d'+k') \cdot 2^{\lfloor \frac{h}{d'} \rfloor}$. Let $a = \left\lfloor \frac{h}{d} \right\rfloor$ and $a' = \left\lfloor \frac{h}{d'} \right\rfloor$, then $a' = a$ or $a' \leq a-1$. If $a' = a$, we have

$$\begin{aligned}
\left\langle \frac{h}{d} \right\rangle - \left\langle \frac{h}{d'} \right\rangle &= (d+k) \cdot 2^a - (d+1+k') \cdot 2^a \\
&= (d+h-da) \cdot 2^a - (d+1+h-(d+1)a) \cdot 2^a \\
&= 2^a \cdot (d+h-da-d-1-h+da+a) \\
&= 2^a \cdot (a-1) \geq 0,
\end{aligned}$$

If $a' = a-1$, then $d \cdot a = (d+1) \cdot (a-1)$, which implies $k' = h-(d+1) \cdot (a-1) = h - da = 0$. Therefore, we have

$$\begin{aligned}
\left\langle \frac{h}{d} \right\rangle - \left\langle \frac{h}{d'} \right\rangle &= d \cdot 2^a - (d+1) \cdot 2^{a-1} \\
&= (2d-d-1) \cdot 2^{a-1} \\
&= (d-1) \cdot 2^{a-1} \geq 0,
\end{aligned}$$

If $a' \leq a-2$, then we have

$$\begin{aligned}
\left\langle \frac{h}{d} \right\rangle - \left\langle \frac{h}{d'} \right\rangle &= (d+k) \cdot 2^a - (d+1+k') \cdot 2^{a'} \\
&\geq 4 \cdot (d+k) \cdot 2^{a'} - (d+1+k') \cdot 2^{a'} \\
&= (4d+4k-d-1-k') \cdot 2^{a'} \\
&= (3d-1-k'+4k) \cdot 2^{a'}.
\end{aligned}$$

Since $0 \leq k' \leq d$, we have $(3d-1-k'+4k) \cdot 2^{a'} \geq 0$, which implies $\left\langle \frac{h}{d} \right\rangle \geq \left\langle \frac{h}{d'} \right\rangle$. □

Q Security of Inhomogeneous WOTS-TW

This section formalizes the security of the inhomogeneous one-time signature underlying Construction 1. We use its address-separated WOTS-TW counterpart, with the public seed, tweakable-hash collection Th_λ, and domain separation described in Section 3.

Q.1 Inhomogeneous WOTS-TW

Let $\mathbf{w} = (w_0, \ldots, w_{\ell-1}) \in \mathbb{Z}_+^\ell$, define

$$W_\Sigma = \sum_{i=0}^{\ell-1} w_i, \qquad w_{\max} = \max_{0 \le i < \ell} w_i.$$

Here, W_Σ is the inhomogeneous counterpart of the term ℓw in the uniform-chain proof. Let $\mathcal{M}$ be any finite message space and let

$$\mathsf{Encode} : \mathcal{M} \longrightarrow \prod_{i=0}^{\ell-1} [w_i].$$

The resulting scheme is the WOTS-TW version of Construction 1, with chain i ending at position $w_i - 1$ and with distinct address-derived tweaks for every instance, chain, and chain step. Construction 1 is the special case $\mathcal{M} = \mathbb{F}_2^{8n}$.

We use the multi-instance D-EU-naCMA experiment of [HK22] unchanged, including its shared-public-seed, distinct-address-prefix, collection-oracle, and query restrictions.

For a primitive $\mathcal{P}$ and a security property X, let

$$\mathrm{InSec}^{\mathsf{X}}_{\mathcal{P}}(t, q)$$

denote the maximum success probability of an adversary running in time t in the corresponding q-target security experiment.

Theorem 5 (Security of inhomogeneous WOTS-TW). *Suppose that* Encode *is injective and that* $\mathsf{Encode}(\mathcal{M})$ *is an antichain with respect to the coordinatewise partial order, and suppose that* $w_{\max}$ *is polynomially bounded in the security parameter. Assume that* PRF *is a secure pseudorandom function and that* F*, as a member of the shared-seed tweakable-hash collection, satisfies the* SM-TCR*,* SM-PRE*, and* SM-UD *properties required in the corrected* WOTS-TW *security proof [HK22]. Then, for every* D*-instance adversary running in time* t*,*

$$\begin{aligned}
&\mathrm{InSec}^{D\text{-EU-naCMA}}_{\texttt{WOTS-TW}_{\mathbf{w}}}(t, D) \\
&\quad \le \mathrm{InSec}^{\mathsf{PRF}}_{\mathsf{PRF}}(\widetilde{t}, D\ell) + w_{\max}\, \mathrm{InSec}^{\mathsf{SM\text{-}UD}}_{F \in \mathsf{Th}_\lambda}(\widetilde{t}, D\ell) \\
&\qquad + \mathrm{InSec}^{\mathsf{SM\text{-}TCR}}_{F \in \mathsf{Th}_\lambda}(\widetilde{t}, DW_\Sigma) + \mathrm{InSec}^{\mathsf{SM\text{-}PRE}}_{F \in \mathsf{Th}_\lambda}(\widetilde{t}, D\ell),
\end{aligned}$$

where

$$\widetilde{t} = t + DW_\Sigma$$

when time is measured in primitive evaluations; this upper-bounds the simulation of all inhomogeneous hash chains and secret-value generations. If the initial chain values are sampled independently and uniformly, as in Construction 1, the PRF term is omitted.

Proof. We use the multi-instance `WOTS-TW` games from [HK22]. The cryptographic reductions remain unchanged; only the numbers of chain positions and chain-depth hybrids have to be adjusted.

First, the $D\ell$ chain secret values generated by PRF are replaced by independently and uniformly sampled values. The difference between the two games is bounded by

$$\mathrm{InSec}^{\mathsf{PRF}}_{\mathsf{PRF}}(\widetilde{t}, D\ell).$$

When the initial chain values are already sampled independently and uniformly, this game transition is unnecessary.

Next, apply the chain-depth hybrids in the order used in the uniform-chain proof. There are at most $w_{\max}$ hybrid levels, and each level contains at most $D\ell$ targets; chains that have already ended simply do not participate. The total undetectability contribution is bounded by

$$w_{\max}\ \mathrm{InSec}^{\mathsf{SM\text{-}UD}}_{F\in\mathsf{Th}_\lambda}(\widetilde{t}, D\ell).$$

This is precisely the inhomogeneous counterpart of the factor w in the uniform-chain proof.

The D instances contain $D\sum_i(w_i - 1) < DW_\Sigma$ hash-chain steps. Consequently, the collision-resistance transition that uses at most $D\ell w$ targets in the uniform-chain proof now uses at most DW_Σ targets. Its contribution is bounded by

$$\mathrm{InSec}^{\mathsf{SM\text{-}TCR}}_{F\in\mathsf{Th}_\lambda}(\widetilde{t}, DW_\Sigma).$$

It remains to consider the final preimage reduction. Let m be the message signed in the target instance and let $m^* \neq m$ be the message in a successful forgery. Write

$$\mathbf{x} = \mathsf{Encode}(m), \qquad \mathbf{x}^* = \mathsf{Encode}(m^*).$$

Since Encode is injective, $\mathbf{x}^* \neq \mathbf{x}$. Moreover, because $\mathsf{Encode}(\mathcal{M})$ is an antichain, it cannot hold that $x_i^* \geq x_i$ for every $i \in [\ell]$; otherwise, $\mathbf{x}$ and $\mathbf{x}^*$ would be comparable under the coordinatewise order. Hence, there exists an index $i \in [\ell]$ such that $x_i^* < x_i$.

The legitimate signature reveals the chain value at position x_i, whereas a valid forgery requires the value at the earlier position x_i^*. Therefore, conditioned on the preceding games, a successful forgery yields an $\mathsf{SM\text{-}PRE}$ solution for one of the $D\ell$ chains. This contribution is bounded by

$$\mathrm{InSec}^{\mathsf{SM\text{-}PRE}}_{F\in\mathsf{Th}_\lambda}(\widetilde{t}, D\ell).$$

Adding the differences between the games proves the claimed bound.

Q.2 Application to the Three Encodings

The deterministic encoding of the inhomogeneous $\texttt{WOTS}^{+}$ construction is injective, and Lemma 3 proves that its image is an antichain. Therefore, Theorem 5 applies directly.

For inhomogeneous $\texttt{WOTS}^{\alpha}$, the encoding $\tau^{s}_{\mathbf{w}}$ is injective and maps the message space into

$$\mathcal{C}_s\left(\prod_{i=0}^{\ell-1}[w_i]\right).$$

By Lemma 6, this constant-sum set is an antichain. Hence, Theorem 5 also applies directly, without introducing an additional security term.

The counter-based preprocessing of $\texttt{WOTS}^{\mathsf{C}}$ is different: it is not an injective deterministic map from roots to codewords. Let $\mathsf{Good}_{\mathbf{w},z_b}$ be the predicate defined in Section 3, and let

$$\mathcal{G}_{\mathbf{w},z_b} = \{d \in \mathbb{F}_2^{8n} : \mathsf{Good}_{\mathbf{w},z_b}(d) = 1\}.$$

For $d = 0^{z_b} \parallel d_1 \in \mathcal{G}_{\mathbf{w},z_b}$, the underlying $\texttt{WOTS-TW}$ message encoding is

$$d \longmapsto \texttt{IntSeq}_{(\log_2 w_0,\ldots,\log_2 w_{\ell-1})}(d_1).$$

This map is injective because the prescribed z_b-bit prefix is fixed and the mixed-radix parsing of d_1 is bijective. Its image lies in the constant-sum set specified by $\mathsf{Good}_{\mathbf{w},z_b}$ and is therefore an antichain by Lemma 6. Thus Theorem 5 applies to the underlying $\texttt{WOTS-TW}$ scheme with message space $\mathcal{G}_{\mathbf{w},z_b}$.

Theorem 6 (Security of inhomogeneous $\texttt{WOTS}^{\mathsf{C}}$). *Conditioned on successful counter search, the insecurity of the D-instance inhomogeneous $\texttt{WOTS}^{\mathsf{C}}$ construction satisfies*

$$\begin{aligned}&\mathrm{InSec}^{D\text{-EU-naCMA}}_{\texttt{WOTS}^{\mathsf{C}}_{\mathbf{w}}}(t, D)\\ &\quad\le \mathrm{InSec}^{\mathsf{S\text{-}TCR}(\mathsf{Good}_{\mathbf{w},z_b})}_{\mathsf{H}_{\mathsf{Root}}\in\mathsf{Th}_\lambda}(\widehat{t}, D) + \mathrm{InSec}^{D\text{-EU-naCMA}}_{\texttt{WOTS-TW}_{\mathbf{w}}}(\widehat{t}, D),\end{aligned}$$

where the second term uses message space $\mathcal{G}_{\mathbf{w},z_b}$ and is bounded by Theorem 5, and $\widehat{t}$ includes the chain simulation and the prescribed counter-search work. The first term is the collection version of the special-target-collision experiment of [HKRY23], under the same public seed and address-separated tweaks.

Proof. This is the D-instance version of the $\texttt{WOTS}^{\mathsf{C}}$ reduction of [HKRY23]. In the forged instance the root M^* is fresh. If its good digest d^* equals the signed target digest d, then (M^*, γ^*) solves $\mathsf{S\text{-}TCR}(\mathsf{Good}_{\mathbf{w},z_b})$ because $M^* \neq M$ and the tweak is unchanged. If $d^* \neq d$, successful verification implies $d^* \in \mathcal{G}_{\mathbf{w},z_b}$, and the chain values give a forgery against the underlying inhomogeneous $\texttt{WOTS-TW}$ instance. These cases are exhaustive. The probability that a bounded counter search finds no good value is a correctness failure, not a third unforgeability case, and must be reported separately for a concrete implementation.

R EU-CMA Security of CEDRUS$^+$

We now place Theorem 5 inside the SPHINCS$^+$ reduction of Hülsing and Kudinov [HK22]. We use their games and security notions, the only changes are the inhomogeneous chain counts and the unequal layer heights.

R.1 Instance Counts for Unequal Layers

Let layer 0 be the lowest layer, let $h_i > 0$ be the height in layer i, and set

$$s_0 = 0, \qquad s_i = \sum_{r=0}^{i-1} h_r \quad (1 \le i \le d), \qquad s_d = h.$$

The number of WOTS instances in layer i is 2^{h-s_i}, so

$$D_{\mathsf{HT}} = \sum_{i=0}^{d-1} 2^{h-s_i} < 2^{h+1}. \tag{21}$$

The number of internal Merkle nodes across all hypertree layers telescopes:

$$\sum_{i=0}^{d-1} \left(2^{h-s_i} - 2^{h-s_{i+1}}\right) = 2^h - 1. \tag{22}$$

Finally, the 2^{h-h_0} bottom-layer trees each contain 2^{h_0} leaves, hence the total number of FORS instances is

$$2^{h-h_0} 2^{h_0} = 2^h. \tag{23}$$

Thus unequal layer heights do not increase any of the conservative target bounds used for a regular hypertree of total height h.

R.2 Full Reduction

Use the primitive notation of [HK22]: F is the n-to-n tweakable function used for chains and secret-key nodes, H is the $2n$-to-n tree hash, Th_k compresses a k-element FORS public key, Th_ℓ compresses an ℓ-element WOTS public key, and $\mathsf{H}_{\mathsf{msg}}$, PRF, and $\mathsf{PRF}_{\mathsf{msg}}$ have their usual SPHINCS$^+$ roles. The tweakable functions are understood as members of the shared-public-seed, distinct-tweak collection used in that proof.

Theorem 7 (Security of CEDRUS$^+$). *Let $(n, h, d, m, k, t, \mathbf{w})$ be a CEDRUS$^+$ parameter set with $t = 2^a$ and $\ell = |\mathbf{w}|$. Let q_s be the number of signing queries, and suppose the address encoding is injective and separates all primitive domains. Under the same assumptions as the SPHINCS$^+$ proof [HK22], and under*

the hypotheses of Theorem 5, for any adversary time bound $\xi_{\mathcal{A}}$,

$$\begin{aligned}
&\mathrm{InSec}^{\text{EU-CMA}(q_s)}_{\textit{CEDRUS}^+}(\xi_{\mathcal{A}}) \\
&\quad \le \mathrm{InSec}^{\mathsf{PRF}}_{\mathsf{PRF}}(\xi', q_1) + \mathrm{InSec}^{\mathsf{PRF}}_{\mathsf{PRF}_{\mathsf{msg}}}(\xi', q_s) + \mathrm{InSec}^{\mathsf{ITSR}}_{\mathsf{H}_{\mathsf{msg}}}(\xi', q_s) \\
&\qquad + w_{\max}\,\mathrm{InSec}^{\mathsf{SM\text{-}UD}}_{F}(\xi', q_2) + \mathrm{InSec}^{\mathsf{SM\text{-}TCR}}_{F}(\xi', q_3 + q_7) + \mathrm{InSec}^{\mathsf{SM\text{-}PRE}}_{F}(\xi', q_2) \\
&\qquad + \mathrm{InSec}^{\mathsf{SM\text{-}TCR}}_{H}(\xi', q_4) + \mathrm{InSec}^{\mathsf{SM\text{-}TCR}}_{\mathsf{Th}_k}(\xi', q_5) + \mathrm{InSec}^{\mathsf{SM\text{-}TCR}}_{\mathsf{Th}_\ell}(\xi', q_6) \\
&\qquad + 3\,\mathrm{InSec}^{\mathsf{SM\text{-}TCR}}_{F}(\xi', q_8) + \mathrm{InSec}^{\mathsf{SM\text{-}DSPR}}_{F}(\xi', q_8),
\end{aligned}$$

where ξ' is a common upper bound on the running times of the derived reductions, including their simulation overhead, and the conservative target bounds may be chosen as

$$\begin{aligned}
&q_1 < 2^{h+1}(kt+\ell),\ q_2 < 2^{h+1}\ell,\ q_3 < 2^{h+1}W_\Sigma, \\
&q_4 < 2^{h+1}k(2t), \quad q_5 \le 2^h, \quad q_6 < 2^{h+1}, \\
&q_7 < 2^{h+1}kt, \quad\ \ q_8 \le 2^h kt.
\end{aligned} \tag{24}$$

Proof. Run the game sequence of the corrected SPHINCS$^+$ proof [HK22]. The $\mathsf{PRF}_{\mathsf{msg}}$ and ITSR transitions are unchanged. The first global PRF transition covers exactly $2^h kt + D_{\mathsf{HT}}\ell$ secret values: $2^h kt$ from the FORS instances and $D_{\mathsf{HT}}\ell$ from the WOTS instances. This is bounded by q_1, so the PRF term of Theorem 5 is already included rather than added a second time.

For the remaining one-time-signature transitions, apply that theorem with $D = D_{\mathsf{HT}}$. There are exactly $D_{\mathsf{HT}}\ell$ WOTS chains and fewer than $D_{\mathsf{HT}}W_\Sigma$ chain positions; by Equation (21), these are bounded by q_2 and q_3, respectively. The depth hybrid has at most $w_{\max}$ levels, which gives the displayed $w_{\max}$SM-UD term.

Equations (22) and (23) bound all remaining instances. The Merkle-tree, FORS, and public-key-compression transitions therefore use no more targets than in the source proof. They give the H, Th_k, Th_ℓ, and the q_7 part of the F terms in exactly the same games.

The remaining same-leaf branch is the T-openPRE branch of the corrected proof. Reducing that branch gives three SM-TCR terms and one SM-DSPR term on q_8 targets. It is not another SM-PRE term. Summing the game differences yields the claimed bound.

R.3 Adaptation to CEDRUS$^\alpha$ and CEDRUS$^+$C

For CEDRUS$^\alpha$, retain the SPHINCS$^+$ game sequence of [HK22], but replace its FORS-specific ITSR step by the FORC-specific experiment of [ZCY22]. The inhomogeneous WOTS$^\alpha$ component is covered directly by Theorem 5, with W_Σ and $w_{\max}$ replacing ℓw and w. Unequal layer heights still give $D_{\mathsf{HT}} < 2^{h+1}$ and exactly 2^h FORC instances. Each FORC position remains indexed by (J, L) with the coverage relation defined in that work, and its chain length w' is retained in the FORC-related F-target counts (the analogues of q_7 and q_8). The residual T-openPRE branch still reduces to three SM-TCR terms and one SM-DSPR term.

For $\texttt{CEDRUS}^{+}\texttt{C}$, apply Theorem 5.2 of the $\texttt{SPHINCS}^{+}\texttt{C}$ reduction in [HKRY23]. The inhomogeneous $\texttt{WOTS}^{\texttt{C}}$ substitution is exactly Theorem 6. At hypertree level, the corresponding $\texttt{CEDRUS}^{+}$ bound gains the term

$$\mathrm{InSec}^{\mathsf{S\text{-}TCR}(\mathsf{Good}_{\mathbf{w},z_b})}_{\mathsf{H}_{\mathsf{Root}}\in\mathsf{Th}_\lambda}(\xi', q_6), \qquad q_6 = D_{\mathsf{HT}} < 2^{h+1}.$$

The $\texttt{FORS+C}$ reuse/coverage case remains in the same ITSR step used by that reduction and does not create a second counter-security term. Unequal layer heights change only the target bookkeeping established in Equations (21)–(23). Any probability of exhausting the finite $\texttt{WOTS}^{\texttt{C}}$ or $\texttt{FORS+C}$ counter range is a correctness/performance quantity and must be checked separately from the EU-CMA bound.

S Benchmark Results for the Schemes in Table 1

Table 19: Benchmark results for the schemes listed in Table 1. Key generation, signing and verification time are measured in 1000 CPU cycles. Key and signature sizes are in Bytes. Our codes are compiled with `gcc-11.4.0-O3-march=native-fomit-frame-pointer-flto` on a `Ubuntu 22.04` machine with `Intel Core i5-1135G7` CPU and 16GB RAM, and all cycle counts are the median of 1000 runs. Note that for `CEDRUS`$^+$`C`, we also provide implementations without using the `AVX2` instruction set to have a fair comparison with the original codes provided by the designers of `SPHINCS`$^+$`C`.

Scheme	Signature Size	Signing	Verification	`AVX2`
`SPHINCS`$^+$`-128f`	17088	20367.3	1526.0	✓
`CEDRUS`$^+$`-128f`	16528	20308.4	817.4	✓
`CEDRUS`$^+$`[0x02]`	12096	39455.1	1096.0	
`SPHINCS`$^+$`-128s`	7856	422739.1	608.4	✓
`CEDRUS`$^+$`-128s`	7840	405516.1	565.5	✓
`CEDRUS`$^+$`[0x05]`	7184	718992.9	600.8	
`SPHINCS-`α`-128f`	17040	18755.3	1338.7	✓
`CEDRUS`$^\alpha$`-128f`	16560	18354.2	1229.4	✓
`CEDRUS`$^\alpha$`[0x02]`	12000	37623.8	1312.5	
`SPHINCS-`α`-128s`	6960	404975.5	924.2	✓
`CEDRUS`$^\alpha$`-128s`	6960	404004.4	920.9	✓
`CEDRUS`$^\alpha$`[0x05]`	6528	697934.1	824.7	
`SPHINCS`$^+$`C-128f`	14904	122062.3	5519.3	✗
`CEDRUS`$^+$`C-128f`	14340	120135.9	5160.0	✗
		23930.4	1266.8	✓
`CEDRUS`$^+$`C[0x02]`	10504	237583.1	3485.3	✗
		55919.4	883.2	✓
`SPHINCS`$^+$`C-128s`	6304	2135243.2	12850.7	✗
`CEDRUS`$^+$`C-128s`	6304	2052623.1	11684.0	✗
		579372.0	2707.7	✓
`CEDRUS`$^+$`C[0x05]`	5684	3862931.4	9882.3	✗
		9865421.4	2655.2	✓

T Parameter Settings for CEDRUS$^{\alpha}$

Table 20: Representative parameter sets for CEDRUS$^{\alpha}$

ID	Name	n	h	d	k	a	w'	l	$\mathbf{w}$	Bitsec
–	SPHINCS-α-128f	16	66	22	23	7	2	37	$[13]^{37}$	131
0x00	CEDRUS$^{\alpha}$-128f	16	65	21	24	7	2	37	$[12]^{27} \times [13]^{10}$	128
0x01	–	16	65	17	24	7	2	43	$[8]^{17} \times [9]^{26}$	128
0x02	–	16	64	16	21	8	2	31	$[20]^{25} \times [21]^{6}$	128
–	SPHINCS-α-128s	16	64	8	11	13	2	27	$[32]^{27}$	128
0x03	CEDRUS$^{\alpha}$-128s	16	64	8	11	13	2	27	$[31] \times [32]^{26}$	128
0x04	–	16	67	8	12	11	2	27	$[31] \times [32]^{26}$	128
0x05	–	16	64	7	11	13	2	27	$[31] \times [32]^{26}$	128
–	SPHINCS-α-192f	24	66	22	37	7	2	51	$[15]^{51}$	193
0x06	CEDRUS$^{\alpha}$-192f	24	66	22	37	7	2	51	$[14]^{9} \times [15]^{42}$	193
0x07	–	24	64	16	43	7	2	63	$[8]^{12} \times [9]^{51}$	192
0x08	–	24	66	16	30	8	2	46	$[19]^{2} \times [20]^{44}$	192
–	SPHINCS-α-192s	24	64	8	19	12	2	38	$[38]^{38}$	193
0x09	CEDRUS$^{\alpha}$-192s	24	66	8	19	11	4	40	$[31]^{17} \times [32]^{23}$	193
0x0A	–	24	66	8	21	10	8	39	$[34]^{17} \times [35]^{22}$	193
0x0B	–	24	63	7	19	12	8	37	$[42]^{36} \times [43]$	192
–	SPHINCS-α-256f	32	64	16	48	8	2	66	$[16]^{66}$	256
0x0C	CEDRUS$^{\alpha}$-256f	32	64	16	48	8	2	66	$[15]^{15} \times [16]^{51}$	256
0x0D	–	32	65	16	45	8	2	66	$[15]^{15} \times [16]^{51}$	257
0x0E	–	32	66	13	34	9	4	62	$[18]^{7} \times [19]^{55}$	256
–	SPHINCS-α-256s	32	63	9	27	12	2	48	$[46]^{48}$	256
0x0F	CEDRUS$^{\alpha}$-256s	32	65	9	27	11	4	50	$[38]^{17} \times [39]^{33}$	256
0x10	–	32	65	9	27	11	4	49	$[42]^{14} \times [41]^{35}$	256
0x11	–	32	65	8	22	13	4	49	$[42]^{14} \times [41]^{35}$	257

Table 21: A theoretical comparison of the signature schemes instantiated with the parameter sets listed in Table 20 with the SPHINCS-α algorithms given in [ZCY23]

Scheme	PK Size (Bytes)	SK Size (Bytes)	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
SPHINCS-α-128f	32	64	17040	91681	5157
CEDRUS$^{\alpha}$-128f	32	64	16560	91576	4670
0x01	32	64	15824	101469	3069
0x02	32	64	12000	177213	5042
SPHINCS-α-128s	32	64	6960	2128599	3581
CEDRUS$^{\alpha}$-128s	32	64	6960	2126559	3573
0x04	32	64	6848	2527966	3567
0x05	32	64	6528	3896450	3155
SPHINCS-α-192f	48	96	35640	145849	8253
CEDRUS$^{\alpha}$-192f	48	96	35640	144353	8165
0x07	48	96	34008	160359	4387
0x08	48	96	25752	288372	7342
SPHINCS-α-192s	48	96	14784	3265007	5954
CEDRUS$^{\alpha}$-192s	48	96	14760	3464391	5247
0x0A	48	96	14640	3663213	5684
0x0B	48	96	13680	6349607	5756
SPHINCS-α-256f	64	128	49696	311762	8464
CEDRUS$^{\alpha}$-256f	64	128	49696	308034	8352
0x0D	64	128	48864	321637	8323
0x0E	64	128	38816	622244	7723
SPHINCS-α-256s	64	128	27104	2977382	10161
CEDRUS$^{\alpha}$-256s	64	128	26880	3046336	8948
0x10	64	128	26592	3202120	9452
0x11	64	128	24512	5785012	8423

Table 22: Performance comparison between CEDRUS$^{\alpha}$ and SPHINCS-α simple variants, with tweakable hash functions instantiated with SHAKE-avx2. Key generation, signing and verification time are measured in 1000 CPU cycles. Key and signature sizes are in bytes.

	Key Generation		Signing		Verification		Size	
	SPHINCS-α	CEDRUS$^{\alpha}$	SPHINCS-α	CEDRUS$^{\alpha}$	SPHINCS-α	CEDRUS$^{\alpha}$	SPHINCS-α	CEDRUS$^{\alpha}$
SHAKE-128f	810.3	766.3 (-5.4%)	18755.3	18354.2 (-2.6%)	1338.7	1229.4 (-8.2%)	17040	16560 (-4.3%)
SHAKE-128s	41992.3	40859.9 (-2.7%)	404975.5	404004.4 (-0.2%)	924.2	920.9 (-0.4%)	6960	6960 (-0%)
SHAKE-192f	1287.0	1215.5 (-5.6%)	29703.9	29314.2 (-1.2%)	2143.4	2077.8 (-3.1%)	35640	35640 (-0%)
SHAKE-192s	71213.6	61791.3 (-13.2%)	623945.6	660579.6 (+5.9%)	1481.0	1319.5 (-10.9%)	14784	14760 (-0.2%)
SHAKE-256f	3269.3	3205.2 (-2.0%)	62477.9	61297.6 (-1.9%)	2230.1	2214.2 (-0.7%)	49696	49472 (-0.5%)
SHAKE-256s	53508.0	47079.7 (-12.0%)	570849.5	578761.8 (-1.4%)	2411.2	2116.4 (-12.2%)	27104	26880 (-0.8%)

U Parameter Settings for CEDRUS^{+}C

Table 23: Representative parameter sets for CEDRUS$^+$C

ID	Name	n	h	d	k	a	a'	l	$\mathbf{w}$	z_b	Bitsec
–	SPHINCS$^+$C-128f	16	63	21	19	9	8	32	$[16]^{32}$	0	128
0x00	CEDRUS$^+$C-128f	16	66	20	23	7	6	32	$[8]^1 \times [16]^{31}$	1	128
0x01	–	16	66	19	23	7	6	32	$[8]^1 \times [16]^{31}$	1	128
0x02	–	16	65	13	19	8	11	32	$[8]^2 \times [16]^{30}$	2	128
–	SPHINCS$^+$C-128s	16	66	11	9	13	18	18	$[128]^{18}$	2	128
0x03	CEDRUS$^+$C-128s	16	66	11	9	13	18	18	$[64]^3 \times [128]^{15}$	5	128
0x04	–	16	66	10	10	12	17	19	$[64]^9 \times [128]^{10}$	4	128
0x05	–	16	66	8	9	13	18	20	$[64]^{18} \times [128]^2$	6	128
–	SPHINCS$^+$C-192f	24	63	21	30	9	13	48	$[16]^{48}$	0	192
0x06	CEDRUS$^+$C-192f	24	65	19	41	7	10	48	$[8]^1 \times [16]^{47}$	1	192
0x07	–	24	65	18	41	7	10	48	$[8]^1 \times [16]^{47}$	1	192
0x08	–	24	65	13	27	9	9	48	$[8]^2 \times [16]^{46}$	2	192
–	SPHINCS$^+$C-192s	24	66	11	13	15	12	27	$[128]^{27}$	3	192
0x09	CEDRUS$^+$C-192s	24	66	9	15	13	17	31	$[64]^{31}$	6	192
0x0A	–	24	67	10	16	12	17	27	$[64]^1 \times [128]^{26}$	4	192
0x0B	–	24	68	9	13	14	17	27	$[64]^2 \times [128]^{25}$	5	192
–	SPHINCS$^+$C-256f	32	64	16	34	10	10	64	$[16]^{64}$	0	256
0x0C	CEDRUS$^+$C-256f	32	67	16	35	9	8	64	$[8]^1 \times [16]^{63}$	1	256
0x0D	–	32	67	16	35	9	8	63	$[16]^{62} \times [32]$	3	256
0x0E	–	32	65	12	32	10	12	61	$[16]^{52} \times [32]^9$	3	256
–	SPHINCS$^+$C-256s	32	66	11	19	14	19	42	$[64]^{42}$	4	256
0x0F	CEDRUS$^+$C-256s	32	67	10	23	12	14	42	$[64]^{42}$	4	256
0x10	–	32	67	10	23	12	14	41	$[64]^{36} \times [128]^5$	5	256
0x11	–	32	68	9	20	13	17	40	$[64]^{30} \times [128]^{10}$	6	256

Table 24: A theoretical comparison of the signature schemes instantiated with the parameter sets listed in Table 23 with the SPHINCS$^+$C algorithms given in [HKRY23]

Scheme	PK Size (Bytes)	SK Size (Bytes)	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
SPHINCS$^+$C-128f	32	64	14904	111925	5294
CEDRUS$^+$C-128f	32	64	14340	111771	4971
0x01	32	64	13824	119958	4735
0x02	32	64	10504	223680	3253
SPHINCS$^+$C-128s	32	64	6304	2110892	12766
CEDRUS$^+$C-128s	32	64	6304	2089934	11710
0x03	32	64	6236	2198738	9387
0x04	32	64	5684	4207118	5745
SPHINCS$^+$C-192f	48	96	33016	177541	7924
CEDRUS$^+$C-192f	48	96	31424	177349	7158
0x06	48	96	30268	189723	6802
0x07	48	96	23096	354728	4912
SPHINCS$^+$C-192s	48	96	13776	3739477	19140
CEDRUS$^+$C-192s	48	96	13384	3689083	9070
0x09	48	96	13148	4078769	17106
0x0A	48	96	12208	6856222	15123
SPHINCS$^+$C-256f	64	128	46884	361626	8119
CEDRUS$^+$C-256f	64	128	46212	358466	8034
0x0C	64	128	45700	369921	8114
0x0D	64	128	36852	716275	6778
SPHINCS$^+$C-256s	64	128	26096	3389656	14905
CEDRUS$^+$C-256s	64	128	25228	3259565	13597
0x0F	64	128	24908	3600137	14887
0x11	64	128	22728	6569024	14569

Table 25: Performance comparison between CEDRUS$^+$C and SPHINCS$^+$C simple variants, with tweakable hash functions instantiated with the SHA3 standard SHAKE. Key generation, signing and verification time are measured in 1000 CPU cycles. Key and signature sizes are in bytes. The implementations are not optimized with the AVX2 instruction set for a fair comparison with the original codes provided by the designers of SPHINCS$^+$C.

	Key Generation		Signing		Verification		Size	
	SPHINCS$^+$C	CEDRUS$^+$C	SPHINCS$^+$C	CEDRUS$^+$C	SPHINCS$^+$C	CEDRUS$^+$C	SPHINCS$^+$C	CEDRUS$^+$C
SHAKE-128f	4309.6	4162.3 (-3.4%)	122062.3	120135.9 (-1.6%)	5519.3	5160.0 (-6.5%)	14904	14340 (-3.8%)
SHAKE-128s	150691.6	137310.2 (-8.9%)	2155243.2	2052623.1 (-4.8%)	12850.7	11684.0 (-9.1%)	6304	6304 (-0%)
SHAKE-192f	6384.2	6280.7 (-1.6%)	195146.9	189817.3 (-2.7%)	8367.6	7481.0 (-10.6%)	33016	31424 (-4.8%)
SHAKE-192s	228674.0	257014.3 (+12.4%)	3881711.9	3740386.8 (-3.6%)	19651.7	9192.7 (-53.2%)	13776	13384 (-2.8%)
SHAKE-256f	16887.6	16805.8 (-0.5%)	381758.7	378183.0 (-0.9%)	8515.8	8424.9 (-1.1%)	46884	46212 (-1.4%)
SHAKE-256s	174839.8	174137.7 (-0.4%)	3511072.6	3326166.7 (-5.3%)	15212.4	13820.8 (-9.1%)	26096	25228 (-3.3%)

Table 26: Performance comparison between CEDRUS$^+$C and SPHINCS$^+$C simple variants, with tweakable hash functions instantiated with the SHA3 standard SHAKE. Key generation, signing and verification time are measured in 1000 CPU cycles. Key and signature sizes are in bytes. The implementations are optimized with the AVX2 instruction set.

	Key Generation		Signing		Verification		Size	
	SPHINCS$^+$C	CEDRUS$^+$C	SPHINCS$^+$C	CEDRUS$^+$C	SPHINCS$^+$C	CEDRUS$^+$C	SPHINCS$^+$C	CEDRUS$^+$C
SHAKE-128f	855.8	839.6 (-1.9%)	23934.2	23930.4 (-0.0%)	1351.1	1266.8 (-6.2%)	14904	14340 (-3.8%)
SHAKE-128s	27688.3	25527.8 (-7.8%)	590544.0	579372.0 (-1.9%)	2982.9	2707.7 (-9.2%)	6304	6304 (-0%)
SHAKE-192f	1284.4	1271.7 (-1.0%)	40844.7	37766.3 (-7.5%)	1954.1	1805.3 (-7.6%)	33016	31424 (-4.8%)
SHAKE-192s	42631.5	48276.4 (+13.2%)	800339.1	800332.2 (-0%)	4251.8	2056.4 (-51.6%)	13776	13384 (-2.8%)
SHAKE-256f	3183.2	3149.5 (-1.1%)	74478.8	72336.5 (-2.9%)	2051.3	2016.5 (-1.7%)	46884	46212 (-1.4%)
SHAKE-256s	33686.6	32615.6 (-3.2%)	1031505.3	659447.9 (-36.1%)	3298.9	3018.4 (-8.5%)	26096	25228 (-3.3%)

V Theoretical Comparison And Parameter Settings for CEDRUS$^+$FP

Table 27: Parameter sets for CEDRUS$^+$FP, where $m_{\texttt{max}}$ is the maximum number of authentication nodes.

ID	Name	n	h	d	k	t	$m_{\texttt{max}}$	$\mathbf{w}$	Bitsec
0x00	CEDRUS$^+$FP-128f	16	64	16	29	3712	156	$[4] \times [8]^{42} \times [8]^3$	128
0x01	–	16	65	15	18	8192	121	$[16]^{32} \times [16] \times [32]$	128
0x02	CEDRUS$^+$FP-128s	16	62	7	13	90112	127	$[16]^{32} \times [8]^3$	128
0x03	–	16	66	7	9	262144	102	$[16]^{32} \times [16] \times [32]$	128
0x04	CEDRUS$^+$FP-192f	24	68	17	37	4352	198	$[8]^{56} \times [16]^6 \times [8]^3$	192
0x05	–	24	66	15	27	13312	194	$[16]^{43} \times [32]^4 \times [32]^2$	192
0x06	CEDRUS$^+$FP-192s	24	64	7	18	126976	184	$[16]^{48} \times [8]^2 \times [16]$	192
0x07	–	24	66	7	11	720896	165	$[16]^{48} \times [8]^2 \times [16]$	192
0x08	CEDRUS$^+$FP-256f	32	64	16	43	20480	311	$[16]^{64} \times [32]^2$	256
0x09	–	32	65	13	34	32768	276	$[16]^{32} \times [32]^{24} \times [8] \times [16]^2$	256
0x10	CEDRUS$^+$FP-256s	32	66	8	23	159744	240	$[64]^{64} \times [8]^2 \times [16]$	256
0x11	–	32	64	7	22	344064	256	$[16]^{49} \times [32]^{12} \times [8] \times [16]^2$	256

Table 28: A theoretical comparison of the signature schemes instantiated with the parameter sets listed in Table 27 with the SPHINCS$^+$FP algorithms given in [AK25]

Scheme	PK Size (Bytes)	SK Size (Bytes)	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
SPHINCS$^+$FP-128f	32	64	16420	105281	6472
CEDRUS$^+$FP-128f	32	64	15780	105668	3208
0x01	32	64	11444	207692	4439
SPHINCS$^+$FP-128s	32	64	7332	2188970	2195
CEDRUS$^+$FP-128s	32	64	7172	2155750	2100
0x03	32	64	6660	3724420	2155
SPHINCS$^+$FP-192f	48	96	34540	169405	9349
CEDRUS$^+$FP-192f	48	96	33820	169007	5187
0x05	48	96	24556	337490	6931
SPHINCS$^+$FP-192s	48	96	15316	3732430	3162
CEDRUS$^+$FP-192s	48	96	14980	3731269	3093
0x07	48	96	14452	6389687	3066
SPHINCS$^+$FP-256f	64	128	48132	346260	9530
CEDRUS$^+$FP-256f	64	128	47204	345805	9183
0x09	64	128	37412	666778	9212
SPHINCS$^+$FP-256s	64	128	28260	3264214	4666
CEDRUS$^+$FP-256s	64	128	27716	3251444	4586
0x11	64	128	25316	6018424	4601

Table 29: A theoretical comparison of the signature schemes instantiated with the parameter sets listed in Table 27 with the SPHINCS$^+$FP algorithms

	Signing(#Hash call)		Verification(#Hash call)		Size(Bytes)	
	SPHINCS$^+$FP	CEDRUS$^+$FP	SPHINCS$^+$FP	CEDRUS$^+$FP	SPHINCS$^+$FP	CEDRUS$^+$FP
128f	105281	105668 (+0.4%)	6472	3208 (-50.4%)	16420	15780 (-3.9%)
128s	2188970	2155750 (-1.5%)	2195	2100 (-4.3%)	7332	7172 (-2.2%)
192f	169405	169007 (-0.2%)	9349	5187 (-44.5%)	34540	33820 (-2.1%)
192s	3732430	3731269 (-0.0%)	3162	3093 (-2.2%)	15316	14980 (-2.2%)
256f	346260	345805 (-0.1%)	9530	9183 (-3.6%)	48132	47204 (-1.9%)
256s	3264214	3251444 (-0.4%)	4666	4586 (-1.7%)	28260	27716 (-1.9%)

Table 30: Parameter sets for CEDRUS$^+$C-FP, where $m_{\texttt{max}}$ is the maximum number of authentication nodes.

ID	Name	n	h	d	k	t	$m_{\texttt{max}}$	$\mathbf{w}$	z_b	Bitsec
0x00	CEDRUS$^+$C-FP-128f	16	65	20	24	3869	134	$[8] \times [16]^{31}$	1	128
0x01	–	16	65	13	19	6817	123	$[8]^3 \times [16]^{29}$	3	128
0x02	CEDRUS$^+$C-FP-128s	16	65	9	12	46954	106	$[32]^2 \times [64]^{19}$	4	128
0x03	–	16	68	9	11	37291	93	$[64]^4 \times [128]^{14}$	6	128
0x04	CEDRUS$^+$C-FP-192f	24	65	19	40	6507	235	$[8]^5 \times [16]^{44}$	1	192
0x05	–	24	65	13	27	16241	200	$[16]^{46} \times [32]$	3	192
0x06	CEDRUS$^+$C-FP-192s	24	66	9	17	102997	167	$[64]^{31}$	6	192
0x07	–	24	67	8	15	186124	161	$[64]^{26} \times [128]^4$	8	192
0x08	CEDRUS$^+$C-FP-256f	32	67	16	37	16734	264	$[8] \times [16]^{63}$	1	256
0x09	–	32	66	12	36	21802	269	$[16]^{52} \times [32]^9$	3	256
0x10	CEDRUS$^+$C-FP-256s	32	67	10	25	81271	236	$[64]^{42}$	4	256
0x11	–	32	64	8	23	259708	253	$[32]^1 \times [64]^{41}$	5	256

Table 31: A theoretical comparison of the signature schemes instantiated with the parameter sets listed in Table 30 with the SPHINCS$^+$C-FP algorithms given in [AK25]

Scheme	PK Size (Bytes)	SK Size (Bytes)	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
SPHINCS$^+$C-FP-128f	32	64	14504	115271	5289
CEDRUS$^+$C-FP-128f	32	64	13908	111580	4968
0x01	32	64	10040	229803	3191
SPHINCS$^+$C-FP-128s	32	64	6096	2109422	12763
CEDRUS$^+$C-FP-128s	32	64	6008	2079282	5866
0x03	32	64	5400	4139889	9319
SPHINCS$^+$C-FP-192f	48	96	32104	187668	7917
CEDRUS$^+$C-FP-192f	48	96	30608	177264	6993
0x05	48	96	21752	374652	5013
SPHINCS$^+$C-FP-192s	48	96	13104	3733821	19126
CEDRUS$^+$C-FP-192s	48	96	12760	3677060	9061
0x07	48	96	11652	7429218	8843
SPHINCS$^+$C-FP-256f	64	128	45412	361537	8109
CEDRUS$^+$C-FP-256f	64	128	44644	357703	8022
0x09	64	128	35380	721871	6768
SPHINCS$^+$C-FP-256s	64	128	25168	3316446	14896
CEDRUS$^+$C-FP-256s	64	128	24012	3257770	13584
0x11	64	128	21700	6516312	10820

Table 32: A theoretical comparison of the signature schemes instantiated with the parameter sets listed in Table 30 with the SPHINCS$^+$C-FP algorithms

	Signing(#Hash call)		Verification(#Hash call)		Size(Bytes)	
	SPHINCS$^+$C-FP	CEDRUS$^+$C-FP	SPHINCS$^+$C-FP	CEDRUS$^+$C-FP	SPHINCS$^+$C-FP	CEDRUS$^+$C-FP
128f	115271	111580 (-3.2%)	5289	4968 (-6.1%)	14504	13908 (-4.1%)
128s	2109422	2079282 (-1.4%)	12763	5866 (-54.0%)	6096	6008 (-1.4%)
192f	187668	177264 (-5.5%)	7917	6993 (-11.7%)	32104	30608 (-4.7%)
192s	3733821	3677060 (-1.5%)	19126	9061 (-52.6%)	13104	12760 (-2.6%)
256f	361537	357703 (-1.1%)	8109	8022 (-1.1%)	45412	44644 (-1.7%)
256s	3316446	3257770 (-1.8%)	14896	13584 (-8.8%)	25168	24012 (-4.6%)

W Theoretical Comparison And Parameter Settings for Smaller CEDRUS$^+$

Table 33: Theoretical comparison and parameter settings of Smaller SPHINCS$^+$ versus CEDRUS$^+$

Name	n	N	hash limit	h	d	a	k	w	Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)	overuse safety
SPHINCS$^+$-A-1	16	2^{20}	10^5	20	4	8	23	$[16]^{35}$	5888	89576	1353	3
CEDRUS$^+$-A	16	2^{20}	10^5	20	4	9	19	$[16]^{32} \times [8]^3$	5616	98028	1288	3
SPHINCS$^+$-E-2	16	2^{30}	10^6	32	4	10	15	$[32]^{28}$	4960	965616	1995	8
CEDRUS$^+$-E	16	2^{30}	10^6	32	4	11	13	$[16]^2 \times [32]^{26}$	4816	966642	1922	8
SPHINCS$^+$-I-2	16	2^{40}	10^7	45	5	16	8	$[128]^{21}$	4592	8459254	6908	59
CEDRUS$^+$-I	16	2^{40}	10^7	46	5	14	9	$[128]^{16} \times [256]^2 \times [64]^2$	4512	8706037	6908	61
SPHINCS$^+$-N-8	24	2^{20}	10^6	28	7	12	16	$[256]^{26}$	10056	942284	23541	2683
CEDRUS$^+$-N	24	2^{20}	10^6	28	5	12	16	$[64]^{11} \times [128]^{18} \times [64]^2$	9408	999918	8083	2683
SPHINCS$^+$-P-2	24	2^{20}	10^8	26	2	21	9	$[16]^{51}$	7848	70025208	1044	6793
CEDRUS$^+$-P	24	2^{20}	10^8	27	2	21	9	$[16]^{48} \times [32]^2$	7824	77119480	1061	13588
SPHINCS$^+$-Q-1	24	2^{30}	10^6	30	5	12	20	$[64]^{34}$	11064	942698	5737	40
CEDRUS$^+$-Q	24	2^{30}	10^6	35	6	12	17	$[64]^{32} \times [32] \times [64]$	11064	964268	6696	512

Table 34: Theoretical comparison and parameter settings of Smaller SPHINCS$^+$FP versus CEDRUS$^+$FP

Name	n	N	hash limit	h	d	t	k	w	$m_{\max}$	Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
SPHINCS$^+$FP-A-1	16	2^{20}	10^5	20	4	5888	23	$[16]^{35}$	143	5236	90529	1335
CEDRUS$^+$FP-A	16	2^{20}	10^5	20	4	9216	19	$[16]^{32} \times [8]^3$	134	5028	97132	1270
SPHINCS$^+$FP-D-9	16	2^{20}	10^8	17	1	8388608	8	$[16]^{35}$	125	2980	99800923	441
CEDRUS$^+$FP-D	16	2^{20}	10^8	17	1	6291456	8	$[16]^{33} \times [32]$	123	2932	93012258	439
SPHINCS$^+$FP-E-2	16	2^{30}	10^6	32	4	15360	15	$[32]^{28}$	114	4388	973177	1974
CEDRUS$^+$FP-E	16	2^{30}	10^6	32	4	24576	13	$[16]^2 \times [32]^{26}$	108	4260	968704	1900
SPHINCS$^+$FP-PPP-1	24	2^{20}	10^9	30	2	67108864	8	$[256]^{26}$	144	5644	653380841	6850
CEDRUS$^+$FP-PPP	24	2^{20}	10^9	30	2	50331648	8	$[256]^{24} \times [64] \times [128]$	140	5548	590480050	6526
SPHINCS$^+$FP-R-1	24	2^{30}	10^8	32	4	851968	13	$[256]^{26}$	168	7636	9501433	13544
CEDRUS$^+$FP-R	24	2^{30}	10^8	32	4	720896	13	$[256]^{24} \times [64] \times [128]$	163	7516	8984335	12899

Table 35: Parameter sets for Smaller CEDRUS$^+$FP supporting 2^{24} signatures

Name	n	h	d	k	t	$m_{\max}$	$\mathbf{w}$	Bitsec
CEDRUS$^+$FP-rls128cs1-1	16	21	1	6	201326592	114	$[8]^{36} \times 16^5 \times [8]^3$	128
CEDRUS$^+$FP-rls128cs1-2	16	22	1	7	14680064	108	$[4]^{28} \times 8^{24} \times [4] \times [8]^2$	128
CEDRUS$^+$FP-rls192cs1-1	24	21	1	10	83886080	181	$[8]^{16} \times [16]^{36} \times [8]^2 \times [16]$	192
CEDRUS$^+$FP-rls192cs1-2	24	21	1	10	83886080	181	$[8]^{32} \times [16]^{24} \times [8]^2 \times [16]$	192
CEDRUS$^+$FP-rls256cs1-1	32	21	1	14	58720256	250	$[4]^2 \times [8]^{84} \times [4]^2 \times [8]^2$	256
CEDRUS$^+$FP-rls256cs1-2	32	21	1	14	58720256	249	$[4]^{71} \times [8]^{38} \times [4]^3 \times [8]$	256

Table 36: A theoretical comparison of the signature schemes instantiated with the parameter sets listed in Table 35 with the Smaller SPHINCS$^+$FP algorithms supporting 2^{24} signatures

Scheme	Signature Size (Bytes)	Signing (#Hash call)	Verification (#Hash call)
SPHINCS$^+$FP-rls128cs1	3316	1457500225	283
CEDRUS$^+$FP-rls128cs1-1	2980	1450957761	346
CEDRUS$^+$FP-rls128cs1-2	3092	1447067476	309
SPHINCS$^+$FP-rls192cs1	6748	2048864457	493
CEDRUS$^+$FP-rls192cs1-1	6436	2048470556	593
CEDRUS$^+$FP-rls192cs1-2	6532	1914252828	561
SPHINCS$^+$FP-rls256cs1	13444	2336980368	567
CEDRUS$^+$FP-rls256cs1-1	12036	2107392397	654
CEDRUS$^+$FP-rls256cs1-2	12740	2226681799	605