

# Improved constant-sum encodings for hash-based signatures

Lucas Pandolfo Perin · Gustavo Zambonin ·  
Ricardo Custódio · Lucia Moura · Daniel Panario

Received: 20 April 2020 / Accepted: 4 May 2021

**Abstract** The Winternitz one-time signature scheme is one of the cornerstones of hash-based signatures. Cruz, Kaji, and Yatani (CKY) propose to use a constant-sum encoding function with this scheme to obtain signature verification at lower and predictable costs in exchange for increased costs of key generation and signature verification. We give a novel description of this scheme called Wots-cs that greatly reduces the costs associated with key and signature generation, as well as signature verification. We achieve this by introducing new deterministic constant-sum encoding algorithms that accept larger sets of parameters than the original proposal. In addition, we provide a security proof of our scheme that relies on weaker assumptions than the CKY variant, reducing signature sizes by 50%. Finally, we compare our work with Wots+ for parameters with the same signature size, and

experiment with Xmss to discuss the impact of the encoding and possible applications.

**Keywords** Post-quantum cryptography · Hash-based signatures · Winternitz one-time signature scheme · XMSS

## 1 Introduction

Hash-based signature schemes are well-known, quantum-safe cryptographic algorithms with well-established security reductions. Generally, in such schemes, the private key can be used to sign only a single message. However, instances of the same one-time signature scheme can be grouped in a Merkle tree to achieve a many-time signature scheme. A central one-time signature method is the Winternitz one-time signature scheme, or Wots [18]. It is widely used in more complex schemes such as Xmss, published as RFC 8391 [13] and recommended as a supplement to FIPS 186 [7], and SPHINCS<sup>+</sup> [3]. The latter is currently in the third round of the NIST post-quantum standardization process [1].

Encoding functions applied to messages are pivotal in hash-based signatures. They define essential aspects of the digital signature, for example, security, signature sizes and overall performance. Notably, in Wots, the encoding output determines the signature generation and verification performance. Recently, this fact has been exploited by repeatedly randomizing and electing a “better” encoding of the original message with a nonce [19]. This approach has been proved secure and used for faster Xmss signature verification in embedded devices [6].

The default Winternitz encoding is called base-w. This encoding is trivial to compute from any given message, and it requires a simple checksum to preserve the security of the scheme. Several alternatives to base-w have also been explored in the literature. For example, the run lengths of the

---

L.P. Perin and G. Zambonin were supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. L. Moura and D. Panario were partially supported by NSERC of Canada.

---

L.P. Perin, G. Zambonin and R. Custódio  
Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis, SC, 88040-900, Brasil.  
E-mail: lucas.perin@posgrad.ufsc.br  
E-mail: gustavo.zambonin@posgrad.ufsc.br  
E-mail: ricardo.custodio@ufsc.br

L.P. Perin  
Technology Innovation Institute, P.O. Box 9639, Masdar City, Abu Dhabi, United Arab Emirates

L. Moura  
School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON, K1N 6N5, Canada.  
E-mail: lmoura@uottawa.ca

D. Panario  
School of Mathematics and Statistics, Carleton University, Ottawa, ON, K1S 5B6, Canada.  
E-mail: daniel@math.carleton.ca

message are considered in [22], and non-adjacent forms are used in [21].

In this work, we bring to attention recent developments by Cruz, Kaji, and Yatani (CKY) [8, 14]. They introduce a new feature, namely that of a constant-sum encoding. This approach maps the input message to a tuple of integers whose sum is a previously established value. This is useful to accurately predict the resources needed by the digital signature scheme and is not contemplated by any of the previously mentioned works. Moreover, the content of the message to be signed no longer determines the performance of the underlying scheme, and a checksum is no longer required. However, we note that the CKY approach has some restrictions regarding the choice of the summation target, the shape of the resulting tuples, and the performance of their encoding method.

The constant-sum encoding provides valuable properties that can significantly reduce costs associated with the number of chaining function iterations of Winternitz-like schemes. In addition to the aforementioned predictability of costs, it can also reduce key generation costs while preserving the signature length. Indeed, such properties have been examined and briefly commented on by the SPHINCS<sup>+</sup> team in their submission to the third round of the NIST standardization process [2]. However, the complexity of the current state of the art for the constant-sum encoding algorithms is the main obstacle for its adoption.

Finally, we observe that for schemes such as SPHINCS<sup>+</sup> and XMSS, the key generation algorithm may be invoked multiple times when signing a message. This is mainly due to the computation of the authentication path, needed to verify the signature against the root node of the Merkle tree. Although optimized tree traversal algorithms may increase signature performance, we remark that reducing key generation costs can also contribute to this improvement. This motivates our present research, suggesting that further investigation of different encoding algorithms may prove interesting for such schemes.

### 1.1 Our contributions

Our main contribution is Wots-cs, a variation of the Wots+ scheme [12], with the replacement of base-w with a new deterministic constant-sum encoding. Our proposal allows more flexibility in the choice of parameters, and consequently, further gains in the key generation performance. Indeed, we provide experimental results showing that the work factor associated with our encoding algorithms is greatly reduced when compared to methods due to CKY. Furthermore, we give a security proof with concrete security levels that relies on weaker assumptions than the one presented in [14]. This reduces the signature size by 50%, mainly due to the

use of security features present in the Wots+ scheme that were not considered in the original constant-sum proposals.

In more specific terms, we list the following additional contributions:

1. We provide new constant-sum encoding algorithms that are deterministic. We benchmark our algorithms and show that they reduce the cost of encoding a message before signing or verifying by up to 87% when compared to CKY. We also give alternative algorithms that further reduce the encoding cost by using a caching method or by efficiently verifying a pre-computed encoding given with the signature.
2. We propose new sets of parameters that reduce the number of hash operations in the key generation of Wots-cs to be the same as Wots+ in the worst case, and 11.37% less than Wots+ in the best case; see Table 3. We decrease the key generation cost by reducing the length of the hash chains, thus reducing the number of hash operations for both signature generation and signature verification. We benchmark our scheme with XMSS and show how it performs compared to Wots+ considering the encoding running time; see Table 4.
3. We also propose alternative parameters that reduce signature generation cost instead of signature verification. These parameters are interesting for applications where a device receives a pre-computed encoding, reducing signature generation cost of Wots-cs by up to 20.39% if compared to Wots+; see Table 5. This result is also carried over to XMSS, but with less impact (2–3% for  $t = 67$ ) due to the cost of generating the verification path. We remark that by optimizing the XMSS traversal algorithm, this result can be improved. Furthermore, by considering the cases where our parameters reduce the key generation time, the signature performance of XMSS is further improved (15% for  $t = 34$ ) without any optimization beyond what is being used in the reference implementation; see Sect. 4.2.

While we demonstrate several advantages of Wots-cs, we remark that the increased cost of the encoding algorithm is a major drawback when compared with Wots+. In this paper, we study the impact of this cost and point to future research for improving it.

### 1.2 Limitations and applicability

In this paper, we present two methods to find parameters for Wots-cs: (i) MINVER searches for parameters that reduce signature verification costs, while (ii) MINGEN searches for parameters that reduce key generation costs. We remark that MINVER is the approach explored in [14], and that it is most desirable for devices with scarce resources. However, as a

trade-off, MINVER drastically increases key generation cost and signature costs. Furthermore, we believe that the computation of the encoding in such devices poses a challenge due to the high complexity of the constant-sum encoding algorithms.

Due to these reasons, we focus on the MINGEN strategy. We argue that using MINGEN to search for parameters in the context of constant-sum encodings provides novel results related to the reduction of key generation costs. The predictability of costs may be useful even if MINGEN gives comparable generation costs to WOTS+, since this approach allows to choose a lower relative cost for signature generation or verification (see Fig. 1), according to the needs of the application. In addition, this parameter strategy seems to be more beneficial to applications where the cost of encoding may be less critical than the cost of hash operations and where we benefit from the predictability of performance of signature or verification. We discuss such applications in Sect. 1.3 and elaborate how the limitations of constant-sum encoding algorithms can be addressed in future works in Sect. 6.1.

### 1.3 Applications

The main applications that can benefit from our contributions are those that require predictability of the costs associated with the generation of keys and signatures, and the subsequent signature verification. These applications are most commonly found on computing devices with limited resources. However, the benefit of predictability leads to an increased encoding cost. Therefore, through examples, we present applications where the encoding costs can be adequately treated, clarifying the benefits of the schemes proposed in this work.

A case of interest is a typical scenario in which devices such as smart cards are used to sign electronic documents. In general, these devices generate private keys with minimal memory and processing capabilities, without ever exposing them to the outside world. For a document to be signed, it is necessary to determine its hash externally to the smart card. In practice, this step is done by computers with higher computational power, and the hash is then sent to the smart card to be signed. Upon receiving the signature, the external computer proceeds with the assembly of the representation structure of the signed document [20, Chapter 23]. Considering this scenario, the constant-sum encoding can be previously computed and sent to the signing device. We can also choose the exact number of hash operations that we want to perform during the signature by observing the parameters proposed in this work. We show in Sect. 4.2 that we can improve the generation of signatures for this case, using parameters compatible to the standard ones for WOTS+ [13].

The second scenario of considerable practical importance is the preservation of the long-term authenticity of data. For example, consider large signature databases that need to be updated to be resistant against quantum attacks. Such applications might require a substantial volume of signatures to be generated over time. Since hash-based signatures are most likely going to be deployed in secure platforms that can guarantee state integrity [7], it is also likely that the signing device will be provided only with the necessary data to perform the signature, despite its CPU capabilities. Hence, a trusted party can perform the document digest and its encoding, and place it in a queue. In this case, the signing device may benefit from our parameter selection, with a smaller and fixed number of hash operations to perform for each signature, increasing its throughput. Furthermore, to decrease storage space requirements, less traditional parameters can be used to output WOTS-CS signatures that are close to half of the size required by using standard parameters (e.g.,  $t = 34$  in Table 3). We also obtain significant improvements to key generation costs for these new parameter selections, which may also impact signature performance, further decreasing the number of required hash operations.

Finally, in some other situations, it is interesting to focus on signatures that are verified more efficiently. This is a natural assumption since a single signature is most likely to be generated only once and verified repeatedly. The proposal in [19] achieves this result with WOTS+ by introducing a trade-off technique that increased signature generation costs. It has also been shown to produce remarkable results on the Cortex-M4 microcontroller [6]. Considering this scenario, WOTS-CS can control and reduce the number of hash operations needed to be performed in the verification process. We recall that, whenever possible, the encoding can be included in the signed message, which the verifier can either trust or use to reduce the encoding cost, before verifying the signature.

### 1.4 Outline

In Sect. 2, we introduce WOTS+ and its variant WOTS-CS, featuring a constant-sum encoding method as presented in the literature. In Sect. 3, we provide several improvements to the constant-sum encoding algorithm. In Sect. 4, we propose new sets of parameters for WOTS-CS, discuss experimental results including the encoding cost, provide detailed running time results, and compare them to previous results in the literature. We also explore the consequences of using our proposals in the context of XMSS. In Sect. 5, we show that our proposal is existentially unforgeable under an adaptive chosen-message attack (EU-CMA). We remark that the security argument in [14] has a flaw, and we provide a revised argument that additionally considers our new parameters. Furthermore, we give concrete estimates of its security level

under classical and quantum attacks. Finally, in Sect. 6, we present our conclusions and suggest future work.

## 2 Preliminaries

### 2.1 Winternitz one-time signature scheme

In this subsection, we define the Winternitz signature scheme based on the Wots+ variant of [12]. This variant is used on current iterations of Merkle-based signature schemes [13], since it reduces the security requirements needed for the underlying hash function. Consider the security parameter  $\lambda \in \mathbb{N}$ , the message length  $m \in \mathbb{N}$  and the Winternitz parameter  $w \geq 1$ , usually a small power of 2. Then, we obtain the values

$$t_1 = \left\lceil \frac{m}{\log_2 w} \right\rceil, \quad t_2 = \left\lceil \frac{\log_2 t_1 (w-1)}{\log_2 w} \right\rceil + 1 \text{ and } t = t_1 + t_2.$$

Let  $\mathcal{F}_\lambda : \{f_k : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda \mid k \in \mathcal{K}_\lambda\}$  be a family of functions with key space  $\mathcal{K}_\lambda$ . Moreover, consider a “chaining” function  $c_k^i(x, r)$  where  $i \in \mathbb{N}$ ,  $x \in \{0, 1\}^\lambda$  and  $r = (r_1, \dots, r_j) \in \{0, 1\}^{\lambda \times j}$  with  $j \geq i$ . This function is defined recursively as

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

For improved readability, we denote  $(r_a, \dots, r_b) \in r$  as  $r_{a,b}$  when  $a \leq b$ , otherwise  $r_{a,b}$  is an empty string. We also use throughout the paper the notation  $x \xleftarrow{\$} \mathcal{X}$  to denote that  $x$  is taken uniformly at random from the set  $\mathcal{X}$ .

We hereafter define the three algorithms of a signature scheme, i.e., key generation, signature generation and signature verification, respectively, as  $\text{GEN}$ ,  $\text{SIG}$  and  $\text{VER}$ .

$\text{GEN}(1^\lambda)$ . Randomly choose  $t+w-1$  strings of  $\lambda$  bits to create the secret key  $\mathbf{sk} = (x_{t-1}, \dots, x_0)$  and  $r = (r_1, \dots, r_{w-1})$ .

Then, by choosing  $k \xleftarrow{\$} \mathcal{K}_\lambda$ , compute the public key  $\mathbf{pk} = ((r, k), c_k^{w-1}(x_{t-1}, r), \dots, c_k^{w-1}(x_0, r))$ .

$\text{SIG}(M, \mathbf{sk})$ . Take an  $m$ -bit message  $M$  and represent it as a  $t_1$ -tuple of base- $w$  words, i.e.,  $\mathcal{B}_1 = (b_{t-1}, \dots, b_{t-t_1})$ . The generic version of this encoding is referred to as base- $w$  in RFC 8391 [13]. A checksum is computed by considering the integer representation of each element in  $\mathcal{B}_1$  as  $\sum_{j=1}^{t_1} (w-1-b_{t-j})$ . Finally, applying base- $w$  to the checksum analogously yields  $\mathcal{B}_2 = (b_{t-1}, \dots, b_0)$ . Let  $\mathcal{B} = (b_{t-1}, \dots, b_0)$ . The signature is generated as

$$\begin{aligned} \sigma &= (\sigma_{t-1}, \dots, \sigma_0) \\ &= (c_k^{b_{t-1}}(x_{t-1}, r), \dots, c_k^{b_0}(x_0, r)). \end{aligned}$$

$\text{VER}(M, \sigma, \mathbf{pk})$ . Obtain the  $t$ -tuple  $\mathcal{B}$ . The correctness of  $\sigma$  is asserted by computing the remaining iterations over

signature blocks and comparing the results with the public key, i.e.,

$$\mathbf{pk} \stackrel{?}{=} ((r, k), c_k^{w-1-b_{t-1}}(\sigma_{t-1}, r_{b_{t-1}+1, w-1}), \dots, c_k^{w-1-b_0}(\sigma_0, r_{b_0+1, w-1})).$$

We remark that the checksum prevents signature forgeries by forcing a malicious party to calculate a preimage on iterates of  $f_k$  if any value in  $\mathcal{B}_1$  is modified. Let the number of iterations of  $f_k$  for  $\text{GEN}$  be denoted as  $C(\text{GEN})$  and analogously for  $\text{SIG}$  and  $\text{VER}$ . Evidently,  $C(\text{GEN}) = C(\text{SIG}) + C(\text{VER})$ . In the case of WOTS, we have that  $C(\text{GEN}) = t(w-1)$ .

Let  $M$  be taken uniformly at random; then, the elements of  $\mathcal{B}_1$  are also uniformly distributed. The checksum elements may not behave uniformly, but as  $t_2 \ll t_1$  for commonly used Wots+ parameters, it contributes very little to the overall cost of the scheme. This entails costs, on average, of  $\overline{C(\text{SIG})} \approx \overline{C(\text{VER})} \approx \frac{1}{2}C(\text{GEN}) \pm \epsilon$ , where  $\epsilon$  is the average distance of the checksum distribution from uniform. It is expected that  $\epsilon$  is small; for more information about the average cost of Wots+ signature and verification, see [6]. Usual parameters for Wots+ from [13] are presented in Table 1, associated with their respective average costs.

**Table 1** Number of iterations of  $c_k$  for usual parameters of Wots+.

| $m$ | $t$ | $w$ | $C(\text{GEN})$ | $\approx \overline{C(\text{SIG})} \mid \overline{C(\text{VER})}$ |
|-----|-----|-----|-----------------|------------------------------------------------------------------|
| 256 | 67  | 16  | 1005            | 502.5                                                            |
|     | 45  | 64  | 2835            | 1417.5                                                           |
|     | 34  | 256 | 8670            | 4335.0                                                           |
| 512 | 131 | 16  | 1965            | 982.5                                                            |
|     | 89  | 64  | 5607            | 2803.5                                                           |
|     | 66  | 256 | 16830           | 8415.0                                                           |

### 2.2 Constant-sum Winternitz variant

In this subsection, we review the CKY proposal [14], which is a Wots variant with an alternative encoding to base- $w$ , where the running time performance of the signature-related procedures is independent of the signed message. We adapt the original definitions to use the improvements of Wots+ as previously discussed. We recall the security parameter  $\lambda \in \mathbb{N}$  and message length  $m \in \mathbb{N}$ . The definition of the function family  $\mathcal{F}_\lambda$  and  $r$  are also maintained.

Since the base- $w$  encoding is not used in this variant, the number of signature blocks  $t$  is no longer determined by the parameter  $w$ . Hence, to avoid confusion, we introduce a new parameter  $n$  that serves a similar purpose as the “chain” length and corresponds to  $w-1$  in the Wots+ scheme. Let

$n, s \in \mathbb{N}$  where  $n \leq s$  such that

$$\tau_{(t,n,s)} = \{(b_{t-1}, \dots, b_0) : 0 \leq b_i \leq n \text{ and } \sum_{i=0}^{t-1} b_i = s\},$$

that is, the set of  $t$ -tuples whose elements are individually bounded by  $n$  and their sum is exactly equal to  $s$ . We also define the set  $\iota_{(t,n,s)} = \{0, \dots, |\tau_{(t,n,s)}| - 1\}$ .

Finally, we also define a constant-sum encoding function  $\mathbf{E} : \iota_{(t,n,s)} \rightarrow \tau_{(t,n,s)}$ , i.e., a bijective map with integer representations of the message space in the domain. It is required that messages are in  $\iota_{(t,n,s)}$ , i.e.,  $2^m \leq |\tau_{(t,n,s)}|$ . We are now able to define a variant scheme WOTS-CS( $\lambda, m, t, n, s$ ), as follows:

**GEN**( $1^\lambda$ ). Randomly choose  $t + n$  strings of  $\lambda$  bits to create the secret key  $\mathbf{sk} = (x_{t-1}, \dots, x_0)$  and  $r = (r_1, \dots, r_n)$ . The public key is obtained by choosing  $k \leftarrow_{\$} \mathcal{K}_\lambda$  and by computing  $\mathbf{pk} = ((r, k), c_k^n(x_{t-1}, r), \dots, c_k^n(x_0, r))$ .

**SIG**( $M, \mathbf{sk}$ ). Consider an  $m$ -bit message  $M$  and its unique constant-sum encoding  $\mathbf{E}(M) = (b_{t-1}, \dots, b_0)$ . Then, the signature is generated as:

$$\begin{aligned} \sigma &= (\sigma_{t-1}, \dots, \sigma_0) \\ &= (c_k^{n-b_{t-1}}(x_{t-1}, r), \dots, c_k^{n-b_0}(x_0, r)). \end{aligned}$$

**VER**( $M, \sigma, \mathbf{pk}$ ). Obtain  $\mathbf{E}(M) = (b_{t-1}, \dots, b_0)$ . The correctness of  $\sigma$  is asserted by computing the remaining iterations over signature blocks and comparing the results with the public key, namely

$$\mathbf{pk} \stackrel{?}{=} ((r, k), c_k^{b_{t-1}}(\sigma_{t-1}, r_{n-b_{t-1},n}), \dots, c_k^{b_0}(\sigma_0, r_{n-b_0,n})).$$

We point out that the checksum calculation is now not needed for  $\mathbf{E}(M)$ . By letting

$$\mathbf{E}'(M) = \text{base-w}(M) + \text{base-w}(\text{checksum}(M)),$$

then  $\mathbf{E}(M)$  is analogous to  $\mathbf{E}'(M)$  in the sense that tampering with intermediate  $b_i$  values while maintaining the constant-sum limitation implies in decreasing a  $b_i$  value. Consequently, a malicious party would need to obtain a preimage on iterates of  $f_k$ . Moreover, the costs  $C(\text{GEN}) = tn$ ,  $C(\text{SIG}) = tn - s$  and  $C(\text{VER}) = s$  are precisely determined. In addition to this fact, a result [9] on the symmetry of  $|\tau_{(t,n,s)}|$  around  $s = \frac{tn}{2}$  implies the interchangeability of  $C(\text{SIG})$  and  $C(\text{VER})$ . We restate this result and give a short proof for completion.

**Proposition 1** *Let  $n \geq 1$ ,  $t \geq 1$ ,  $0 \leq s \leq tn$ . Then,*

$$|\tau_{(t,n,s)}| = |\tau_{(t,n,tn-s)}|.$$

*Proof* Consider the bijection

$$\begin{aligned} B : \tau_{(t,n,s)} &\rightarrow \tau_{(t,n,tn-s)}, \\ B(x_{t-1}, \dots, x_0) &= (n - x_{t-1}, \dots, n - x_0). \end{aligned}$$

We verify that  $(x_{t-1}, \dots, x_0) \in \tau_{(t,n,s)}$  if and only if  $\sum_{i=0}^{t-1} x_i = s$  if and only if  $\sum_{i=0}^{t-1} (n - x_i) = tn - s$  if and only if  $(n - x_{t-1}, \dots, n - x_0) \in \tau_{(t,n,tn-s)}$ . The fact that  $B$  is a bijection guarantees that both sets have the same cardinality.  $\square$

In other words, we can use an analogous definition of WOTS-CS to exchange the performance of SIG and VER. In Fig. 1, we display how the message encoding affects the hash chains for SIG and VER in: (a) WOTS+, (b) WOTS-CS with minimization of  $C(\text{VER})$ , and (c) WOTS-CS with minimization of  $C(\text{SIG})$ . In each picture, the rectangular areas represent the number of hash chains for SIG and for VER, which are average values in (a) while exact values in (b) and (c). We now explore the possible implementations of the encoding mapping  $\mathbf{E}$ .

### 2.3 CKY encoding algorithms

The original method to compute a constant-sum encoding from an  $m$ -bit message [8], which is called CKY-O hereafter, works specifically for the case that “chain” lengths are bounded by  $s$ . Thus,  $n = s$  and  $|\tau_{(t,s,s)}| = \binom{s+t-1}{s}$ . The main idea behind the algorithm is to consider the message as an integer  $I \in \iota_{(t,s,s)} = \{0, \dots, |\tau_{(t,s,s)}| - 1\}$  and to compute its unique representation by unranking it to  $\tau_{(t,s,s)}$ .

A direct improvement by the same authors [14], which we call CKY-I, explores the fact that tuples in  $\tau_{(t,s,s)}$  usually have all  $b_i$  distant from their maximum value  $s$ . Indeed, if there exists an integer  $n$  much smaller than  $s$  such that  $0 \leq b_i \leq n$  and  $2^m \leq |\tau_{(t,n,s)}| < |\tau_{(t,s,s)}|$ , then costs of signature operations are substantially reduced. The following theorem gives details on the number of such tuples.

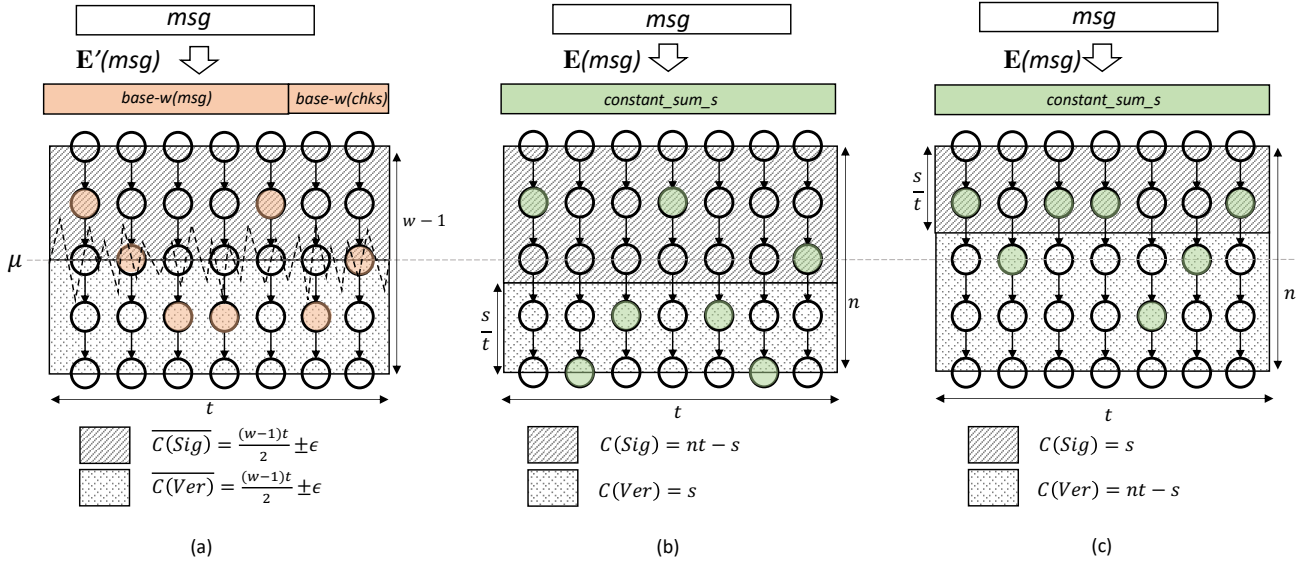
**Theorem 1 ([4, Item 1d])** *Consider the set of  $t$ -tuples with elements bounded by  $n$  whose sum is  $s$ . The cardinality of this set is given by*

$$|\tau_{(t,n,s)}| = \sum_{i=0}^k (-1)^i \binom{t}{i} \binom{s - (n+1)i + t - 1}{t-1},$$

where  $k = \min\left(t, \left\lfloor \frac{s}{n+1} \right\rfloor\right)$ .

*Proof* See Appendix A.1.

For some input  $I$  taken uniformly at random from  $\iota_{(t,s,s)}$ , the CKY-I algorithm uses the result above observing that  $\tau_{(t,n,s)} \subseteq \tau_{(t,s,s)}$ , and that  $\mathbf{E}(I) \in \tau_{(t,s,s)}$  is also in  $\tau_{(t,n,s)}$  with probability  $\Pr_{\text{enc}} = \frac{|\tau_{(t,n,s)}|}{|\tau_{(t,s,s)}|}$ . Thus, after  $n$  is fixed, the



**Fig. 1** Illustrative example of the hash operation costs of Wots+ (average) and Wots-cs (fixed) for the same  $t$  and where  $w - 1 = n$  when using MINGEN parameter selection strategy.

algorithm proceeds with a trial-and-error procedure to obtain an encoding in  $\tau_{(t,n,s)}$ . To ensure that it eventually succeeds, for each attempt, the input message is concatenated against a nonce, e.g.,  $d = \mathcal{H}(M \parallel r)$ , until  $\mathbf{E}(d) \in \tau_{(t,n,s)}$ . Algorithm 1 shows the method used in each trial, which degenerates to the deterministic CKY-O when  $n = s$ . We refer the reader to the original publication [14, Section 3.3] for a detailed presentation.

---

**Algorithm 1** Probabilistic encoding CKY-I

---

**Input:**  $t, n, s \in \mathbb{N}$ ,  $I \in \iota_{(t,s,s)}$   
**Output:**  $(b_{t-1}, \dots, b_0) \in \tau_{(t,n,s)}$

```

1: if  $t = 1$  then
2:   return  $(s)$ 
3:  $b \leftarrow 0$ 
4:  $\ell_l \leftarrow 0$ 
5:  $\ell_r \leftarrow 1$ 
6:  $a \leftarrow 1$                                  $\triangleright a \leftarrow |\tau_{(t-1,0,0)}|$ 
7: while not  $\ell_l \leq I < \ell_r$  do
8:    $b \leftarrow b + 1$ 
9:    $a \leftarrow \frac{a(b+t-2)}{b}$                      $\triangleright a \leftarrow |\tau_{(t-1,b,b)}|$ 
10:   $\ell_l \leftarrow \ell_r$ 
11:   $\ell_r \leftarrow \ell_r + a$ 
12: if  $s - b > n$  then
13:   abort
14: return  $(s - b) \parallel \text{CKY-I}(t - 1, n, b, I - \ell_l)$ 

```

---

We now discuss choices of parameters that reduce the costs  $C(\text{SIG})$  or  $C(\text{VER})$  of Wots-cs when compared to Wots+. For this task, we search for the appropriate parameters  $n$  and  $s$ , for a fixed  $t$ . The first approach consists of finding the smallest  $s$  such that  $|\tau_{(t,s-1,s-1)}| < 2^m \leq |\tau_{(t,s,s)}|$ , and subsequently choosing the smallest  $n \leq s$  such that

$2^m \leq |\tau_{(t,n,s)}|$ . We call this approach MINVER, since it achieves the smallest  $C(\text{VER})$  for Wots-cs.

This is evidenced in Table 2, where we compare CKY-O and CKY-I using values of  $t$  taken from Table 1. For example, when  $m = 256$  and  $t = 34$ ,  $C(\text{VER})$  is reduced by 34% if compared to the equivalent case for Wots+ in Table 1. Also, we observe that the case  $n < s$  (CKY-I) yields reduced key generation and signature generation costs when compared to the case  $n = s$  (CKY-O). This is a great improvement by the authors, however, still more than doubling the cost of signature and key generation in the best case.

**Table 2** Number of iterations of  $f_k$  for Wots-cs using MINVER, and success probability of CKY-I given as  $\text{Pr}_{\text{enc}}$ . Recall that  $C(\text{VER}) = s$ .

| $m$ | $t$ | $n$  | $C(\text{GEN})$ | $C(\text{SIG})$ | $C(\text{VER})$ | $\text{Pr}_{\text{enc}}$ |
|-----|-----|------|-----------------|-----------------|-----------------|--------------------------|
| 256 | 67  | 341  | 22847           | 22506           | 341             | 1.00                     |
|     |     | 42   | 2814            | 2473            |                 | $\sim 0.9792$            |
|     | 45  | 952  | 42840           | 41888           | 952             | 1.00                     |
|     |     | 145  | 6525            | 5573            |                 | $\sim 0.9646$            |
|     | 34  | 2832 | 96288           | 93456           | 2832            | 1.00                     |
|     |     | 661  | 22474           | 19642           |                 | $\sim 0.9945$            |
| 512 | 131 | 688  | 90128           | 89440           | 688             | 1.00                     |
|     |     | 47   | 6157            | 5469            |                 | $\sim 0.9756$            |
|     | 89  | 1849 | 164561          | 162712          | 1849            | 1.00                     |
|     |     | 240  | 21360           | 19511           |                 | $\sim 0.9994$            |
|     | 66  | 5855 | 386430          | 380575          | 5855            | 1.00                     |
|     |     | 750  | 49500           | 43645           |                 | $\sim 0.9907$            |

We now propose a new approach for parameter selection based on the result below, which states that for fixed  $t$  and  $n$ , the largest cardinality of  $\tau_{(t,n,s)}$  occurs when  $s = \lceil \frac{tn}{2} \rceil$  or  $s = \lfloor \frac{tn}{2} \rfloor$ .

**Theorem 2** *Let  $n \geq 1$ ,  $t \geq 1$  and  $0 \leq s \leq \lceil \frac{tn}{2} \rceil$ . For any  $0 \leq j \leq s$  we have*

$$|\tau_{(t,n,j)}| \leq |\tau_{(t,n,s)}|.$$

*Proof* See Appendix A.2.

We choose the smallest  $n$  satisfying  $2^m \leq |\tau_{(t,n,\lceil \frac{tn}{2} \rceil)}|$  before we attempt to decrease  $s$ . We call this approach **MIN-GEN**. As a result, the parameter  $s$  is no longer constrained to  $|\tau_{(t,s-1,s-1)}| < 2^m$ , and can be substantially larger than the  $s$  obtained with **MIN-VER**. Even though  $C(\text{VER})$  is not minimal, a clear advantage of this method is that it obtains minimum  $C(\text{GEN})$  that consequently reduces  $C(\text{SIG})$  and  $C(\text{VER})$  at the same time.

We remark that there are limitations in **CKY-I** that prevents the algorithm from taking full advantage of **MIN-GEN**. As we have discussed, **CKY-I** encodes  $\{0, 1\}^m$  to  $\tau_{(t,n,s)}$  by trial-and-error via a mapping to  $\tau_{(t,s,s)}$ . If  $2^m \sim |\tau_{(t,n,s)}|$  but  $2^m$  is much smaller than  $|\tau_{(t,s,s)}|$ , Algorithm 1 is bound to fail with a probability far greater than  $\text{Pr}_{\text{enc}}$ . This happens since Algorithm 1 orders tuples in non-increasing  $b_{t-1}$ , so the first  $2^m$  elements of  $\tau_{(t,s,s)}$  very likely have  $b_{t-1} > n$ , and thus, they are not valid tuples in  $\tau_{(t,n,s)}$ . For example, let  $(t, n, s) = (34, 226, 3643)$  with  $m = 256$ , where  $|\tau_{(t,s,s)}| \approx 2^{267} > 2^{256}$ . Since  $I < 2^{256}$  and  $\sum_{j=0}^{s-(n+1)} |\tau_{(t-1,j,j)}| \approx 2^{264} > 2^{256}$  at the end of the **while** loop, we have that  $b < s - (n + 1)$  which gives  $s - b > n + 1 > n$ , and thus, Algorithm 1 always aborts.

In the next section, we solve these shortcomings by changing the ordering of tuples in Algorithm 1 and providing a deterministic encoding directly into  $\tau_{(t,n,s)}$ . Moreover, we further explore the **MIN-GEN** approach in Sect. 4, where we suggest optimal parameters for **WOTS-CS**.

### 3 Deterministic constant-sum encoding algorithms

In this section, we propose distinct mapping algorithms that eliminate the probabilistic approach used in **CKY-I**. While our methods are interoperable, they do not produce the same outcome as previous works. Going forward, we show how our algorithms allow for a better choice of parameters regarding costs  $C(\text{GEN})$ ,  $C(\text{SIG})$ , and  $C(\text{VER})$ , by using the **MIN-GEN** method.

A known binomial identity is used in [8] to compute  $\ell_r$ , in Algorithm 1, with  $n = s$ . The general case where  $n \leq s$  is given in the next proposition. This result is mentioned in [10, Table 1], for which we give a short proof.

**Proposition 2** *The cardinality of the set  $\tau_{(t,n,s)}$  satisfies*

$$|\tau_{(t,n,s)}| = \sum_{j=0}^n |\tau_{(t-1,n,s-j)}|. \quad (1)$$

*Proof* A  $t$ -tuple  $\mathbf{b} = (b_{t-1}, \dots, b_0) \in \tau_{(t,n,s)}$  if and only if there exists a  $(t-1)$ -tuple  $(b_{t-2}, \dots, b_0) \in \tau_{(t-1,n,s-b_{t-1})}$ . Then we must have that

$$|\tau_{(t,n,s)}| = |\tau_{(t-1,n,s)}| + |\tau_{(t-1,n,s-1)}| + \dots + |\tau_{(t-1,n,s-n)}|$$

by considering all possible values of  $b_{t-1}$ ,  $0 \leq b_{t-1} \leq n$ .  $\square$

In the following subsections, we present each of our encoding algorithms, but refrain from discussing their relative performance. This is addressed in Sect. 4, in which we give experimental results showing that Algorithms 2 and 3 are considerably faster than both techniques from [8, 14], for various choices of  $t$ ,  $n$  and  $s$ .

#### 3.1 Deterministic encoding with minimal parameters

In order to remove the trial-and-error strategy of **CKY-I**, we work on mapping inputs directly to  $\tau_{(t,n,s)}$ . Our proposal, which is called **DCS**, is shown in Algorithm 2, and has a total number of iterations bounded by  $t(n + 1)$ .

---

#### Algorithm 2 Deterministic constant-sum encoding **DCS**

---

**Input:**  $t, n, s \in \mathbb{N}$ ,  $I \in \iota_{(t,n,s)}$

**Output:**  $(b_{t-1}, \dots, b_0) \in \tau_{(t,n,s)}$

---

```

1: if  $t = 1$  then
2:   return  $(s)$ 
3:  $b \leftarrow 0$ 
4:  $\ell_l \leftarrow 0$ 
5:  $\ell_r \leftarrow |\tau_{(t-1,n,s)}|$ 
6: while not  $\ell_l \leq I < \ell_r$  do
7:    $b \leftarrow b + 1$ 
8:    $\ell_l \leftarrow \ell_r$ 
9:    $\ell_r \leftarrow \ell_r + |\tau_{(t-1,n,s-b)}|$ 
10: return  $(b) \parallel \text{DCS}(t-1, n, s-b, I-\ell_l)$ 
```

---

The correctness of **DCS** is assured directly from Proposition 2. Let  $\tau_{(t,n,s)}^{(l)} = \{(j, b_{t-2}, \dots, b_0) \in \tau_{(t,n,s)} : 0 \leq j \leq l\}$  for  $0 \leq l \leq n$ . Then,  $|\tau_{(t,n,s)}^{(l)}| = \sum_{j=0}^l |\tau_{(t-1,n,s-j)}|$ , i.e. the partial sum up to  $l$  from Eq. (1). We give a simplified explanation next. A digest value  $I \in \iota_{(t,n,s)}$  is bounded by  $|\tau_{(t,n,s)}^{(l)}| \leq I < |\tau_{(t,n,s)}^{(l+1)}|$  for some  $l$ . Hence,  $b_{t-1} = l$  and the algorithm continues recursively with new parameters  $(I - |\tau_{(t,n,s)}^{(l)}|) \in \iota_{(t-1,n,s-l)}$ , until all elements of the tuple are determined.

Our method allows for the same parameter space as exemplified in Table 2. However, there is a significant improvement when comparing **CKY-I** and **DCS** with regard to a different parameter strategy. Since Algorithm 2 does not use

$|\tau_{(t,s,s)}|$  to compute  $\ell_l$ , we can search for parameters that yield minimal  $C(\text{GEN})$  by drastically decreasing  $n$ , using the  $\text{MINGEN}$  method. These parameters and associated costs are discussed in Sect. 4 and detailed in Table 3.

Nevertheless, this advantage comes at a price, as we are now challenged with the task of computing large binomials given from Proposition 2, which uses Theorem 1 repeatedly for every iteration of the **while** loop. This elevated complexity was avoided in  $\text{CKY-I}$  by reusing values previously assigned to  $\ell_r$ ; see line 11 of Algorithm 1.

### 3.2 Binary search encoding

Due to the differences between Algorithm 1 and Algorithm 2 described previously, the latter is relatively more complex to compute as a result of repeated large binomial calculations. It is desirable to perform the underlying unranking algorithm more efficiently since constant-sum encodings are calculated in every signature generation and verification.

The task of searching for the aforementioned boundary  $l$  by individually computing and summing terms of Eq. (1) constrains previous algorithms to a linear number of iterations. To efficiently compute  $|\tau_{(t,n,s)}^{(l)}|$ , we manipulate the binomial coefficients used to evaluate  $|\tau_{(t,n,s)}|$  through Proposition 2 by expanding the technique from Theorem 1.

**Proposition 3** For any  $0 \leq l \leq n$ ,

$$|\tau_{(t,n,s)}^{(l)}| = \sum_{i=0}^k (-1)^i \binom{t-1}{i} \times \left[ \binom{s - (n+1)i + t - 1}{t-1} - \binom{s - (n+1)i + t - 2 - l}{t-1} \right],$$

where  $k = \min(t, \lfloor \frac{s}{n+1} \rfloor)$ .

*Proof* See Appendix A.3.

Proposition 3 allows us to compute  $|\tau_{(t,n,s)}^{(l)}|$  directly, for any  $l$ . From this, we maintain the rationale of the previous algorithm, but remove its recursive approach and additionally employ a binary search strategy to determine the boundary  $l$ . Our new method, called  $\text{DBCS}$ , is given in Algorithm 3, with total number of iterations at most  $t \log_2(n+1)$ .

### 3.3 Encoding as part of signature

As seen in the previous sections, the tuples resulting from our deterministic algorithms are not trivially computable. This impacts the performance of operations with signatures. With the intent of reducing such costs, we propose a modification to  $\text{WOTS-CS}$  which replaces the encoding calculation in  $\text{VER}$ , increasing its performance even further, at the expense of increased signature sizes.

#### Algorithm 3 Binary search constant-sum encoding DBCS

---

**Input:**  $t, n, s \in \mathbb{N}$ ,  $I \in \iota_{(t,n,s)}$   
**Output:**  $p \in \tau_{(t,n,s)}$

```

1:  $p \leftarrow (0, \dots, 0)$ 
2: for  $0 \leq i \leq t-1$  do
3:    $c \leftarrow \min(n, s)$ 
4:    $b \leftarrow 0$ 
5:   while  $c > 0$  do
6:      $step \leftarrow \lfloor \frac{c}{2} \rfloor$ 
7:      $j \leftarrow b + step$ 
8:     if  $|\tau_{(t-i,n,s)}^{(j)}| \geq I$  then
9:        $j \leftarrow j + 1$ 
10:     $b \leftarrow j$ 
11:     $c \leftarrow c - step + 1$ 
12:   else
13:      $c \leftarrow step$ 
14:    $p_{t-1-i} \leftarrow b$ 
15:   if  $b > 0$  then
16:      $I \leftarrow I - |\tau_{(t-i,n,s)}^{(b-1)}|$ 
17:      $s \leftarrow s - b$ 
18: return  $p$ 
```

---

We suggest two adjustments to the underlying scheme. In  $\text{SIG}$ , the encoding  $p$  is sent as part of  $\sigma$ . This requires the signature size to be incremented by  $t \lceil \log_2(n+1) \rceil$  bits. In  $\text{VER}$ , we propose to use Algorithm 4, hereafter denoted as  $\text{VCS}$ , to assert the correspondence of  $I$  and  $p$  without searching for boundaries, which could be costly due to the large binomial calculations.

#### Algorithm 4 Constant-sum encoding verification VCS

---

**Input:**  $t, n, s \in \mathbb{N}$ ,  $I \in \iota_{(t,n,s)}$ ,  $p \in \tau_{(t,n,s)}$   
**Output:**  $\mathbf{E}(I) \stackrel{?}{=} p$

```

1: for  $0 \leq i \leq t-1$  do
2:   if not  $|\tau_{(t-i,n,s)}^{(p_i-1)}| \leq I < |\tau_{(t-i,n,s)}^{(p_i)}|$  then
3:     return false
4:    $I \leftarrow I - |\tau_{(t-i,n,s)}^{(p_i-1)}|$ 
5:    $s \leftarrow s - p_i$ 
6: return true
```

---

The method  $\text{VCS}$  is essentially a simplified version of Algorithm 3, since it only checks that the boundary conditions calculated in  $\text{SIG}$  are correct. If any of the  $t$  elements in the tuple  $p$  does not lie in the correct interval given by the implementation of  $\mathbf{E}$ , the signature verification fails. We note that the modifications in the signature structure proposed to use  $\text{VCS}$  do not make it easier for a malicious party to create a forgery since the encoding is a bijective map, and thus a unique representation of the original message. The detailed security claims of the scheme, however, are presented in Sect. 5.

### 3.4 Memoization of intermediate sums

A common technique of dynamic programming may be employed in the methods above, namely DCS, DBCS and VCS, in the form of a look-up table containing intermediate values  $|\tau_{(\cdot,\cdot,\cdot)}|$ . This creates a memory trade-off that hinders the portability of the algorithm to constrained devices, but other situations may benefit from the sharp performance increase. We denote these variant encodings as DCS-M, DBCS-M and VCS-M, respectively.

Storage requirements are in the order of  $s(t-1)$  integers with up to  $m$  bits each for DCS-M. This comes directly from Eq. (1), in which every value from the range  $|\tau_{(\gamma,n,\delta)}|$  with  $1 \leq \gamma \leq t-1$  and  $1 \leq \delta \leq s$  may be needed, and thus stored. The resulting look-up table is partially shaped like an upper triangular matrix. For  $0 < i < \lfloor \frac{s}{n} \rfloor$ , the  $i$ -th row has exactly  $n \times i$  integers, and other rows are completely filled with  $s$  values.

In the case of DBCS-M and VCS-M, a three-dimensional matrix of order  $s(t-1)(n+1)$  is needed, composed of  $m$ -bit integers. In this case, we have access to all partial sums from the range  $|\tau_{(\gamma,n,\delta)}^{(\zeta)}|$  with  $0 \leq \zeta \leq n$ . The resulting matrix is also partially upper triangular. We note that these requirements may be optimized to reduce storage demands and be used as typical implementation strategies. However, depending on the choice of parameters and encoding algorithms, the storage requirement may become impractical.

For instance, we estimate the worst case scenario without optimization for parameters given in Table 3. We assume that each stored value is an arbitrary precision integer with at most  $m$ -bits. Then, if we consider  $m = 256$  and  $t = 67$ , we require roughly 0.8 MB for DCS-M and 13.5 MB for DBCS-M and VCS-M. Now if we let  $t = 34$  for the same  $m$ , the memory requirement increases to 3.8 MB and 870 MB, respectively. We discuss the trade-off associated with these estimates in Sect. 4 along with our benchmark results.

## 4 Parameter choice and benchmarking

This section shows how the choice of parameters and techniques for the constant-sum encoding impacts the efficiency of WOTS-CS. The performance of such algorithms is measured via the number of applications of  $f_k$  needed for the entire scheme, as well as their running time. Additionally, previous work in the literature [8] shows results computed with cryptographic hash functions that are currently considered insecure, such as SHA-1 [17]. We thus consider updated security parameters in our results.

For  $m = 256$  and  $m = 512$ , we search combinations of parameters bounded by  $30 \leq t \leq 80$  and  $60 \leq t \leq 140$ , respectively, such that  $m \lesssim \log_2 |\tau_{(t,n,s)}|$ . From this space, according to the MINGEN approach, we select values of  $n$  and  $s$  that minimize costs of GEN for each value of  $t$ . Table 3

shows the results of the search grouped into usual values of  $t$ , where the cost prepended with a  $\Delta$  represents changes in relation to WOTS+.

**Table 3** Suggested parameters for WOTS-CS using MINGEN for a given  $t$  as compared to WOTS+.

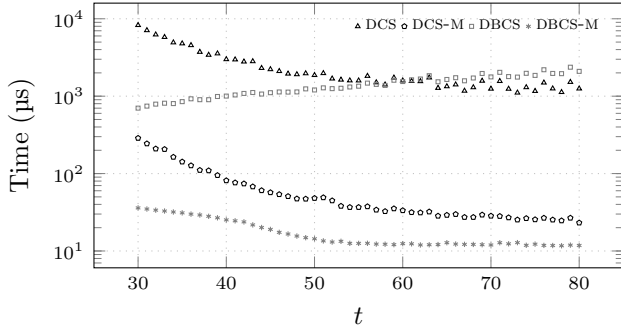
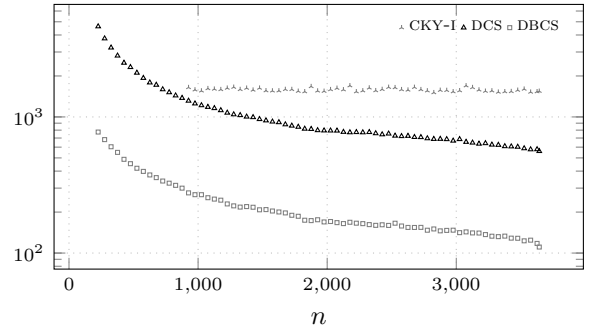
| $m$ | $t$ | $n$ | $C(\text{GEN})$ | $C(\text{SIG})$ | $C(\text{VER})$ | $\Delta C(\text{GEN})$ |
|-----|-----|-----|-----------------|-----------------|-----------------|------------------------|
| 256 | 67  | 15  | 1005            | 605             | 400             | +0.00%                 |
|     | 45  | 58  | 2610            | 1405            | 1205            | -7.94%                 |
|     | 34  | 226 | 7684            | 4041            | 3643            | -11.37%                |
| 512 | 131 | 15  | 1965            | 1120            | 845             | +0.00%                 |
|     | 89  | 57  | 5073            | 2695            | 2378            | -9.52%                 |
|     | 66  | 241 | 15906           | 8227            | 7679            | -5.49%                 |

The dominant term  $t$  in the signature size has direct implications on the number of iterations of  $f_k$ . A trade-off happens if  $t$  is increased, to the extent that the values  $n$  and  $s$  may be reduced. In the literature, parameters are given to obtain an efficient signature verification. For sufficiently small values of  $t$ , e.g.  $t = 34$  for  $m = 256$  and  $t = 66$  for  $m = 512$ , we highlight that, by applying the MINGEN approach, we minimize  $C(\text{GEN})$  and obtain *at the same time* increased performance of GEN, SIG and VER, when compared to WOTS+.

It is also fundamental to discuss the cost of encoding algorithms. In order to perform a fair comparison with the original methods, we give running time results of each algorithm performing encodings in the context of the aforementioned parameter space. We implement CKY-I, DCS, DBCS, VCS and memoized variants using the GMP C++ library. Resulting software is available at [github.com/zambonin/wots-cs](https://github.com/zambonin/wots-cs) and compiled using clang 11.0.0 with -O3 optimization level. Times are measured considering an AMD Ryzen™ 3300X, running at 3.8GHz.

Figure 2 shows the running time of the different methods considered. The y-axis is presented in a logarithmic scale for better readability. We give a twofold discussion as follows, remarking that the base-w computation is absent from the comparisons due to its negligible running time. Figure 2a depicts a comparison of the different techniques proposed in Sect. 3, with sets of parameters as calculated in Table 3. The relative performance of Algorithms DCS and DBCS changes with the size of the target encoding tuples.

The lower complexity of the binary search is more evident when the search space  $n$  is large. Such behavior is expected due to the higher cost of computing  $|\tau_{(t,n,s)}^{(l)}|$  defined in Proposition 3. Algorithm DBCS outperforms DCS when  $t < 58$ , that is, it is much faster for smaller values of  $t$ . For example, encoding into tuples of the set  $\tau_{(34,226,3643)}$  is  $\approx 82.7\%$  faster when compared with DCS

(a) Our proposals with  $n \neq s$  and parameters minimizing  $C(\text{GEN})$ .(b) Encoding to  $\tau_{(34,n,3643)}$  with  $n \in \{226, 276, \dots, 3626, 3643\}$ .**Fig. 2** Comparison of running time between distinct constant-sum encoding techniques, in microseconds, for  $m = 256$ .

We also give remarks on the running time of vcs. Recalling that  $p$  must be available during VER, then this algorithm can be used to verify the correspondence of the signed message with  $p$ . We refrain from showing it in Fig. 2, since it is significantly faster to use this approach, instead of repeatedly encoding the signed message with any of the other proposed algorithms. Indeed, it essentially removes the cost associated with the encoding.

Figure 2b shows the average time to encode a digest into a tuple with fixed  $t$  and  $s$  for different values of  $n$ . For this experiment, we also consider the original probabilistic encoding CKY-I, which fails to yield encodings for lower values of  $n$ , the most interesting cases due to savings in  $C(\text{GEN})$ . This is expected, due to its limitation when  $|\tau_{(t,s,s)}|$  is much larger than  $2^m$ , as seen in Sect. 2.3. Disregarding these sets of parameters, we find that DCS and DBCS are, on average,  $\approx 46\%$  and  $\approx 87\%$  faster than CKY-I, respectively. We also experimentally find that for sets of parameters in Table 3, at least DBCS is again consistently faster.

#### 4.1 Results in Merkle signature schemes

We now demonstrate how Wots-cs performs in the context of a Merkle-based signature scheme, observing the cost related to all variations of the encoding algorithms that we have proposed so far. We compare instantiations of XMss with Wots+ in the leaf nodes, in contrast to the usage of Wots-cs. We modify the existing XMss reference implementation from RFC 8391 [13] to use our algorithms in consonance with this goal.

Table 4 shows running times of distinct combinations of the encoding algorithms for all steps of the signature scheme, called  $C_x(\text{GEN})$ ,  $C_x(\text{SIG})$  and  $C_x(\text{VER})$  for brevity, and additionally highlights the encoding cost for signature operations. The underlying XMss tree has been configured with a height parameter of  $h = 10$  and SHA-256 as the chaining function, i.e. the RFC parameter XMSS\_SHA2\_10\_256. Measurements are done in the same CPU specification as in the previous section and were repeated  $2^3$  times for  $C_x(\text{GEN})$  and  $2^{3+h}$

for  $C_x(\text{SIG})$  and  $C_x(\text{VER})$ , with the average running time shown in the respective columns. We focus on the parameters  $m = 256$  with  $t = 67$  and  $t = 34$ , as given by Tables 1 and 3, with the goal of minimizing  $C_x(\text{GEN})$ .

In the case of  $t = 67$ , the behavior shown in Fig. 2a is evidenced with DCS being slightly faster than DBCS. Minimal differences between all algorithms are observed with regard to  $C_x(\text{GEN})$ , due to the fact that  $w - 1 = n$  from Table 3 and Table 1. The most interesting claim, in this case, is the effect of encoding execution times. Gains to  $C_x(\text{VER})$  are only observed for two cases: DCS-M + vcs-M and DBCS-M + vcs-M. Fortunately, for these parameters, the memory requirement for the memoization is significantly lower when compared to the next case, thus viable for implementations that are not memory constrained.

We recall that the use of vcs entails an increased signature size. When  $t = 67$ ,  $|\sigma|$  is increased by 34 bytes, i.e., 67 four-bit integers. To make a fair comparison with Wots+, we would need to use  $t = 66$ , which implies a similar signature size and  $\approx 1.5\%$  improvement on  $C_x(\text{GEN})$ , as per Table 7. However, since  $C_x(\text{VER})$  is also increased, we do not find this an interesting avenue to pursue. With regard to  $t = 34$ , we achieve significantly reduced  $C_x(\text{GEN})$ . In fact, with the exception of DCS, all other combinations of algorithms provide faster overall running time execution. For this case, it is important to evaluate two distinct aspects.

The memory cost for the algorithms DBCS-M or vcs-M may reach roughly 870 MB in the worst case, limiting their usefulness only for very specific devices where memory is abundant. When using DCS-M, the memory cost reaches nearly 4 MB, which is tolerable in exchange for the extra performance gain in a wider range of platforms. Moreover, we cannot provide a reasonable parameter set if  $t = 33$ , to make up for the 34 extra bytes in the signature when using vcs. Therefore, we highlight that algorithm DBCS still achieves better overall costs, with no extra memory cost and no increment to the signature size.

Overall, we show that there exist sets of parameters where the constant-sum encoding complexity is counterbalanced

**Table 4** Performance of different encoding algorithms in the context of `XMSS_SHA2_10_256`, in tens of millions of CPU cycles ( $\times 10^7$ ). Values in square brackets are the percentage of the cost dedicated solely to encoding.

| Encoding algorithm(s) | $C_x(\text{GEN})$ | $t = 67$          |                   |                   | $t = 34$          |                   |  |
|-----------------------|-------------------|-------------------|-------------------|-------------------|-------------------|-------------------|--|
|                       |                   | $C_x(\text{SIG})$ | $C_x(\text{VER})$ | $C_x(\text{GEN})$ | $C_x(\text{SIG})$ | $C_x(\text{VER})$ |  |
| base-w                | 155               | 0.53 [0.006%]     | 0.08 [0.029%]     | 1163              | 4.12 [0.001%]     | 0.58 [0.005%]     |  |
| DCS                   | 158               | 0.79 [29.86%]     | 0.31 [76.78%]     | 1061              | 4.62 [18.10%]     | 1.35 [62.34%]     |  |
| DBCS                  | 155               | 0.86 [36.08%]     | 0.38 [81.32%]     | 1027              | 3.83 [3.448%]     | 0.61 [21.08%]     |  |
| DCS + VCS             | 157               | 0.79 [30.28%]     | 0.20 [65.77%]     | 1039              | 4.55 [18.41%]     | 0.52 [6.288%]     |  |
| DBCS + VCS            | 161               | 0.88 [35.15%]     | 0.20 [64.89%]     | 1075              | 3.99 [3.318%]     | 0.53 [5.916%]     |  |
| DCS-M                 | 153               | 0.60 [7.122%]     | 0.11 [37.52%]     | 1051              | 3.81 [1.055%]     | 0.53 [6.789%]     |  |
| DBCS-M                | 148               | 0.60 [10.67%]     | 0.13 [48.07%]     | 1032              | 3.77 [0.754%]     | 0.51 [5.040%]     |  |
| DCS-M + VCS-M         | 151               | 0.59 [7.250%]     | 0.07 [5.028%]     | 1059              | 3.82 [1.065%]     | 0.49 [0.716%]     |  |
| DBCS-M + VCS-M        | 152               | 0.62 [10.78%]     | 0.07 [2.967%]     | 1038              | 3.75 [0.712%]     | 0.49 [0.653%]     |  |

so that the scheme is efficient. It should also be noted that  $t = 34$  is not the most commonly used parameter for Wots+ instances. However, it is useful to demonstrate our results, and it has applications in the case of smaller signature requirements that we discussed in Sect. 1.3.

Finally, the experiment shows that some practical limitations restrain us from reaching theoretical improvement estimates. For example, we observe that both DCS and DBCS use a large portion of the verification running time. In fact, this fraction may even increase if we consider implementations that are more optimized for verification running time than the XMSS reference code. For this reason, we believe that more extensive research of constant-sum encoding could provide optimizations that may compensate for this encoding cost, making the scheme more competitive. Alternatively, we discuss, in the following subsection, some applications that can avoid the impact of the encoding costs.

#### 4.2 Alternative parameter choices

To show the benefit of reduced chain lengths for signature generation, in spite of higher encoding costs, it is interesting to point out that not every application will require the message encoding to be performed by the same device that produces the signature. It is common, for signature applications, to perform all cost-intensive computations outside the secure platform that holds the private key. One clear example of this is the analogous case where large documents are digested into small hash outputs before being sent to a secure device in order to be signed, e.g., a smart card, a trusted platform module (TPM) or a hardware security module (HSM).

Some of these applications might benefit from a signature scheme that can be configured to have a small key generation cost and small signature generation cost. For such cases, the cost of the signature verification may be less critical. Hence, we propose the use of the same parameters in Table 3, but exchanging the costs of signature and verification

using Proposition 1. In Table 5, we display the alternative parameters with the changes in relation to Wots+ regarding the cost of signature generation.

**Table 5** Suggested parameters for Wots-CS using MinGen for a given  $t$  as compared to Wots+.

| $m$ | $t$ | $n$ | $C(\text{GEN})$ | $C(\text{SIG})$ | $C(\text{VER})$ | $\Delta C(\text{SIG})$ |
|-----|-----|-----|-----------------|-----------------|-----------------|------------------------|
| 256 | 67  | 15  | 1005            | 400             | 605             | -20.39%                |
|     | 45  | 58  | 2610            | 1205            | 1405            | -14.99%                |
|     | 34  | 226 | 7684            | 3643            | 4041            | -15.96%                |
| 512 | 131 | 15  | 1965            | 845             | 1120            | -13.99%                |
|     | 89  | 57  | 5073            | 2378            | 2695            | -15.17%                |
|     | 66  | 241 | 15906           | 7679            | 8227            | -8.74%                 |

If we carefully examine Wots-CS, there is a clear advantage on the number of hash operations performed during the signing process for all parameter choices. We highlight the 20.39% reduced cost for signature generation when  $t = 67$ . For this particular case, there is a balanced trade-off in the verification time, with 20.39% more hash functions in the verification, when compared to Wots+. For the other cases, all costs related to GEN, SIG and VER are reduced (compare Table 1 with Table 5).

We note that this performance gain may be less effective when used with XMSS. For the same parameters used in the previous benchmark, we obtain XMSS signature costs reduced by only 2–3% when  $t = 67$  (excluding the encoding cost). This is mainly attributed to the predominant cost of key generation, necessary to build the Merkle tree verification path during the signature. In fact, depending on the application, it is possible to optimize the verification path construction by caching tree nodes, such that the key generation cost is mitigated, thus achieving greater performance gains closer to the ones stated in Table 5.

For  $t = 34$ , the improvements are more noticeable. We can use the same parameters in Table 4 and obtain 15% improvement that is closer to the 15.96% described in Table 5, since there is an additional 11.37% improvement for every key generated during the construction of the verification path. In addition, this parameter selection also presents 11% faster key generation and roughly 13% faster key verification (excluding the encoding cost).

Last, we find that it would be more interesting to study the practical results of our improvements with experiments that are tailored for each application. There are several different ways to combine our parameters with different optimization strategies available for XMSS, considering distinct computation and memory capabilities. We leave such studies for consideration in future works.

## 5 Security proof of Wors-cs

In [14], the authors claim to have proved that Wors-cs is existentially unforgeable under an adaptive chosen-message attack (EU-CMA) assuming that the constant-sum encoding function is collision resistant and  $f_k$  is one-way collision resistant. However, we find that their estimation of certain probabilities is incorrect. More specifically, the authors make incorrect implicit assumptions about the uniformity of the probability distribution of tuples controlled by the adversary.

In order to avoid such assumptions, we provide a security proof that Wors-cs is EU-CMA by closely following the argument in [12] and a recent revision of the same security proof in [16]. Several results on the behavior of  $|\tau_{(t,n,s)}|$  are needed to obtain the correct probability bounds of our proof. These results are presented in Appendix A.

In this section, we provide brief security definitions, necessary properties of the distribution of  $\tau_{(t,n,s)}$  and a detailed proof of Wors-cs under the assumption that  $\mathcal{F}_\lambda$  is undetectable and second-preimage resistant one-way function. This result is a major improvement on the previous works by CKY, since the proof does not require collision resistance. In particular, we can instantiate Wors-cs with signatures about half the size when compared to [14]. Moreover, we give a concrete bound of the security level of our scheme, showing that its security level is comparable with Wors+.

### 5.1 Formal security definitions

We first define some terms used throughout the security arguments. In this subsection, we closely follow [12] and we refer the reader to that paper for more details.

**Definition 1 (Negligible [11, Def. 2.1])** A function  $v$  is negligible if for every constant  $c \geq 0$ , there exists an integer  $k_c$  such that  $v(k) < k^{-c}$  for all  $k \geq k_c$ . For short we write “ $v$  is negligible in  $k$ ” as  $\text{negl}(k)$ .

For simplicity, we consider in the following that sampling  $k \leftarrow_{\$} \mathcal{K}_\lambda$  and evaluating functions from  $\mathcal{F}_\lambda$  can be done in polynomial time. Then we can define the success probability of an adversary  $\mathcal{A}$  against the one-wayness (OW) of  $\mathcal{F}_\lambda$  as:

$$\text{Succ}_{\mathcal{F}_\lambda}^{\text{OW}}(\mathcal{A}) = \Pr[k \leftarrow_{\$} \mathcal{K}_\lambda, x \leftarrow_{\$} \{0, 1\}^\lambda, y \leftarrow f_k(x), \\ x' \leftarrow_{\$} \mathcal{A}(k, y) : y = f_k(x')],$$

and against the second preimage resistance (SPR) of  $\mathcal{F}_\lambda$  as:

$$\text{Succ}_{\mathcal{F}_\lambda}^{\text{SPR}}(\mathcal{A}) = \Pr[k \leftarrow_{\$} \mathcal{K}_\lambda, x \leftarrow_{\$} \{0, 1\}^\lambda, \\ x' \leftarrow \mathcal{A}(k, x) : (x \neq x') \wedge (f_k(x) \neq f_k(x'))].$$

Now let  $\text{InSec}$  denote the maximum success probability of all possible probabilistic adversaries  $\mathcal{A}$  running in time less than or equal to  $z$ . We define one-wayness and second preimage resistance as follows:

**Definition 2 (OW)** Let  $\lambda$  be the security parameter, then  $\mathcal{F}_\lambda$  is one-way if

$$\text{InSec}^{\text{OW}}(\mathcal{F}_\lambda; z) = \max_{\mathcal{A}}(\text{Succ}_{\mathcal{F}_\lambda}^{\text{OW}}(\mathcal{A})) = \text{negl}(\lambda).$$

**Definition 3 (SPR)** Let  $\lambda$  be the security parameter, then  $\mathcal{F}_\lambda$  is second preimage resistant if

$$\text{InSec}^{\text{SPR}}(\mathcal{F}_\lambda; z) = \max_{\mathcal{A}}(\text{Succ}_{\mathcal{F}_\lambda}^{\text{SPR}}(\mathcal{A})) = \text{negl}(\lambda).$$

Now we define the undetectability (UD) property of  $\mathcal{F}_\lambda$ . We start by denoting  $\mathcal{U}_\lambda$  as the uniform distribution over  $\lambda$ -bit strings. Then, for distributions over  $\{0, 1\}^\lambda \times \mathcal{K}_\lambda$ , we let a sample  $(u, k)$  from distribution  $\mathcal{D}_{\text{UD}, \mathcal{U}}$  be obtained by choosing  $u \leftarrow_{\$} \mathcal{U}_\lambda$  and  $k \leftarrow_{\$} \mathcal{K}_\lambda$ . Otherwise, we let a sample  $(u, k)$  be obtained from distribution  $\mathcal{D}_{\text{UD}, \mathcal{F}}$  by letting  $u$  as the evaluation of  $f_k$  on a uniformly distributed  $\lambda$ -bit string, where  $k \leftarrow_{\$} \mathcal{K}_\lambda$ . In other words, by allowing a slight abuse of notation, we write  $u \leftarrow f_k(\mathcal{U}_\lambda)$ . From this, we define the distinguishing advantage of an adversary, followed by the undetectability of  $\mathcal{F}_\lambda$ .

**Definition 4 (Advantage)** Given two distributions  $\mathcal{X}$  and  $\mathcal{Y}$ , the advantage  $\text{Adv}_{\mathcal{X}, \mathcal{Y}}(\mathcal{A})$  of an adversary  $\mathcal{A}$  in distinguishing between these two distributions is defined as:

$$\text{Adv}_{\mathcal{X}, \mathcal{Y}}(\mathcal{A}) = |\Pr[1 \leftarrow \mathcal{A}(\mathcal{X})] - \Pr[1 \leftarrow \mathcal{A}(\mathcal{Y})]|.$$

**Definition 5 (UD)** Let  $\lambda$  be the security parameter, then  $\mathcal{F}_\lambda$  is undetectable if

$$\text{InSec}^{\text{UD}}(\mathcal{F}_\lambda; z) = \max_{\mathcal{A}}(\text{Adv}_{\mathcal{D}_{\text{UD}, \mathcal{U}}, \mathcal{D}_{\text{UD}, \mathcal{F}}}(\mathcal{A})) = \text{negl}(\lambda).$$

Finally, we recall the definition by Boneh et al. [5] of a digital signature scheme being existentially unforgeable under an adaptive chosen-message attack (EU-CMA). This is based on a game with three phases, given below.

**Setup.** The challenger runs  $\text{GEN}(1^m)$ , obtains  $(\mathbf{sk}, \mathbf{pk})$ , provides the adversary  $\mathcal{A}$  with  $\mathbf{pk}$  and keeps  $\mathbf{sk}$  to itself.

**Query.** The adversary  $\mathcal{A}$  submits signature queries of messages  $M_1, \dots, M_q$  of its choice to the challenger, who replies with  $\sigma_1 = \text{SIG}(M_1, \mathbf{sk}), \dots, \sigma_q = \text{SIG}(M_q, \mathbf{sk})$ . These queries can be made adaptively by  $\mathcal{A}$ .

**Output.** The adversary outputs a pair  $(M', \sigma')$  and wins the game if  $\text{VER}(M', \sigma', \mathbf{pk})$  is true and  $M' \notin \{M_1, \dots, M_q\}$ .

We recall that  $\mathcal{A}$  is always considered to be a probabilistic polynomial time algorithm. The formal definition of EU-CMA is given below.

**Definition 6 ((EU-CMA [5, Def. 1])** A signature scheme is  $(z, q, \epsilon)$ -existentially unforgeable under an adaptive chosen-message attack if no  $z$ -time adversary  $\mathcal{A}$  making at most  $q$  signature queries has advantage at least  $\epsilon$  in the above game.

It is clear that for the security of one-time signature schemes we only need to consider  $q = 1$ . Thus, for any security parameter  $\lambda$ , our objective is to prove that WOTS-CS is  $(z(\lambda), 1, \epsilon(\lambda))$ -existentially unforgeable for any polynomial  $z(\lambda)$  and any non-negligible probability  $\epsilon(\lambda)$ .

## 5.2 Properties of $\tau_{(t,n,s)}$

We briefly introduce two properties of  $\tau_{(t,n,s)}$  in the form of Proposition 4 and Proposition 5. These results are imperative for the characterization of the probability assertions we make in Theorem 3, providing the security bounds of WOTS-CS. The proofs of these properties are omitted in this section, but can be found in the appendix of the paper, together with other required properties.

We recall the definition of the constant-sum encoding and address its distribution probability with respect to the choice of parameters and the size of the signed message. Remark that  $\mathbf{E} : \iota_{(t,n,s)} \rightarrow \tau_{(t,n,s)}$ , as introduced in Sect. 2, with  $\iota_{(t,n,s)} = \{0, \dots, |\tau_{(t,n,s)}| - 1\}$  and  $2^m \leq |\tau_{(t,n,s)}|$ . Let us call  $\tau_{(t,n,s)}^m$  the image of  $\{0, 1\}^m$  under  $\mathbf{E}$ , namely the evaluation of  $\mathbf{E}(M)$ , allowing a slight abuse of notation as we naturally identify  $\{0, \dots, 2^m - 1\}$  with  $\{0, 1\}^m$ . Because the image of  $\mathbf{E}(M)$  is  $\tau_{(t,n,s)}^m$  instead of  $\tau_{(t,n,s)}$ , we must ensure that  $2^m$  is very close to  $|\tau_{(t,n,s)}|$ , so that we can use results about distribution of tuples in  $\tau_{(t,n,s)}$ , when considering such a distribution in  $\tau_{(t,n,s)}^m$ .

**Proposition 4** Let  $m$  and  $t$  be integers such that  $t = o(m)$ , and let  $n$  be such that

$$|\tau_{(t,n-1, \lfloor \frac{(n-1)t}{2} \rfloor)}| < 2^m \leq |\tau_{(t,n, \lfloor \frac{nt}{2} \rfloor)}|.$$

Let  $\tau_{(t,n, \lfloor \frac{nt}{2} \rfloor)}^m$  be the image of the encoding  $\mathbf{E}(\{0, 1\}^m) \subseteq \tau_{(t,n, \lfloor \frac{nt}{2} \rfloor)}$ . Consider the sequence of independent and uniformly distributed random variables  $X = X_m$  from  $\tau_{(t,n, \lfloor \frac{nt}{2} \rfloor)}$ ,

and the sequence of independent and uniformly distributed random variables  $Y = Y_m$  from  $\tau_{(t,n, \lfloor \frac{nt}{2} \rfloor)}^m$ . Then,  $X$  and  $Y$  are statistically indistinguishable.

*Proof* See Appendix A.4.

Proposition 4 guarantees that taking  $(\lambda, m, t, n, s)$  such that  $m = \text{poly}(\lambda)$ ,  $t = o(m)$ ,  $s = \frac{tn}{2}$  and  $n$  such that

$$|\tau_{(t,n-1, \frac{(n-1)t}{2})}| < 2^m \leq |\tau_{(t,n, \frac{nt}{2})}|$$

yields acceptable parameters, as defined in Definition 7 in Sect. 5.3, since  $X = X_m$  and  $Y = Y_m$  are statistically indistinguishable. This ensures that parameters chosen in our experiments using MINGEN are acceptable and satisfy the requirements for the security of the scheme.

The following proposition gives a probability bound used in the security proof given in Theorem 3.

**Proposition 5** Let  $t, n \geq 2$ ,  $0 \leq \eta \leq n$  and  $s \leq \frac{tn}{2}$ . Let  $(b_{t-1}, \dots, b_0)$  be chosen uniformly at random from  $\tau_{(t,n,s)}$ . Then, for any  $0 \leq j \leq \eta$  and for any  $0 \leq i \leq t-1$ ,

$$\frac{|\tau_{(t-1,n,s-\eta)}|}{|\tau_{(t,n,s)}|} \leq \Pr[b_i = j] \leq \frac{|\tau_{(t-1,n,s)}|}{|\tau_{(t,n,s)}|}.$$

*Proof* See Appendix A.5.

## 5.3 Security proof

We now prove the security of WOTS-CS through the following theorem, which closely follows the arguments in [12] and a more recent revision of the same proof framework [16]. We remark that the proof is given only for constant-sum encoding algorithms compatible with parameters given in Definition 7.

**Definition 7** For (growing) security parameter  $\lambda$  the parameters  $(\lambda, m, t, n, s)$  are acceptable if  $m = \text{poly}(\lambda)$ ,  $t \geq 2$ ,  $n \geq 2$ ,  $2n-1 \leq s \leq \frac{tn}{2}$ ,  $2^m \leq |\tau_{(t,n,s)}|$  and the uniform distribution  $X = X_m$  on  $\tau_{(t,n,s)}$  is computationally indistinguishable from the uniform distribution  $Y = Y_m$  on  $\tau_{(t,n,s)}^m$ .

**Theorem 3** Let  $(\lambda, m, t, n, s)$  be acceptable WOTS-CS parameters where  $\lambda$  is the security parameter. Let  $\mathcal{F}_\lambda = \{f_k : \{0, 1\}^\lambda \rightarrow \{0, 1\}^\lambda, k \in \mathcal{K}_\lambda\}$  be a one-way, second preimage resistant and undetectable function family. The insecurity of WOTS-CS scheme against an EU-CMA attack satisfies

$$\begin{aligned} \text{InSec}^{\text{EU-CMA}}(\text{WOTS-CS}(\lambda, m, t, n, s); z, 1) \\ \leq \frac{t \cdot |\tau_{(t,n,s)}|}{|\tau_{(t-1,n,s-(n-1))}|} \left( n \cdot \text{InSec}^{\text{UD}}(\mathcal{F}_\lambda; \tilde{z}) \right. \\ \left. + \text{InSec}^{\text{OW}}(\mathcal{F}_\lambda; z') + n \cdot \text{InSec}^{\text{SPR}}(\mathcal{F}_\lambda; z') \right), \end{aligned}$$

with  $\tilde{z} = z + 3tn + n$  and  $z' = z + 3tn$  and where time is given in number of evaluations of the function  $f_k$  in  $\mathcal{F}_\lambda$ .

*Proof* Assume, for the sake of contradiction, that there exists an adversary  $\mathcal{A}$  that can produce existential forgeries for  $\text{WOTS-CS}(\lambda, m, t, n, s)$  by running an adaptive chosen message attack in time at most  $z$  and with success probability  $\epsilon_{\mathcal{A}} \equiv \text{Succ}_{\text{WOTS-CS}(\lambda, m, t, n, s)}^{\text{EU-CMA}}(\mathcal{A})$  that is greater than the insecurity bound claimed by the theorem. Then, we construct an oracle machine  $\mathcal{M}^{\mathcal{A}}$  that either breaks OW or SPR properties of  $\mathcal{F}_{\lambda}$  using  $\mathcal{A}$ .

---

**Algorithm 5**  $\mathcal{M}^{\mathcal{A}}$ 


---

**Input:**  $\lambda, k \in \mathcal{K}$  and challenges  $y_c, x_c \in \{0, 1\}^{\lambda}$ .  
**Output:** A value  $x$  that is either a preimage of  $f_k(y_c)$  or a second preimage of  $f_k(x_c)$  or fail.  
1:  $(\mathbf{sk}, \mathbf{pk}) \leftarrow \text{GEN}(1^{\lambda})$ .  
2:  $\alpha \leftarrow_{\$} \{0, \dots, t-1\}$   
3:  $M_0 \leftarrow_{\$} \{L \in \{0, 1\}^m : (a_{t-1}, \dots, a_0) = \mathbf{E}(L) \wedge a_{\alpha} \neq n\}$   
4:  $a = (a_{t-1}, \dots, a_0) \leftarrow \mathbf{E}(M_0)$   
5:  $\beta = a_{\alpha}$  ▷ Note that  $\beta \neq n$   
6: **if**  $\beta = 0$  **then**  
7:    $r' \leftarrow r$   
8: **else**  
9:    $\gamma \leftarrow_{\$} \{0, \dots, \beta-1\}$   
10:    $r' = r$  and replace  $r'_{\gamma} = c_k^{\beta-\gamma-1}(y_c, r_{n-\beta+1, n}) \oplus x_c$ .  
11: Obtain  $\mathbf{pk}'_i$  where  $\mathbf{pk}'_i = c_k^n(\mathbf{sk}_i, r')$  for  $0 \leq i \leq t-1, i \neq \alpha$  and  $\mathbf{pk}'_{\alpha} = c_k^{\beta}(y_c, r'_{n-\beta+1, n})$ .  
12: Receive  $M$  from  $\mathcal{A}$  in the query phase and  $(b_{t-1}, \dots, b_0) \leftarrow \mathbf{E}(M)$   
13: **if**  $b_{\alpha} > \beta$  **then**  
14:   **return** fail  
15:  $\sigma = \text{SIG}(M, \mathbf{sk}, r')$   
16: Replace  $\sigma_{\alpha} = c_k^{\beta-b_{\alpha}}(y_c, r'_{n-\beta+1, n})$   
17: Reply to query with  $\sigma$  and obtain  $(M', \sigma')$  from  $\mathcal{A}$   
18: **if**  $(M', \sigma')$  is a valid signature **then**  
19:   Let  $(b'_{t-1}, \dots, b'_0) \leftarrow \mathbf{E}(M')$   
20:   **if**  $b'_{\alpha} \leq \beta$  **then**  
21:     **return** fail  
22:   **else if**  $\beta = 0$  **or**  $c_k^{b'_{\alpha}-\beta}(\sigma'_{\alpha}, r'_{n-b'_{\alpha}+1, n}) = y_c$  **then**  
23:     **return** preimage  $x = c_k^{b'_{\alpha}-\beta-1}(\sigma'_{\alpha}, r'_{n-b'_{\alpha}+1, n}) \oplus r_{n-\beta}$   
24:   **else if**  $x = c_k^{b'_{\alpha}-\gamma-1}(\sigma'_{\alpha}, r'_{n-b'_{\alpha}+1, n}) \oplus r_{n-\gamma} \neq x_c$   
25:     **and**  $c_k^{b'_{\alpha}-\gamma}(\sigma'_{\alpha}, r'_{n-b'_{\alpha}+1, n}) = c_k^{\beta-\gamma}(y_c, r'_{n-\beta+1, n})$  **then**  
26:       **return** second preimage  $x$   
26: **return** fail

---

We let  $\tilde{\epsilon}_{\mathcal{A}}$  be the probability that  $\mathcal{M}^{\mathcal{A}}$  (given in Algorithm 5) arrives at line 22 by calling  $\mathcal{A}$ . That is

$$\tilde{\epsilon}_{\mathcal{A}} = \Pr[b_{\alpha} \leq \beta \wedge \text{“Forgery is valid”} \wedge b'_{\alpha} > \beta],$$

where “Forgery is valid” is 1 if and only if  $\text{VER}(M', \sigma', \mathbf{pk})$  succeeds and  $M' \neq M$ , or 0 otherwise.

From line 22 and forward, there are two possibilities:

1. either  $\beta = 0$  or the intermediate steps of the verification of  $\sigma'_{\alpha}$  contains a preimage of  $y_c$ . Then,  $\mathcal{M}^{\mathcal{A}}$  returns preimage  $x$  with probability 1.
2.  $\beta > 0$  and the intermediate steps of the verification of  $\sigma'_{\alpha}$  do not contain a preimage of  $y_c$ . Then,  $\mathcal{M}^{\mathcal{A}}$  returns

a second preimage for  $x_c$ , if the chaining function over  $\sigma'_{\alpha}$  collides with the chaining function of  $y_c$  at position  $n - \gamma$ . Since we have picked  $\gamma$  uniformly at random in  $\{0, \dots, \beta-1\}$ , this event occurs with probability  $\frac{1}{\beta}$  lower bounded by  $\frac{1}{n}$ .

Then, by considering that the first case occurs with probability  $p$  and the other with probability  $1 - p$ , we use the OW and SPR assumptions to upper bound  $\tilde{\epsilon}_{\mathcal{A}}$  as

$$p \cdot \tilde{\epsilon}_{\mathcal{A}} \leq \text{InSec}^{\text{OW}}(\mathcal{F}_{\lambda}; z')$$

for the first case, and

$$(1 - p) \frac{\tilde{\epsilon}_{\mathcal{A}}}{n} \leq \text{InSec}^{\text{SPR}}(\mathcal{F}_{\lambda}; z')$$

for the second. The new  $z' = z + 3tn$  is an upper bound of the time  $\mathcal{M}^{\mathcal{A}}$  takes to setup, sign and call  $\mathcal{A}$ . Finally, we sum both equations and obtain

$$\tilde{\epsilon}_{\mathcal{A}} \leq \text{InSec}^{\text{OW}}(\mathcal{F}_{\lambda}; z') + n \cdot \text{InSec}^{\text{SPR}}(\mathcal{F}_{\lambda}; z'). \quad (2)$$

In the remainder of the proof, we derive a lower bound on  $\tilde{\epsilon}_{\mathcal{A}}$  as a function of  $\epsilon_{\mathcal{A}}$ , considering the possibility that  $\mathcal{A}$  might be able to distinguish the challenges inserted by  $\mathcal{M}^{\mathcal{A}}$ . In this case,  $\mathcal{A}$  may intentionally provide a message  $M$  such that  $b_{\alpha} > \beta$  or a forgery where  $b'_{\alpha} \leq \beta$ , thus making the probability  $\tilde{\epsilon}_{\mathcal{A}}$  much smaller than  $\epsilon_{\mathcal{A}}$ .

Let two distributions  $\mathcal{D}_{\mathcal{M}}$  and  $\mathcal{D}_{\text{GEN}}$  over  $\{0, \dots, n-1\} \times \{0, 1\}^{\lambda} \times \{0, 1\}^{\lambda \times n} \times \mathcal{K}_{\lambda}$ . A sample  $\{\beta, u, r, k\} \in \mathcal{D}_{\mathcal{M}}$  is equivalent to choosing  $\beta$  as described in lines 2–5 of  $\mathcal{M}^{\mathcal{A}}$ , and  $u, r, k$  uniformly at random. A sample  $\{\beta, u, r, k\} \in \mathcal{D}_{\text{GEN}}$  corresponds to choosing  $\beta$  as described in lines 2–5 of  $\mathcal{M}^{\mathcal{A}}$ , choosing  $r, k$  uniformly at random and  $u = c_k^{n-\beta}(x, r)$  with  $x \leftarrow_{\$} \{0, 1\}^m$ . That is,  $\mathcal{D}_{\mathcal{M}}$  corresponds to the elements generated by  $\mathcal{M}^{\mathcal{A}}$ , while  $\mathcal{D}_{\text{GEN}}$  corresponds to the elements generated in one WOTS-CS signature chain up to  $n - \beta$ .

Let  $\mathcal{M}'^{\mathcal{A}}$  be a new algorithm that is similar to  $\mathcal{M}^{\mathcal{A}}$ , but takes input from  $\mathcal{D}_{\mathcal{M}}$  or  $\mathcal{D}_{\text{GEN}}$  where  $y_c = u$  (see Appendix A.7). Then,  $\mathcal{M}'^{\mathcal{A}}$  behaves as  $\mathcal{M}^{\mathcal{A}}$  up until line 20, where if condition  $b'_{\alpha} \leq \beta$  is not satisfied, returns 1, and otherwise returns 0. Hence, given inputs from  $\mathcal{D}_{\mathcal{M}}$ ,  $\mathcal{M}'^{\mathcal{A}}$  outputs 1 with probability  $\tilde{\epsilon}_{\mathcal{A}}$ , identical to  $\mathcal{M}^{\mathcal{A}}$ . However, given inputs from  $\mathcal{D}_{\text{GEN}}$ , the probability that  $\mathcal{M}'^{\mathcal{A}}$  outputs 1 is

$$\hat{\epsilon}_{\mathcal{A}} \equiv \Pr[b_{\alpha} \leq \beta \wedge \text{“Forgery is valid”} \wedge b'_{\alpha} > \beta]$$

with  $\Pr[\text{“Forgery is valid”}] = \epsilon_{\mathcal{A}}$ . This holds, since, in this case,  $\mathcal{A}$  receives information from a legitimate WOTS-CS signature, whereas in the previous case  $\mathcal{A}$  receives tampered data. Then, we can rewrite the previous equation as

$$\begin{aligned} \epsilon_{\mathcal{A}} \cdot \Pr[b_{\alpha} \leq \beta \wedge b'_{\alpha} > \beta \mid \text{“Forgery is valid”}] \\ \geq \epsilon_{\mathcal{A}} \cdot \Pr[b_{\alpha} = \beta \wedge b'_{\alpha} > \beta \mid \text{“Forgery is valid”}]. \end{aligned}$$

Now we may evaluate each probability of the right-hand side of the inequality individually. Let us consider a random variable  $X = |\{i : 0 \leq i \leq t-1, b_i < n\}|$  under the condition “Forgery is valid”. That is, the number of elements in  $\mathbf{E}(M)$  at the query phase of  $\mathcal{M}'^A$  that are not equal to  $n$  and produce a valid forgery if  $\sigma$  is sent to  $\mathcal{A}$ . Then, recalling that  $\alpha \leftarrow_{\$} \{0, \dots, t-1\}$ , by Proposition 5 taking  $j = \beta$  and  $\eta = n-1$ , we have

$$\Pr[b_\alpha = \beta \mid \text{“Forgery is valid”}] \geq \frac{X}{t} \cdot \frac{|\tau_{(t-1, n, s-(n-1))}|}{|\tau_{(t, n, s)}|}.$$

Furthermore, due to the properties of the constant-sum encoding, we must have that at least one element in  $\mathbf{E}(M')$  has increased when compared to  $\mathbf{E}(M)$ . These elements then must be at least one among all  $X$  elements, and thus by picking  $\alpha$  uniformly at random we have that

$$\Pr[b'_\alpha > b_\alpha \mid b_\alpha = \beta \wedge \text{“Forgery is valid”}] \geq \frac{1}{X}.$$

By putting together both evaluations we obtain

$$\hat{\epsilon}_A \geq \frac{\epsilon_A}{t} \cdot \frac{|\tau_{(t-1, n, s-(n-1))}|}{|\tau_{(t, n, s)}|}$$

For the final part of the proof, we observe that distinguishing distributions  $\mathcal{D}_M$  and  $\mathcal{D}_{\text{GEN}}$  with  $\mathcal{M}'^A$  is given by

$$\text{Adv}_{\mathcal{D}_M, \mathcal{D}_{\text{GEN}}}(\mathcal{M}'^A) = |\hat{\epsilon}_A - \tilde{\epsilon}_A|.$$

It also follows from  $\mathcal{M}'^A$  that  $\hat{\epsilon}_A$  is an upper bound of  $\tilde{\epsilon}_A$ . Hence, since  $\hat{\epsilon}_A \geq \tilde{\epsilon}_A$ , using the previous two equations, we get

$$\epsilon_A \leq \frac{t \cdot |\tau_{(t, n, s)}|}{|\tau_{(t-1, n, s-(n-1))}|} \left( \text{Adv}_{\mathcal{D}_M, \mathcal{D}_{\text{GEN}}}(\mathcal{M}'^A) + \tilde{\epsilon}_A \right). \quad (3)$$

Now we employ the hybrid argument method to bound this advantage by the undetectability of  $\mathcal{F}_\lambda$ . Let

$$\text{Adv}_{\mathcal{D}_M, \mathcal{D}_{\text{GEN}}}(\mathcal{M}'^A) = \sum_{\beta'=0}^{n-1} \Pr[\beta = \beta'] \cdot \text{Adv}_{\mathcal{D}_M^{\beta=\beta'}, \mathcal{D}_{\text{GEN}}^{\beta=\beta'}}(\mathcal{M}'^A)$$

where  $\mathcal{D}_M^{\beta=\beta'}$  and  $\mathcal{D}_{\text{GEN}}^{\beta=\beta'}$  denote the distributions  $\mathcal{D}_M$  and  $\mathcal{D}_{\text{GEN}}$  with element  $\beta$  fixed to  $\beta'$ . This implies that there must exist at least one  $\beta^*$  such that

$$\text{Adv}_{\mathcal{D}_M^{\beta=\beta^*}, \mathcal{D}_{\text{GEN}}^{\beta=\beta^*}}(\mathcal{M}'^A) \geq \text{Adv}_{\mathcal{D}_M, \mathcal{D}_{\text{GEN}}}(\mathcal{M}'^A).$$

Then, we define hybrids  $H_i = (\beta^*, c_k^{n-\beta^*-i}(x, r_{i+1, n}), r, k)$  with  $0 \leq i \leq n - \beta^*$ ,  $r$  and  $k$  uniformly at random from their corresponding spaces and  $x \leftarrow_{\$} \{0, 1\}^\lambda$ . Hence, we can rewrite the previous inequality as

$$\text{Adv}_{H_{n-\beta^*}, H_0}(\mathcal{M}'^A) \geq \text{Adv}_{\mathcal{D}_M, \mathcal{D}_{\text{GEN}}}(\mathcal{M}'^A).$$

since  $H_{n-\beta^*}$  and  $H_0$  coincide with  $\mathcal{D}_M^{\beta=\beta^*}$  and  $\mathcal{D}_{\text{GEN}}^{\beta=\beta^*}$ , respectively. The hybrid argument states that there must exist two

consecutive distributions  $H_{i^*}$  and  $H_{i^*+1}$  with  $0 \leq i^* < n - \beta^*$  such that

$$\begin{aligned} \text{Adv}_{H_{i^*}, H_{i^*+1}}(\mathcal{M}'^A) &\geq \frac{1}{n - \beta^*} \text{Adv}_{\mathcal{D}_M, \mathcal{D}_{\text{GEN}}}(\mathcal{M}'^A) \\ &\geq \frac{1}{n} \text{Adv}_{\mathcal{D}_M, \mathcal{D}_{\text{GEN}}}(\mathcal{M}'^A). \end{aligned}$$

Finally, assuming that  $\mathcal{A}$  can distinguish  $H_0$  and  $H_{n-\beta^*}$ , we construct a machine  $\mathcal{M}''^A$  that uses  $\mathcal{M}'^A$  to break the undetectability of  $\mathcal{F}_\lambda$  (see Appendix A.7). Machine  $\mathcal{M}''^A$  takes security parameter  $\lambda$  and a distinguishing challenge  $(u, k)$ . Then,  $\mathcal{M}''^A$  chooses  $r$  uniformly at random and calls  $\mathcal{M}'^A$  with  $(\beta^*, c_k^{n-\beta^*-(i^*+1)}(u, r_{i^*+2, n}), r, k)$  as input and returns its result. Denote a sample from distribution  $\mathcal{D}_{\text{UD}, \mathcal{U}}$  as taking  $u \leftarrow_{\$} \mathcal{U}_\lambda$  and  $k \leftarrow_{\$} \mathcal{K}_\lambda$  and a sample from  $\mathcal{D}_{\text{UD}, \mathcal{F}}$  as choosing  $k \leftarrow_{\$} \mathcal{K}_\lambda$  and  $u \leftarrow f_k(\mathcal{U}_\lambda)$ . Then, a sample  $(u, k)$  taken from  $\mathcal{D}_{\text{UD}, \mathcal{U}}$  entails that  $c_k^{n-\beta^*-(i^*+1)}(\mathcal{U}_\lambda, r_{i^*+2, n})$  is distributed exactly like the second element of the tuple  $H_{i^*+1}$ . Furthermore, a sample  $(u, k)$  from  $\mathcal{D}_{\text{UD}, \mathcal{F}}$  implies that

$$\begin{aligned} &c_k^{n-\beta^*-(i^*+1)}(f_k(\mathcal{U}_\lambda), r_{i^*+2, n}) \\ &= c_k^{n-\beta^*-(i^*+1)+1}(\mathcal{U}_\lambda \oplus r_{i^*+1}, r_{i^*+1, n}) \\ &= c_k^{n-\beta^*-i^*}(\mathcal{U}_\lambda, r_{i^*+1, n}) \end{aligned}$$

is distributed exactly like the second element of the tuple  $H_{i^*}$ , where we use the fact that the  $\oplus$  operator of a uniformly distributed value with some other value with arbitrary distribution results in a uniformly distributed value. From this we have that

$$\text{Adv}_{\mathcal{D}_{\text{UD}, \mathcal{U}}, \mathcal{D}_{\text{UD}, \mathcal{F}}}(\mathcal{M}''^A) = \text{Adv}_{H_{i^*}, H_{i^*+1}}(\mathcal{M}'^A)$$

and furthermore

$$\text{Adv}_{\mathcal{D}_{\text{UD}, \mathcal{U}}, \mathcal{D}_{\text{UD}, \mathcal{F}}}(\mathcal{M}''^A) \leq \text{InSec}^{\text{UD}}(\mathcal{F}_\lambda; \tilde{z})$$

with bounded running time  $\tilde{z} = z + 3tn + n$ . This bound is obtained from the running time cost of running  $\mathcal{M}'^A$  that is the same as the previously calculated running time  $z' = z + 3tn$ , in addition to at most  $n-1$  calls to  $f_k$  for  $\mathcal{M}''^A$  to initialize the input before calling  $\mathcal{M}'^A$ . Lastly, by joining these last results, we have

$$\text{Adv}_{\mathcal{D}_M, \mathcal{D}_{\text{GEN}}}(\mathcal{M}'^A) \leq n \cdot \text{InSec}^{\text{UD}}(\mathcal{F}_\lambda; \tilde{z}).$$

that yields

$$\epsilon_A \leq \frac{t \cdot |\tau_{(t, n, s)}|}{|\tau_{(t-1, n, s-(n-1))}|} \left( n \cdot \text{InSec}^{\text{UD}}(\mathcal{F}_\lambda; \tilde{z}) + \tilde{\epsilon}_A \right)$$

by using (3). Finally, by applying (2) we obtain the bound on  $\epsilon_A$  that leads to the contradiction.  $\square$

### 5.4 Security level

We now follow the same argument as in [12, 16] to compute the security level of our variant signature scheme. WOTS-CS has security level  $q$  if a successful attack is expected to undertake, on average,  $2^{q-1}$  evaluations of the one-way function from the family  $\mathcal{F}_\lambda$ . Hence, we derive a lower bound for  $z$  by considering

$$\text{InSec}^{\text{EU-CMA}}(\text{WOTS-CS}(\lambda, m, t, n, s); z, 1) \geq \frac{1}{2}.$$

We now use the same assumptions as related works [12, 16] regarding the insecurity of  $\mathcal{F}_\lambda$ , following that

$$\text{InSec}^{\text{UD}}(\mathcal{F}_\lambda; z) = \text{InSec}^{\text{OW}}(\mathcal{F}_\lambda; z) = \text{InSec}^{\text{SPR}}(\mathcal{F}_\lambda; z) = \frac{z}{2^t}.$$

Furthermore, since  $z \gg 4tn$  for practical attacks, we can use the fact that  $z \approx \tilde{z} \approx z'$  with negligible difference. We solve the security level bound for  $z$  as

$$\begin{aligned} \frac{t \cdot |\tau_{(t,n,s)}|}{|\tau_{(t-1,n,s-(n-1))}|} \left( n \frac{z}{2^\lambda} + \frac{z}{2^\lambda} + n \frac{z}{2^\lambda} \right) &\geq \frac{1}{2} \\ \frac{t \cdot |\tau_{(t,n,s)}|}{|\tau_{(t-1,n,s-(n-1))}|} (2n+1) \frac{z}{2^\lambda} &\geq \frac{1}{2}. \end{aligned}$$

Finally, we have that

$$\begin{aligned} z &\geq 2^\lambda \cdot \frac{1}{2} \cdot \frac{1}{(2n+1)} \cdot \frac{|\tau_{(t-1,n,s-(n-1))}|}{t \cdot |\tau_{(t,n,s)}|} \\ z &\geq 2^{\lambda+1+\log_2(|\tau_{(t-1,n,s-(n-1))}|) - \log_2(t(2n+1)|\tau_{(t,n,s)}|)} \end{aligned}$$

yields the security bound

$$q \geq 2^{\lambda+\log_2(|\tau_{(t-1,n,s-(n-1))}|) - \log_2(t(2n+1)|\tau_{(t,n,s)}|)}$$

against classical adversaries and

$$q \geq 2^{\frac{\lambda}{2} + \log_2(|\tau_{(t-1,n,s-(n-1))}|) - \log_2(t(2n+1)|\tau_{(t,n,s)}|)}$$

against quantum adversaries.

**Table 6** Security level  $q$  for WOTS+ and WOTS-CS and  $\lambda = m = 256$ .

| Adversary | $t$ | $w$ | $n$ | $s$  | WOTS+  | WOTS-CS |
|-----------|-----|-----|-----|------|--------|---------|
| Classical | 34  | 256 | 226 | 3643 | 233.91 | 233.99  |
|           | 67  | 16  | 15  | 400  | 240.89 | 240.19  |
| Quantum   | 34  | 256 | 226 | 3643 | 105.91 | 105.99  |
|           | 67  | 16  | 15  | 400  | 112.89 | 112.19  |

To give a better idea of the security level, we compare our bound to the ones given in [16] for classical and quantum attacks. We observe that we consider  $\frac{\lambda}{2}$  for quantum attackers using Grover's algorithm, as also presented in previous works. The parameters in Table 6 match those of Table 1 and Table 2 with  $m = 256$  for the respective value of  $t$ . The

WOTS+ security level against classical and quantum attacks, according to [16], are given respectively by

$$\begin{aligned} q &> \lambda - \log_2(tw) - \log_2(2w+1), \\ q &> \frac{\lambda}{2} - \log_2(tw) - \log_2(2w+1). \end{aligned}$$

## 6 Conclusion and future work

We introduce new deterministic encoding algorithms for the constant-sum variant of the Winternitz one-time signature scheme, WOTS-CS. Our methods have reduced costs associated with the encoding function by employing distinct techniques such as dynamic programming and binary search. Not only do we achieve faster encoding than the probabilistic alternative in the literature, but we expand the sets of parameters that can be used with WOTS-CS, by aiming to reduce the cost of key generation for hash-based signatures.

As a result, our approach allows for a flexible trade-off between costs of key generation, signature generation and signature verification, by accepting both strategies for selecting parameters: MINVER or MINGEN. Our approach allows us to reduce the number of applications of the chaining function for the key generation step, achieving better costs that are more competitive to the ones obtained with WOTS+. We emphasize that reducing the cost of key generation is particularly interesting, since it decreases the overall cost for all signature-related steps and is not covered by other constant-sum-related works [8, 14].

For specific sets of parameters, our parameter selection can yield faster key generation, signature generation and signature verification simultaneously, when compared to WOTS+; see Table 3. Indeed, these results are reflected in other hash-based signature schemes, such as XMSS; see  $t = 34$  in Table 4. We also observe that the reduction factor in the key generation cost is invariant to the height of the XMSS tree, that is, the number of leaves containing a Winternitz scheme instance. Experiments in Table 4 show the potential improvements of WOTS-CS for all sets of parameters, if better encoding algorithms are discovered.

Alternatively, we propose distinct parameters that can be used with specific applications, reducing the cost of signature generation instead of verification. This is a relevant scenario for many cases related to the signature of electronic documents. We argue that signing devices can take advantage of the predictable and reduced cost for signing without having to deal with the encoding cost (Sect. 4.2).

Another significant contribution of our work is the comprehensive study of the constant-sum tuples. We prove several interesting and fundamental properties that are crucial for the security analysis of the scheme. Under a wide range of acceptable parameters, we prove that WOTS-CS is EU-CMA provided that  $\mathcal{F}_\lambda$  is a one-way, second preimage resistant

and undetectable function family. This is a major improvement of [14], placing Wors-cs on pair with the security of Wors+ for the same signature sizes. Previously, the constant-sum scheme security relied on stronger assumptions such as collision resistance.

## 6.1 Open problems and future work

At the end of Sect. 4.2, we indicated that alternative parameter combinations that reduce SIG in the context of XMSS could be tailored for specific applications. Such studies are an interesting avenue for further research.

While we believe that several applications may benefit from our contributions, we acknowledge that the encoding performance needs to be improved for widespread applicability. This obstacle could be overcome by future research on more efficient encoding algorithms for the constant-sum tuples. It may be possible to adapt combinatorial algorithms that apply to classic problems such as unranking a generalized composition of a number [15]. The constant-sum encoding functions studied so far use a lexicographical order of tuples in  $\tau_{(t,n,s)}$ . Future work could improve encoding performance for this ordering or could use a different ordering of tuples in  $\tau_{(t,n,s)}$  that may lead to more efficient encoding algorithms. This research is beyond the scope of this article, but could potentially make Wors-cs more widely applicable.

## References

1. Alagic, G., Alperin-Sheriff, J., Apon, D., Cooper, D., Dang, Q., Kelsey, J., Liu, Y.K., Miller, C., Moody, D., Peralta, R., Perlner, R., Robinson, A., Smith-Tone, D.: Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. Internal Report 8309, National Institute of Standards and Technology (NIST) (2020)
2. Aumasson, J.P., Bernstein, D.J., Beullens, W., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Hülsing, A., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Westerbaan, B.: SPHINCS<sup>+</sup>. Submission to the Third Round of the NIST Post-Quantum Cryptography Standardization Process (2020). URL <https://sphincs.org/data/sphincs+-round3-submission-nist.zip>
3. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS<sup>+</sup> Signature Framework. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019)
4. Bollinger, R.C., Burchard, C.L.: Lucas's Theorem and Some Related Results for Extended Pascal Triangles. *The American Mathematical Monthly* **97**(3), 198–204 (1990)
5. Boneh, D., Shen, E., Waters, B.: Strongly Unforgeable Signatures Based on Computational Diffie-Hellman. In: M. Yung, Y. Dodis, A. Kiayias, T. Malkin (eds.) *Public Key Cryptography - PKC 2006, Lecture Notes in Computer Science*, vol. 3958, pp. 229–240 (2006)
6. Bos, J.W., Hülsing, A., Renes, J., van Vredendaal, C.: Rapidly Verifiable XMSS Signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems* **2021**(1), 137–168 (2020)
7. Cooper, D.A., Apon, D.C., Dang, Q.H., Davidson, M.S., Dworkin, M.J., Miller, C.A.: Recommendation for Stateful Hash-Based Signature Schemes. NIST Special Publication 800-208, National Institute of Standards and Technology (NIST) (2020)
8. Cruz, J.P., Yatani, Y., Kaji, Y.: Constant-Sum Fingerprinting for Winternitz One-Time Signature. In: 2016 International Symposium on Information Theory and Its Applications (ISITA), pp. 703–707 (2016)
9. Eger, S.: Stirling's Approximation for Central Extended Binomial Coefficients. *The American Mathematical Monthly* **121**(4), 344–349 (2014)
10. Fahssi, N.E.: Polynomial Triangles Revisited (2012). URL <https://arxiv.org/abs/1202.0228v7>
11. Goldwasser, S., Bellare, M.: *Lecture Notes on Cryptography* (2008). URL <https://cseweb.ucsd.edu/~mihir/papers/gb.pdf>
12. Hülsing, A.: W-OTS<sup>+</sup> – Shorter Signatures for Hash-Based Signature Schemes. In: A. Youssef, A. Nitaj, A.E. Hassanien (eds.) *Progress in Cryptology – AFRICACRYPT 2013, Lecture Notes in Computer Science*, vol. 7918, pp. 173–188 (2013)
13. Hülsing, A., Butin, D., Gazdag, S.L., Rijneveld, J., Mohaisen, A.: XMSS: Extended Hash-Based Signatures. Request for Comments 8391, Internet Engineering Task Force (2018). URL <https://tools.ietf.org/html/rfc8391>
14. Kaji, Y., Cruz, J.P., Yatani, Y.: Hash-Based Signature with Constant-Sum Fingerprinting and Partial Construction of Hash Chains. In: Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018 - Volume 2: SECURE, pp. 463–470 (2018)
15. Knuth, D.E.: *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1*, 1st edn. Addison-Wesley Professional (2011)
16. Kudinov, M.A., Kiktenko, E.O., Fedorov, A.K.: Security analysis of the W-OTS<sup>+</sup> signature scheme: Updating security bounds (2020). URL <https://arxiv.org/abs/2002.07419v1>
17. Leurent, G., Peyrin, T.: SHA-1 is a Shambles - First Chosen-Prefix Collision on SHA-1 and Application to the PGP Web of Trust. *Cryptology ePrint Archive*, Report 2020/014 (2020). URL <https://eprint.iacr.org/2020/014>
18. Merkle, R.C.: A Certified Digital Signature. In: G. Brassard (ed.) *Advances in Cryptology – CRYPTO '89, Lecture Notes in Computer Science*, vol. 435, pp. 218–238 (1989)
19. Perin, L.P., Zamboni, G., Martins, D.M.B., Custódio, R., Martina, J.E.: Tuning the Winternitz Hash-Based Digital Signature Scheme. In: 2018 IEEE Symposium on Computers and Communications (ISCC), pp. 537–542 (2018)
20. Rankl, W., Effing, W.: *Smart Card Handbook*, 4th edn. Wiley (2010)
21. Roh, D., Jung, S., Kwon, D.: Winternitz Signature Scheme Using Nonadjacent Forms. *Security and Communication Networks* **2018** (2018)
22. Steinwand, R., Villányi, V.I.: A one-time signature using run-length encoding. *Information Processing Letters* **108**(4), 179–185 (2008)

## A Proofs of several combinatorial arguments, algorithms and parameter table

### A.1 Proof of Theorem 1

**Theorem 1 ([4, Item 1d])** Consider the set of  $t$ -tuples with elements bounded by  $n$  whose sum is  $s$ . The cardinality of this set is given by

$$|\tau_{(t,n,s)}| = \sum_{i=0}^k (-1)^i \binom{t}{i} \binom{s - (n+1)i + t - 1}{t-1},$$

where  $k = \min\left(t, \left\lfloor \frac{s}{n+1} \right\rfloor\right)$ .

*Proof* Let  $s$  be any non-negative integer. The number of ways that  $t$  non-negative integers, smaller or equal to  $n$ , can be arranged and sum exactly to  $s$ , denoted by  $|\tau_{(t,n,s)}|$ , is given by the coefficient of the term  $x^s$  from the polynomial

$$g(x) = (1 + x^1 + \dots + x^n)^t.$$

We can represent the same polynomial in terms of the sum of the inner geometric series as

$$g(x) = \left(\frac{1 - x^{n+1}}{1 - x}\right)^t = (1 - x^{n+1})^t \frac{1}{(1 - x)^t},$$

and from this we expand the binomials such that

$$g(x) = \left(\sum_{i=0}^t (-1)^i \binom{t}{i} x^{(n+1)i}\right) \left(\sum_{l=0}^{\infty} \binom{l+t-1}{t-1} x^l\right).$$

However, we are only interested in the coefficient of  $x^s$ , which can be expressed as follows:

$$|\tau_{(t,n,s)}| = \sum_{i=0}^t (-1)^i \binom{t}{i} \binom{s - (n+1)i + t - 1}{t-1}.$$

Finally, the maximum value of  $i$  must satisfy  $s - (n+1)i + t - 1 \geq t - 1 \geq 0$  and  $t \geq i$ . Then, since  $t$  is a positive integer, we must have that  $s - (n+1)i \geq 0$  only when  $i \leq \frac{s}{n+1}$  and  $n \geq 0$ . Thus,  $k = \min\left(t, \left\lfloor \frac{s}{n+1} \right\rfloor\right)$  and

$$|\tau_{(t,n,s)}| = \sum_{i=0}^k (-1)^i \binom{t}{i} \binom{s - (n+1)i + t - 1}{t-1}. \quad \square$$

## A.2 Proof of Theorem 2

We note that  $|\tau_{(t,1,s)}| = \binom{s}{t}$ , and the case where  $|\tau_{(t,n,s)}|$  with  $n \geq 1$  is a type of generalization of the binomial coefficients that has been well studied [4, 9, 10]. For any fixed  $n \geq 1$ , together with Proposition 1, the following theorem implies the unimodality property of  $|\tau_{(t,n,s)}|$ , for a fixed  $t$ . We do not claim the result is new, but in the absence of locating a proof of the exact result, we give an inductive proof.

**Theorem 2** Let  $n \geq 1$ ,  $t \geq 1$  and  $0 \leq s \leq \left\lceil \frac{tn}{2} \right\rceil$ . For any  $0 \leq j \leq s$  we have

$$|\tau_{(t,n,j)}| \leq |\tau_{(t,n,s)}|.$$

*Proof* We prove the proposition by induction on  $t$ . To simplify calculations, we extend the definition of  $\tau$  such that  $|\tau_{(t,n,s)}| = 0$  for  $s < 0$  and  $s > tn$ . We prove the result for  $j \in \mathbb{Z}$ ,  $j \leq s$ . To greatly improve readability, we set  $r = \frac{n}{2}$ .

The base case is  $t = 1$ . For any  $0 \leq j < s \leq \lceil tr \rceil$ , we have  $|\tau_{(1,n,s)}| = |\tau_{(1,n,j)}| = 1$ , and whenever  $j < 0$  we have  $|\tau_{(1,n,j)}| = 0$ , so the inequality holds for  $t = 1$ .

For the inductive step, we let  $t \geq 2$ ,  $s \leq \lceil tr \rceil$  and assume the main statement holds for  $t' = t - 1$ , and any  $j'$  and  $s'$  such that  $0 \leq s' \leq \lceil (t-1)r \rceil$  and  $j' < s'$ . It is enough to show that

$$|\tau_{(t,n,s-1)}| \leq |\tau_{(t,n,s)}|, \quad (4)$$

since the repeated application of this equation extends the result for any  $j < s$ . Recalling that  $|\tau_{(t,n,s)}| = \sum_{j=0}^n |\tau_{(t-1,n,s-j)}|$  by Eq. (1), and letting  $\ell = \lceil tr \rceil - s$ , we rewrite this equation in terms of  $\ell$ , which gives

$$|\tau_{(t,n,s)}| = |\tau_{(t,n,\lceil tr \rceil - \ell)}| = \sum_{i=-\lceil r \rceil}^{\lceil r \rceil} |\tau_{(t-1,n,\lfloor (t-1)r \rfloor - \ell + i)}|.$$

The above equality can be verified by carefully analyzing the four cases where  $n$  and  $t$  assume even or odd values. Now, we write the above equation with  $s - 1$  in place of  $s$

$$|\tau_{(t,n,s-1)}| = |\tau_{(t,n,\lceil tr \rceil - \ell - 1)}| = \sum_{i=-\lceil r \rceil}^{\lceil r \rceil} |\tau_{(t-1,n,\lfloor (t-1)r \rfloor - \ell + i - 1)}|.$$

Thus, we get that  $|\tau_{(t,n,s-1)}| \leq |\tau_{(t,n,s)}|$  if and only if

$$|\tau_{(t-1,n,\lfloor (t-1)r \rfloor - \ell - \lceil r \rceil - 1)}| \leq |\tau_{(t-1,n,\lfloor (t-1)r \rfloor - \ell + \lceil r \rceil)}|. \quad (5)$$

If  $\ell \geq \lceil r \rceil$  then  $\lfloor (t-1)r \rfloor - \ell + \lceil r \rceil \leq \lceil (t-1)r \rceil$  and by the induction hypothesis Eq. (5) holds. If  $\ell < \lceil r \rceil$  we apply Proposition 1 to find that

$$|\tau_{(t-1,n,\lfloor (t-1)r \rfloor - \ell + \lceil r \rceil)}| = |\tau_{(t-1,n,\lceil (t-1)r \rceil + \ell - \lceil r \rceil)}|,$$

and thus  $\lfloor (t-1)r \rfloor - \ell - \lceil r \rceil - 1 \leq \lceil (t-1)r \rceil + \ell - \lceil r \rceil \leq \lceil (t-1)r \rceil$ . From this, we apply the induction hypothesis so that

$$\begin{aligned} |\tau_{(t-1,n,\lfloor (t-1)r \rfloor - \ell - \lceil r \rceil - 1)}| &\leq |\tau_{(t-1,n,\lceil (t-1)r \rceil + \ell + \lceil r \rceil)}| \\ &= |\tau_{(t-1,n,\lceil (t-1)r \rceil + \ell - \lceil r \rceil)}|. \end{aligned}$$

This completes the proof of Eq. (5), that implies Eq. (4) and concludes the proof.  $\square$

## A.3 Proof of Proposition 3

**Proposition 3** For any  $0 \leq l \leq n$ ,

$$\begin{aligned} |\tau_{(t,n,s)}^{(l)}| &= \sum_{i=0}^k (-1)^i \binom{t-1}{i} \times \\ &\quad \left[ \binom{s - (n+1)i + t - 1}{t-1} - \binom{s - (n+1)i + t - 2 - l}{t-1} \right], \end{aligned}$$

where  $k = \min\left(t, \left\lfloor \frac{s}{n+1} \right\rfloor\right)$ .

*Proof* We recall that the summation in Theorem 1 has an upper bound  $k$  due to the first binomial coefficient evaluating to zero for values  $i > k$ . Then, from Proposition 2 and by letting

$$k = \max_{j \in \{0, \dots, n\}} \left( \min\left(t, \left\lfloor \frac{s-j}{n+1} \right\rfloor\right) \right) = \min\left(t, \left\lfloor \frac{s}{n+1} \right\rfloor\right),$$

we can express the cardinality  $|\tau_{(t,n,s)}^{(l)}|$  as

$$\begin{aligned} &\sum_{j=0}^l \sum_{i=0}^k (-1)^i \binom{t-1}{i} \binom{s - (n+1)i + t - 2 - j}{t-2} \\ &= \sum_{i=0}^k (-1)^i \binom{t-1}{i} \left[ \sum_{j=0}^l \binom{s - (n+1)i + t - 2 - j}{t-2} \right]. \end{aligned}$$

For any  $0 \leq j \leq l$ , we let  $\alpha = s - (n+1)i$  and  $\beta = t - 2$ . Then, the inner summation can be simplified as:

$$\begin{aligned} \sum_{j=0}^l \binom{\alpha + \beta - j}{\beta} &= \sum_{j=0}^{\alpha + \beta} \binom{j}{\beta} - \sum_{j=0}^{\alpha + \beta - l - 1} \binom{j}{\beta} \\ &= \binom{\alpha + \beta + 1}{\beta + 1} - \binom{\alpha + \beta - l}{\beta + 1}. \end{aligned}$$

Substituting the values of  $\alpha$  and  $\beta$  yields the proof.  $\square$

#### A.4 Proof of Proposition 4

**Proposition 4** *Let  $m$  and  $t$  be integers such that  $t = o(m)$ , and let  $n$  be such that*

$$|\tau_{(t,n-1,\lfloor \frac{(n-1)t}{2} \rfloor})| < 2^m \leq |\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor})|.$$

*Let  $\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}^m$  be the image of the encoding  $\mathbf{E}(\{0,1\}^m) \subseteq \tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}$ . Consider the sequence of independent and uniformly distributed random variables  $X = X_m$  from  $\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}$ , and the sequence of independent and uniformly distributed random variables  $Y = Y_m$  from  $\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}^m$ . Then,  $X$  and  $Y$  are statistically indistinguishable.*

*Proof* We need to prove that for every  $D > 0$  there exists  $m_0$ , such that:

$$\frac{1}{2} \sum_{\alpha \in \tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}} |\Pr[X_m = \alpha] - \Pr[Y_m = \alpha]| < \frac{1}{m^D}, \text{ for all } m \geq m_0.$$

The left-hand side of the inequality is equal to

$$\begin{aligned} & \frac{1}{2} \left( \sum_{\alpha \in \tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}^m} \left| \frac{1}{2^m} - \frac{1}{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}|} \right| \right. \\ & \left. + \sum_{\alpha \in (\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)} \setminus \tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}^m)} \left| \frac{1}{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}|} - 0 \right| \right) \\ & = 1 - \frac{2^m}{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}|} = \frac{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}| - 2^m}{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}|}. \end{aligned} \quad (6)$$

From [9, Eq. (5)], we know that  $|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}| \sim \frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{(n+1)^t}{\sqrt{(n+1)^2 - 1}}$ . So, for  $m, n$  large enough, our hypothesis translates to

$$\frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{n^t}{\sqrt{n^2 - 1}} < 2^m \leq \frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{(n+1)^t}{\sqrt{(n+1)^2 - 1}},$$

Therefore,

$$\begin{aligned} 2^m & \leq |\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}| \\ & \leq \frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{(n+1)^t}{\sqrt{(n+1)^2 - 1}} \\ & = \frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{n^t}{\sqrt{n^2 - 1}} \frac{(n+1)^t}{n^t} \frac{\sqrt{n^2 - 1}}{\sqrt{(n+1)^2 - 1}} \\ & \leq 2^m \left( 1 + \frac{1}{n} \right)^t \frac{\sqrt{n^2 - 1}}{\sqrt{n^2 + 2n}}. \end{aligned}$$

Using  $|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}| \leq 2^m \left( 1 + \frac{1}{n} \right)^t \frac{\sqrt{n^2 - 1}}{\sqrt{n^2 + 2n}}$  to bound the numerator and using  $2^m \leq |\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}|$  to bound the denominator of the left-hand side of (6), we obtain

$$\begin{aligned} \frac{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}| - 2^m}{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}|} & \leq \frac{2^m \left( \left( 1 + \frac{1}{n} \right)^t \frac{\sqrt{n^2 - 1}}{\sqrt{n^2 + 2n}} - 1 \right)}{2^m} \\ & < \left( 1 + \frac{1}{n} \right)^t - 1. \end{aligned}$$

Expanding the binomial in the right-hand side, we obtain

$$\frac{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}| - 2^m}{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}|} < 1 + \frac{t}{n} + \frac{\binom{t}{2}}{n^2} + \dots + \frac{1}{n^t} - 1.$$

Since  $2^m \leq \frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{(n+1)^t}{\sqrt{(n+1)^2 - 1}}$ , we have  $2^{m/t} \leq \frac{1}{(\frac{\pi t}{6})^{1/(2t)}} \frac{(n+1)}{(n^2 + 2n)^{1/(2t)}}$ .

Thus, we get  $\frac{\binom{t}{i}}{n^i} \leq \frac{\binom{t}{i}}{2^{m/t}} \frac{n+1}{n^i} \frac{1}{(\frac{\pi t}{6})^{1/(2t)} (n^2 + 2n)^{1/(2t)}}$ . Since  $t = o(m)$ ,

then  $\frac{\binom{t}{i}}{2^{m/t}}$  approaches zero faster than any polynomial in  $m$ . In particular, for a given positive constant  $D$ , there exists  $m_0^{(i)}$  such that  $\frac{\binom{t}{i}}{n^i} \leq \frac{\binom{t}{i}}{2^{m/t}} \frac{n+1}{n^i} \frac{1}{(\frac{\pi t}{6})^{1/(2t)} (n^2 + 2n)^{1/(2t)}} < \frac{1}{m^D}$ , for all  $m \geq m_0^{(i)}$ . Now, considering  $m_0 = \max\{m_0^{(1)}, m_0^{(2)}, \dots, m_0^{(t)}\}$ , we get  $\frac{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}| - 2^m}{|\tau_{(t,n,\lfloor \frac{nt}{2} \rfloor)}|} < 1 + \frac{t}{n} + \frac{\binom{t}{2}}{n^2} + \dots + \frac{1}{n^t} - 1 \leq \frac{1}{m^D}$ , for all  $m \geq m_0$ .  $\square$

#### A.5 Proof of Proposition 5

**Proposition 5** *Let  $t, n \geq 2, 0 \leq \eta \leq n$  and  $s \leq \frac{tn}{2}$ . Let  $(b_{t-1}, \dots, b_0)$  be chosen uniformly at random from  $\tau_{(t,n,s)}$ . Then, for any  $0 \leq j \leq \eta$  and for any  $0 \leq i \leq t-1$ ,*

$$\frac{|\tau_{(t-1,n,s-\eta)}|}{|\tau_{(t,n,s)}|} \leq \Pr[b_i = j] \leq \frac{|\tau_{(t-1,n,s)}|}{|\tau_{(t,n,s)}|}.$$

*Proof* We observe that  $b_i$  follows the same distribution as  $b_0$  for any  $i$ . This holds since for any  $t$ -tuple in  $\tau_{(t,n,s)}$ , each possible index permutation of the tuple is in  $\tau_{(t,n,s)}$ . Furthermore, we know that  $|\tau_{(t-1,n,s-j)}|$  is the cardinality of the set of all  $t$ -tuples in  $\tau_{(t,n,s)}$  where  $b_0 = j$ . Hence,

$$\Pr[b_i = j] = \Pr[b_0 = j] = \frac{|\tau_{(t-1,n,s-j)}|}{|\tau_{(t,n,s)}|}.$$

Applying Theorem 2 (see Appendix A.2), we get the bounds

$$\frac{|\tau_{(t-1,n,s-\eta)}|}{|\tau_{(t,n,s)}|} \leq \Pr[b_i = j] \leq \frac{|\tau_{(t-1,n,s)}|}{|\tau_{(t,n,s)}|}. \quad \square$$

#### A.6 Technical results on the behavior of $\tau_{(t,n,s)}$

**Theorem 4** *Let  $t, n, s \geq 1$ .*

$$\frac{|\tau_{(t-1,n,s)}|}{|\tau_{(t,n,s)}|} \leq \frac{|\tau_{(t-1,s,s)}|}{|\tau_{(t,s,s)}|} = \frac{t-1}{s+t-1}.$$

*Proof* The case  $t = 1$  is trivial, as both numerators equal zero, so we continue the proof for  $t \geq 2$ . The equality is easy to derive since  $|\tau_{(t,s,s)}| = \binom{s+t-1}{t-1}$  and  $|\tau_{(t-1,s,s)}| = \binom{s+t-2}{t-2}$ , and one only needs to use the formulas for the binomials. So, all we need to prove is:

$$\frac{|\tau_{(t-1,n,s)}|}{|\tau_{(t,n,s)}|} \leq \frac{t-1}{s+t-1}. \quad (7)$$

Consider a bipartite graph  $G = (V = V_1 \cup V_2, E)$ , where the vertex set  $V$  is partitioned in two sets:  $V_1$  labeled by the elements of  $\tau_{(t-1,n,s)}$  and  $V_2$  labeled by the elements of  $\tau_{(t,n,s)}$ . Add an edge between  $A = (a_{t-2}, a_{t-3}, \dots, a_0) \in \tau_{(t-1,n,s)}$  and  $B = (b_{t-1}, b_{t-2}, \dots, b_0) \in \tau_{(t,n,s)}$  if and only if tuple  $B$  is a “refinement” of tuple  $A$ , that is, if there exists  $0 \leq i \leq t-2$ , such that  $a_j = b_j$ , for  $0 \leq j \leq i-1$ ,  $a_j = b_{j+1}$ , for  $i+1 \leq j \leq t-2$  and  $a_i = b_{i+1} + b_i$ .

We count the number of edges  $|E|$  in two ways. First, we note that the degree of each vertex in  $V_1$  is  $s+1$ , as in total there are  $s+1$  ways to refine a tuple whose components add up to  $s$ , by splitting one of its components in every possible way. A way to see this is to consider the stars and bars representation of a tuple. A  $t$ -tuple with elements adding up to  $s$  with components having values in  $\{0, \dots, n\}$  can be represented by a string with  $s$  stars and  $t-1$  bars, where the number of stars between bars (also counting before the first bar and after the last bar) represents the number in each of the  $t$  cells. To refine this tuple amounts to adding an extra bar to the string representation in any of the  $s+1$  possible locations, i.e. each position adjacent to any of the

stars. For example, tuple  $A = (1, 0, 2)$  is represented as  $*|**$  and can be refined as  $|*||**$ ,  $*|||**$ ,  $*||**|$  or  $*||**|$  corresponding to  $(0, 1, 0, 2)$ ,  $(1, 0, 0, 2)$ ,  $(1, 0, 1, 1)$ , or  $(1, 0, 2, 0)$ , respectively. This gives the equation  $|E| = (s+1)|V_1| = (s+1)|\tau_{(t-1,n,s)}|$ .

Second, we give an upper bound on  $|E|$  by taking into account the degree of vertices in  $V_2$ . The neighbors of each vertex  $B = (b_{t-1}, b_{t-2}, \dots, b_0) \in V_2 = \tau_{(t,n,s)}$  are obtained by “merges” of adjacent positions, that is:  $(b_{t-1}+b_{t-2}, b_{t-3}, \dots, b_0)$ ,  $(b_{t-1}, b_{t-2}+b_{t-3}, \dots, b_0)$ , and so on until  $(b_{t-1}, \dots, b_2, b_1+b_0)$ , as long as the new value  $b_{i+1} + b_i \leq n$ . Thus, the maximum degree of a vertex in  $V_2$  is  $t-1$ , which happens if every one of the  $t-1$  “merges” yields a different tuple and each tuple gives a valid merge with  $b_i + b_{i+1} \leq n$ . Hence,  $(t-1)|V_2|$  is clearly an overestimation for  $|E|$ . From this quantity, we subtract the tuples that were overcounted due to merges in two different positions that yielded the same neighbor, which happens once each time a zero is present in one of the positions  $b_{t-2}, b_{t-3}, \dots, b_1$ . For example, for the tuple  $B = (1, \mathbf{0}, \mathbf{0}, 1, \mathbf{0}, 3, 0)$ , instead of 6 distinct neighbors we have only 3:  $(1, 0, 1, 0, 3, 0)$ ,  $(1, 0, 0, 1, 3, 0)$ ,  $(1, 0, 0, 1, 0, 3)$ , since some merges give the same tuple. Indeed, there is one repeat tuple for each bold zero, so the total number of neighbors is  $6-3=3$ . We count the total number of zeros appearing in positions  $b_{t-2}, b_{t-3}, \dots, b_1$  of each tuple  $B \in V_2 = \tau_{(t,n,s)}$ , which is precisely  $(t-2)\tau_{(t-1,n,s)}$ ; this is so, because whenever one fixes a zero in one of the mentioned  $t-2$  positions, the rest of the  $t$ -tuple can be completed by inserting the components of one of the tuples of  $\tau_{(t-1,n,s)}$ . Thus,  $|E| \leq (t-1)|V_2| - (t-2)|\tau_{(t-1,n,s)}| = (t-1)|\tau_{(t,n,s)}| - (t-2)|\tau_{(t-1,n,s)}|$ . Combining with the previous equation on  $|E|$ , we obtain:  $(s+1)|\tau_{(t-1,n,s)}| = |E| \leq (t-1)|\tau_{(t,n,s)}| - (t-2)|\tau_{(t-1,n,s)}|$ . This gives  $(s+t-1)|\tau_{(t-1,n,s)}| \leq (t-1)|\tau_{(t,n,s)}|$ , which implies Eq. (7).  $\square$

**Theorem 5** Let  $t, n \geq 2$ ,  $2n-1 \leq s \leq \frac{tn}{2}$  and tuples  $(b_{t-1}, \dots, b_0)$ ,  $(b'_{t-1}, \dots, b'_0)$  be taken arbitrarily from  $\tau_{(t,n,s)}$ . Let  $(a_{t-1}, \dots, a_0)$  be taken uniformly at random from  $\tau_{(t,n,s)}$ , and  $\gamma$  be taken uniformly at random from  $\{0, \dots, t-1\}$ . Then,

1.  $\Pr[b_\gamma \leq a_\gamma] \geq \frac{1}{s+t-1}$ .
2.  $\Pr[b_\gamma \leq a_\gamma < b'_\gamma \mid b \neq b'] \geq \frac{1}{n+1} \left(\frac{1}{t}\right)^t$ .

*Proof* 1. Since we take  $\gamma$  at random, we get

$$\Pr[b_\gamma \leq a_\gamma] = \frac{1}{t} \sum_{i=0}^{t-1} \Pr[b_i \leq a_i].$$

Since the probability of  $a_i = j$  decreases as  $j$  increases, a lower bound can be obtained by using the average value of all  $b_i$ , namely  $\frac{s}{t}$ , in place of each  $b_i$ . This can be proven with a simple exchange argument, where moving a unit of a larger  $b_i$  to a smaller one does not increase the total probability. Thus,

$$\begin{aligned} \Pr[b_\gamma \leq a_\gamma] &\geq \frac{1}{t} \sum_{i=0}^{t-1} \Pr\left[\frac{s}{t} \leq a_i\right] \\ &= \Pr\left[\frac{s}{t} \leq a_0\right] = 1 - \Pr\left[\frac{s}{t} > a_0\right]. \end{aligned}$$

Now,  $\Pr\left[\frac{s}{t} > a_0\right] = \sum_{j=0}^{\lceil \frac{s}{t} \rceil - 1} \Pr[a_0 = j]$ , and by Proposition 5,

$$\Pr\left[\frac{s}{t} > a_0\right] \leq \left(\frac{s}{t} + 1\right) \left(\frac{t-1}{s+t-1}\right) \leq \frac{s+t}{s+t-1}.$$

Thus,

$$\begin{aligned} \Pr[b_\gamma \leq a_\gamma] &\geq 1 - \Pr\left[\frac{s}{t} > a_0\right] \\ &\geq 1 - \frac{s+t}{s+t-1} = \frac{1}{s+t-1}. \end{aligned}$$

2. Since  $b \neq b'$  and both tuples' components add up to  $s$ , there must be at least one position  $i \in \{0, \dots, t-1\}$  where  $b_i < b'_i$ , and  $\Pr[\gamma = i] = \frac{1}{t}$ . In addition, the lowest probability for  $\Pr[b_i \leq a_i < b'_i]$  occurs when the values of  $b_i < b'_i$  are as large as possible, that is,  $b_i = n-1$  and  $b'_i = n$ . This is true because  $\tau_{(t,n,s)}$  has less tuples with  $a_i = j$ , the larger the  $j$  is. Thus, lower bounding with the smallest probability scenario, we get

$$\Pr[b_\gamma \leq a_\gamma < b'_\gamma \mid b \neq b'] \geq \frac{1}{t} \Pr[a_\gamma = n-1].$$

Now, the rest of the proof consists of finding a lower bound for  $\Pr[a_\gamma = n-1]$ . We know

$$\begin{aligned} \Pr[a_\gamma = n-1] &= \frac{|\tau_{(t-1,n,s-(n-1))}|}{|\tau_{(t,n,s)}|} \\ &= \frac{|\tau_{(t-1,n,s-(n-1))}|}{|\tau_{(t,n,s-(n-1))}|} \frac{|\tau_{(t,n,s-(n-1))}|}{|\tau_{(t,n,s)}|}. \end{aligned}$$

By Eq. (1),  $|\tau_{(t,n,s-(n-1))}| = \sum_{j=0}^n |\tau_{(t-1,n,s-(n-1)-j)}|$ , and by Theorem 2,  $|\tau_{(t-1,n,s-(n-1)-j)}| \leq |\tau_{(t-1,n,s-(n-1))}|$ , for all  $0 \leq j \leq n$ . Combining these facts and that

$$\sum_{j=0}^n |\tau_{(t-1,n,s-(n-1)-j)}| \leq (n+1) |\tau_{(t-1,n,s-(n-1))}|,$$

we get

$$\frac{|\tau_{(t-1,n,s-(n-1))}|}{|\tau_{(t,n,s-(n-1))}|} \geq \frac{1}{n+1}.$$

Thus,

$$\Pr[a_\gamma = n-1] \geq \frac{1}{n+1} \frac{|\tau_{(t,n,s-(n-1))}|}{|\tau_{(t,n,s)}|}.$$

Since  $s \geq 2n-1$ , then  $s-(n-1) \geq n$ . So,  $|\tau_{(t,n,s-(n-1))}| \geq |\tau_{(t,n,n)}| = \binom{n+t-1}{t-1}$ , and since  $\tau_{(t,n,s)} \subseteq \tau_{(t,s,s)}$  we also have that  $|\tau_{(t,n,s)}| \leq |\tau_{(t,s,s)}| = \binom{s+t-1}{t-1}$ . Substituting both relations in the main equation above and using  $n \geq 2$ , we get

$$\begin{aligned} \Pr[a_\gamma = n-1] &\geq \frac{1}{n+1} \frac{|\tau_{(t,n,n)}|}{|\tau_{(t,s,s)}|} = \frac{1}{n+1} \frac{\binom{n+t-1}{t-1}}{\binom{s+t-1}{t-1}} \\ &\geq \frac{1}{n+1} \left(\frac{n}{\frac{nt}{2}+t}\right)^{t-1} = \frac{1}{n+1} \left(\frac{1}{\frac{t}{2}+\frac{t}{n}}\right)^{t-1} \\ &\geq \frac{1}{n+1} \left(\frac{1}{t}\right)^{t-1}. \end{aligned}$$

Finally,

$$\Pr[b_\gamma \leq a_\gamma < b'_\gamma \mid b \neq b'] \geq \frac{1}{t} \Pr[a_\gamma = n-1] \geq \frac{1}{n+1} \left(\frac{1}{t}\right)^t. \quad \square$$

**Theorem 6** Let  $t \geq 2$  be given.

1. For  $m$  and  $s \geq 2$  such that  $|\tau_{(t,s-1,s-1)}| < 2^m \leq |\tau_{(t,s,s)}|$ , we have, as  $m \rightarrow \infty$ ,

$$2^m \leq |\tau_{(t,s,s)}| = 2^m (1 + o(1)).$$

2. For  $m$  and  $n \geq 2$  such that

$$\frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{n^t}{\sqrt{n^2-1}} < 2^m \leq \frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{(n+1)^t}{\sqrt{(n+1)^2-1}},$$

we have, as  $m \rightarrow \infty$ ,

$$2^m \leq \frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{(n+1)^t}{\sqrt{(n+1)^2-1}} = 2^m (1 + o(1)).$$

*Proof* 1. Consider the difference  $|\tau_{(t,s,s)}| - |\tau_{(t,s-1,s-1)}|$ . Expanding into their binomial forms, we have

$$\begin{aligned} & \binom{s+t-1}{t-1} - \binom{s-1+t-1}{t-1} \\ &= \frac{(s+t-1)!}{(t-1)!s!} - \frac{(s+t-2)!}{(t-1)!(s-1)!} = \binom{s+t-2}{t-1} \frac{t-1}{s}. \end{aligned}$$

Hence, since  $t$  is bounded, as  $m \rightarrow \infty$  thus  $s \rightarrow \infty$ , and we have

$$\begin{aligned} |\tau_{(t,s,s)}| &= \binom{s+t-1}{t-1} = \binom{s+t-2}{t-1} \left(1 + \frac{t-1}{s}\right) \\ &< 2^m \left(1 + \frac{t-1}{s}\right) = 2^m(1 + o(1)). \end{aligned}$$

2. Since the rightmost term is larger than the leftmost term, we can write

$$\begin{aligned} 2^m &\leq \frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{(n+1)^t}{\sqrt{(n+1)^2-1}} \\ &= \frac{1}{\sqrt{\frac{\pi t}{6}}} \frac{n^t}{\sqrt{n^2-1}} \frac{(n+1)^t}{n^t} \frac{\sqrt{n^2-1}}{\sqrt{(n+1)^2-1}} \\ &\leq 2^m \left(1 + \frac{1}{n}\right)^t \frac{\sqrt{n^2-1}}{\sqrt{n^2+n}} = 2^m(1 + o(1)), \end{aligned}$$

with the last equality holding since  $t$  is fixed and  $n \rightarrow \infty$  as  $m \rightarrow \infty$ .  $\square$

## A.7 Algorithms $\mathcal{M}'^{\mathcal{A}}$ and $\mathcal{M}''^{\mathcal{A}}$

---

### Algorithm 6 $\mathcal{M}'^{\mathcal{A}}$

---

**Input:**  $\lambda$ , sample  $(\beta, u, r, k)$

**Output:** 0 or 1

```

1:  $(\mathbf{sk}, \mathbf{pk}) \leftarrow \text{GEN}(1^\lambda)$ .  $\triangleright r$  and  $k$  are taken from sample instead
2:  $\alpha \leftarrow_{\$} \{0, \dots, t-1\}$ 
3: Obtain  $\mathbf{pk}'$  where  $\mathbf{pk}'_i = c_k^n(\mathbf{sk}_i, r')$  for  $0 \leq i \leq t-1, i \neq \alpha$  and
    $\mathbf{pk}'_\alpha = c_k^\beta(y_c, r'_{n-\beta+1,n})$ .
4: Receive  $M$  from  $\mathcal{A}$  in the query phase and  $(b_{t-1}, \dots, b_0) \leftarrow \mathbf{E}(M)$ 
5: if  $b_\alpha > \beta$  then
6:   return 0
7:  $\sigma = \text{SIGN}(M, \mathbf{sk}, r')$ 
8: Replace  $\sigma_\alpha = c_k^{\beta-b_\alpha}(y_c, r'_{n-\beta+1,n})$ 
9: Reply to query with  $\sigma$  and obtain  $(M', \sigma')$  from  $\mathcal{A}$ 
10: if  $(M', \sigma')$  is a valid signature then
11:   Let  $(b'_{t-1}, \dots, b'_0) \leftarrow \mathbf{E}(M')$ 
12:   if  $b'_\alpha \leq \beta$  then
13:     return 0
14:   else
15:     return 1
16: return 0
```

---



---

### Algorithm 7 $\mathcal{M}''^{\mathcal{A}}$

---

**Input:**  $\lambda$ , sample  $(u, k)$

**Output:** 0 or 1

```

1: Generate  $r \leftarrow_{\$} \{0, 1\}^{\lambda \times n}$ 
2: Call  $\mathcal{M}'^{\mathcal{A}}$  with inputs  $\lambda$  and  $(\beta^*, c_k^{n-\beta^*-(i^*+1)}(u, r_{i^*+2,n}), r, k)$ 
3: return Output of  $\mathcal{M}'^{\mathcal{A}}$ 
```

---

A.8 List of MINGEN parameters for  $t$ **Table 7** Parameters  $(t, n, s)$  using MINGEN with  $30 \leq t \leq 70$ .

| $t$ | $n$ | $s$  | $C(\text{GEN})$ | $C(\text{SIG})$ | $C(\text{VER})$ |
|-----|-----|------|-----------------|-----------------|-----------------|
| 70  | 13  | 375  | 910             | 535             | 375             |
| 69  | 13  | 427  | 897             | 470             | 427             |
| 68  | 14  | 402  | 952             | 550             | 402             |
| 67  | 15  | 400  | 1005            | 605             | 400             |
| 66  | 15  | 442  | 990             | 548             | 442             |
| 65  | 16  | 439  | 1040            | 601             | 439             |
| 64  | 17  | 445  | 1088            | 643             | 445             |
| 63  | 17  | 531  | 1071            | 540             | 531             |
| 62  | 18  | 519  | 1116            | 597             | 519             |
| 61  | 19  | 532  | 1159            | 627             | 532             |
| 60  | 20  | 556  | 1200            | 644             | 556             |
| 59  | 21  | 596  | 1239            | 643             | 596             |
| 58  | 23  | 562  | 1334            | 772             | 562             |
| 57  | 24  | 603  | 1368            | 765             | 603             |
| 56  | 25  | 681  | 1400            | 719             | 681             |
| 55  | 27  | 666  | 1485            | 819             | 666             |
| 54  | 29  | 687  | 1566            | 879             | 687             |
| 53  | 31  | 722  | 1643            | 921             | 722             |
| 52  | 33  | 772  | 1716            | 944             | 772             |
| 51  | 35  | 862  | 1785            | 923             | 862             |
| 50  | 38  | 876  | 1900            | 1024            | 876             |
| 49  | 41  | 935  | 2009            | 1074            | 935             |
| 48  | 45  | 958  | 2160            | 1202            | 958             |
| 47  | 49  | 1018 | 2303            | 1285            | 1018            |
| 46  | 53  | 1117 | 2438            | 1321            | 1117            |
| 45  | 58  | 1205 | 2610            | 1405            | 1205            |
| 44  | 64  | 1286 | 2816            | 1530            | 1286            |
| 43  | 70  | 1474 | 3010            | 1536            | 1474            |
| 42  | 78  | 1556 | 3276            | 1720            | 1556            |
| 41  | 87  | 1707 | 3567            | 1860            | 1707            |
| 40  | 98  | 1835 | 3920            | 2085            | 1835            |
| 39  | 110 | 2127 | 4290            | 2163            | 2127            |
| 38  | 126 | 2221 | 4788            | 2567            | 2221            |
| 37  | 144 | 2490 | 5328            | 2838            | 2490            |
| 36  | 165 | 2955 | 5940            | 2985            | 2955            |
| 35  | 192 | 3283 | 6720            | 3437            | 3283            |
| 34  | 226 | 3643 | 7684            | 4041            | 3643            |
| 33  | 267 | 4285 | 8811            | 4526            | 4285            |
| 32  | 320 | 4945 | 10240           | 5295            | 4945            |
| 31  | 388 | 5790 | 12028           | 6238            | 5790            |
| 30  | 476 | 6953 | 14280           | 7327            | 6953            |