# Tuning the Winternitz hash-based digital signature scheme

Lucas Pandolfo Perin, Gustavo Zambonin, Douglas Marcelino Beppler Martins, Ricardo Felipe Custódio, and Jean Everson Martina

Universidade Federal de Santa Catarina, Florianópolis, SC, 88040-900, Brazil
lucas.perin@posgrad.ufsc.br

**Abstract.** It is known that, for a given set of parameters, the overall complexity of signature generation and its verification is constant for the Winternitz one-time signature scheme (WOTS), through chained iterations of a function $f$. We introduce a new variant for this scheme that allows the user to specify which stage of the digital signature task will be simplified, i.e. increase the overall signature generation time in favor of faster verification or vice-versa. We decrease the number of iterations of $f$ by up to half, with regards to the verification procedure, for commonly used values of the Winternitz parameter $w$.

**Keywords:** post-quantum cryptography, hash-based signatures, digital signatures, One-time signature, Winternitz

## 1 Introduction

Traditional digital signature schemes are based on number theory problems, such as the difficulty of factoring large composite numbers and determining the discrete logarithm. Such problems are the basis for, for example, the RSA and ECDSA classical algorithms. To generate a digital signature of a message, a large number of operations on integers are required, which limits the use of these traditional schemes in devices with low memory and processing power. This is the case for most IoT devices and sensors. Moreover, Shor [17] has shown that, with the advent of the quantum computer, these traditional digital signature schemes will become fragile, and may no longer guarantee the authenticity and integrity properties of digital signatures.

A possible alternative to this is the use of hash-based digital signatures. It has been show in the literature that one-way functions, the theoretical foundation of a hash function, are the only construction needed to build a secure digital signature scheme [16, 10]. Additionally, hash-based schemes are thought to be resilient against quantum attacks [6, 2, 11]. These reasons motivate several post-quantum digital signature schemes proposed based on the use of hash functions.

The idea of using only hashes to sign digital messages is not new. Lamport [12] proposed, in 1979, a very simple, secure and efficient scheme to generate and verify digital signatures. However, such a scheme presents some drawbacks. The

first is related to the large size of the cryptographic keys. The second, each public and private key pair can only be used to sign a single message.

These drawbacks of the scheme due to Lamport have been the subject of research and several improvements have been proposed since its initial design. Perhaps the most promising method is known as Winternitz one time signature (WOTS) [14]. This scheme is a generalization of the Lamport scheme. Unlike Lamport, WOTS allows more than one bit to be signed simultaneously. As such, the size of the cryptographic keys are decreased. In fact, the lower cost of the key has been replaced by a higher cost of processing. More recently, new hash-based signature schemes have been proposed based on, or using WOTS as part of the scheme [4, 9, 3], some of which are being worked towards standardization by the U.S. National Institute of Standards and Technology (NIST) [7] and the Internet Engineering Task Force (IETF) [8, 13].

In almost all hash-based digital signature schemes, the private key is a randomly generated number, and the public key is the application of a hash function, recursively, a large number of times. To sign a document, a fingerprint function is used, which defines how many times the hash function should be applied to the private key. The result of this application is the signature of the message. In order to verify the signature of the message, it is still necessary to apply the hash until it coincides or not with the public key of the signer. In the case of the WOTS scheme, this process can be tailored to a faster signature generation or verification.

We propose two methods to improve the WOTS scheme signature generation and/or verification steps. Our first approach is based on [18], where we append a nonce to a document to be signed and hash it. We repeat the process and search for hash outputs that can optimize signature generation or verification steps. In fact, by doing this, we do not change the underlying WOTS algorithm. We propose a fixed amount of operations before the signature is generated, so that the chosen step can be computed much faster. It is actually a trade-off choice, where improving one step results in more computations for the other. However, we reiterate that there is always a fixed cost before the signature generation. Our second method changes the checksum computation slightly by padding unused bits with 1's, in the place of zeros. This last proposal can be used to improve signature verification time, but not the other way around. Both proposals can be used independently, but for improving signature verification they can also work together for better results.

### 1.1 Related Works

In practice, signatures are not produced as often as they are verified. Moreover, signatures are usually generated with plenty of computational resource available, whereas the verification is performed by various devices with all kinds of computational power. For this reason, one common practice is to choose parameters that makes it easier for a third party to verify signatures, such as the 65537 exponent of the RSA signature scheme. Papers that focus on modifying the WOTS verification step in order to achieve a speedup, such as [5, 15, 18], consider only

small values of $w$. Besides, situations where a large amount of signed messages need to be generated quickly are not the focus of these works. We aim to cover these limitations with our proposals.

### 1.2 Outline Remarks

*Notation.* We use the following symbols throughout the rest of this work. The length of a word $\omega$ is given by $|\omega|$. For a set of integers $S$, $min(S)$ and $max(S)$ are, respectively, the minimum and maximum elements of $S$. The concatenation of two words $\omega_1, \omega_2$ is given by the symbol $||$. Repeated mapping of a function $f$ on some input $x$ is recursively defined as $f^0(x) = x$ and $f^i(x) = f(f^{i-1}(x))$ for $i \in \mathbb{N}^*$. The symbol $\xleftarrow{\$}$ is read as "chosen randomly from".

*Organization.* The remaining of the paper is organized as follows: Section 2 briefly states the classical WOTS scheme; in Section 3, we review some definitions of the scheme, present bounds for the variables used during the signature generation and verification and proposes the addition of a checksum pad; Section 4 contains the main proposal of this work; finally, Section 5 contains results obtained by implementing our proposals, followed by our final considerations.

## 2 Classical Winternitz

In its original proposal [14], the Winternitz one-time signature scheme (WOTS) works as follows. Take $m$ as a security parameter, usually the output size in bits of a cryptographic hash function, and choose a parameter $w \in \mathbb{N}, w > 1$. In addition, consider a non-compressing one-way function $f : \{0,1\}^m \longrightarrow \{0,1\}^m$ and a cryptographic hash function $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^m$. Then, compute

$$t_1 = \left\lceil \frac{m}{w} \right\rceil, \quad t_2 = \left\lfloor \frac{\lfloor \log_2 t_1 \rfloor + 1 + w}{w} \right\rfloor, \quad t = t_1 + t_2.$$

These parameters are selected so that $w$ is the number of bits to be signed simultaneously. They are also directly related to the key size and signature performance, as we will see in the sequence.

*Key Generation.* Let $X = (x_{t-1}, \ldots, x_0) \xleftarrow{\$} \{0,1\}^m$ be the private key, informally a $t$-tuple of pseudo-random integers. The public key $Y = (y_{t-1}, \ldots, y_0$ can be derived from $X$ by applying $f$ to each of the $x_i$ elements $2^w - 1$ times. Hence, $Y = (f^{2^w-1}(x_{t-1}), \ldots, f^{2^w-1}(x_0))$.

*Signature Generation.* Take a message $M$ and compute its digest $d = \mathcal{H}(M)$. For convenience, we choose $m$ or $w$ such that $w \mid m$. However, one may pad the digest by appending zeros to satisfy this requirement. $d$ is then split into a $t_1$-tuple of base-$w$ words $\mathcal{B}_1 = (b_{t-1}, \ldots, b_{t-t_1})$. Additionally, we compute the checksum using the integer representation of the elements in $\mathcal{B}_1$:

$$c = \sum_{i=t-t_1}^{t-1} 2^w - 1 - b_i.$$

Again, $c$ may be padded until $w \mid |c|$. Finally, it is split into a $t_2$-tuple of base-$w$ words $\mathcal{B}_2 = (b_{t_2-1}, \ldots, b_0)$. With $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ computed, we obtain the signature

$$\alpha = (f^{b_{t-1}}(x_{t-1}), \ldots, f^{b_0}(x_0)).$$

*Signature Verification.* To ascertain the correctness of the signature $\alpha$, each of its blocks $\alpha_i$ need to be verified separately by computing the remaining applications of $f$. The signature verifies correctly if

$$Y = (f^{2^w-1-b_{t-1}}(\alpha_{t-1}), \ldots, f^{2^w-1-b_0}(\alpha_0)).$$

In Figure 1 we illustrate the signature generation and its verification. It is visible that computing a signature block is a step necessarily after the key generation. Hence, the key and signature generation may only be computed by whomever has the secret signature key $X$.

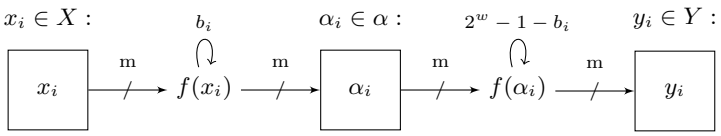


**Fig. 1.** WOTS signature and verification steps for any index $i \in \{0, \ldots, t-1\}$.

## 3 Checksum Padding

As seen in Section 1, in many practical applications, it is desirable that the cost of signature verification be as small as possible. This is because, normally, verification of the signature of a document is done countless times after being signed. In this section, we propose padding the unused bits, reserved for $c$, with 1. The rationale is as follows.

Recall that $c = \sum_{i=t-t_1}^{t-1}(2^w - 1 - b_i)$. Define $c_{max}$ as the greatest possible value of $c$. This happens when $\forall i \in \{t - t_1, \ldots, t-1\}, b_i = 0$. Similarly, define $c_{min}$ as the smallest possible value of $c$. This happens when $\forall i \in \{t - t_1, \ldots, t - 1\}, b_i = 2^w - 1$. Hence, $c_{max} = \sum_{i=t-t_1}^{t-1}(2^w - 1 - 0) = t_1(2^w - 1)$ and $c_{min} = \sum_{i=t-t_1}^{t-1}(2^w - 1 - 2^w - 1) = 0$. Additionally, the number of bits needed to represent all possible values of $c$ is given by $n_c = \lceil \log_2 c_{max} \rceil = \lceil \log_2 t_1(2^w - 1) \rceil$ and the number of blocks to accommodate $c$ is given by $t_2$, defined in Section 2.

In general, the number of bits reserved for $c$, that is, $t_2 w$ bits, is greater than the number of bits actually required for their representation. This difference occurs because an integer number of blocks of size $w$ is used to accommodate $c$. The number of unused bits is defined as $n_u = t_2 w - n_c$.

Therefore, we redefine $c$ to $c^p = c + (2^{n_u} - 1)2^{n_c} = c + 2^{t_2 w} - 2^{n_c}$. We call the modified scheme WOTS-B, with the key generation and signature verification steps left unmodified, and the aforementioned addition performed after $c$ is calculated, inside the signature generation step.

## 4 Tuning $\mathcal{B}_1$

In the following section, we propose a method to speed up signature generation or verification steps without any modification to the WOTS scheme. We note that there is always a fixed cost during the signature generation. We believe that this cost can be compensated for large $w$.

Let $M$ be a message, $R \in \mathbb{N}^*$ and $\lambda$ a $R$-tuple of nonces. We compute all $d_r$ such that $d_r = H(M \mid\mid \lambda_r)$ with $0 \leq r \leq R$. Recall that $d_r$ may be split into a $t_1$-tuple, now defined by $\mathcal{B}_{r,1} = (b_{r,t-1}, \ldots, b_{r,t-t_1})$, and let the set of summations for the integer representations of the elements for these tuples be defined by $S = \{\sum_{b \in \mathcal{B}_{r,1}} : 0 \leq r \leq R\}$.

Finally, we choose $\lambda_r$ from $min(S)$ to obtain a faster signature generation or from $max(S)$ for a faster signature verification. Then we proceed with the classical WOTS signature generation by letting $\mathcal{B}_1 = d_r$.

We call the modified scheme WOTS-R, with the signature verification step left unaltered. The signature generation step is slightly modified with the addition of the computation of $S$. This proposal is heavily inspired in [18], and therefore we observe that $\lambda$ could be replaced by the trivial set $\{0, \ldots, R\}$.

### 4.1 Finding a threshold for $R$

In this proposal, $R$ is a statistical parameter related to the number of hashes needed to find good summations $S$. Intuitively, larger $R$ should obtain better results. However, large $R$ translates to a higher cost for signature generation. We show that there is a suitable threshold for $R$, depending on how much optimization is needed. Consider $\mu$ as the function that calculates the arithmetic mean of a set of integers.

If we assume that $0 \geq b_i \geq 2^w - 1$ follows a uniform distribution, then by repeatedly calculating $\mu(\mathcal{B}_{r,1})$, we expect the average $\mu' = \mu(\mu(B_{r,1})) = 2^{w-1}$ for $0 \leq r \leq R$. Furthermore, the distribution of averages $\mu(\mathcal{B}_{r,1})$ should follow a normal distribution. We experiment with $w = 16$, $m = 256$ and $f =$ BLAKE2s, by computing $\mu(\mathcal{B}_{r,1})$ with $0 \leq r \leq 10000$. Indeed we verify that the distribution of the averages follows a normal distribution, see Figure 2. Hence, using the standard normal table (Z-table) with standard deviation $\sigma$, we have

$$P(\mu(\mathcal{B}_{r,1}) > \mu' + \sigma) = 0.1573,$$

$$P(\mu(\mathcal{B}_{r,1}) > \mu' + 2\sigma) = 0.0214,$$

$$P(\mu(\mathcal{B}_{r,1}) > \mu' + 3\sigma) = 0.0013.$$

In Figure 3, we plot the chance of finding $\mu(\mathcal{B}_{r,1})$ inside these three intervals, with respect to $R$. We use the binomial distribution and distinguish three thresholds: {25, 200, 3500}.

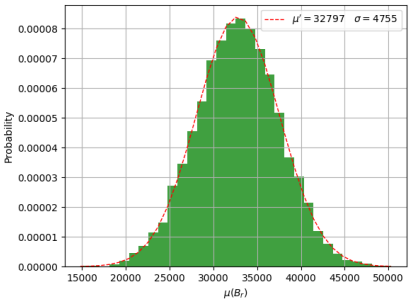


**Fig. 2.** Normalized histogram of $\mu(S)$.

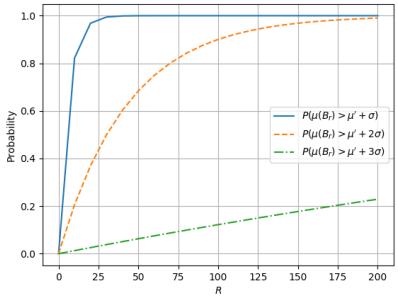


**Fig. 3.** Thresholds for $R$.

### 4.2 Security Considerations

This proposal does not change the entropy of the hash function. By computing distinct hashes independently, we obtain $\mathcal{B}$ tailored for our goal. By allowing the addition of an open value to be concatenated with the original message for the purpose of tuning the signature/verification steps, one would argue that this would lead to the successful execution of the birthday attack from Definition 1.

**Definition 1.** ***The birthday attack.*** *The birthday attack consists of picking points $x_1, \ldots, x_q$ from the domain $D$ and computing $y_i = h(x_i)$ for $i = 1, \ldots, q$. The attack will be successful if there exists a pair $(i, j)$ such that $(x_i, x_j)$ are a collision for the hash function $\mathcal{H}$. The number of trials is identified by $q$.*

As a demonstration we will recur to the birthday attack and its impact on the collision resistance of hash functions [1].

The probability that the birthday attack succeeds on finding a collision for a hash function $\mathcal{H} : D \longrightarrow R$ in $q$ trials can be written as $C_h(q)$. The relation between $q$ and $r$, where $r = |R|$ is the size of the range of $h$ and $q \leq O(\sqrt{r})$ is given by:

$$C_h(q) \approx \binom{q}{2} \cdot \frac{1}{r}.$$

This implies that a collision is expected in $r^{1/2}$ trials approximately. It also predicts that collisions in hash functions with $m$ bits output would take about $2^{m/2}$ trials. This estimate is the basis for the choice of output length $m$, which is typically made just large enough to make $2^{m/2}$ trials unfeasible.

If we assume $\mathcal{H} : \{0,1\}^* \longrightarrow \{0,1\}^r$ being a hash function that behaves as a random oracle the probability for the birthday attack to succeed is given by $C_h(q)$. Thus allowing the tuning of $n$ bits of the input message $M$, where $n \ll \frac{m}{2}$ would reduce the difficulty of the birthday attack with negligible probability as demonstrated in [1].

## 5 Experiments

As a proof-of-concept for the proposals described above, we compare the number of iterations of a function $f$ throughout the entire execution of the digital signature schemes. Again, we denote the four variants tested as WOTS for the classical scheme described on Section 2, WOTS-B for the variant described on Section 3, WOTS-R is the scheme described on Section 4 and WOTS-BR merges the characteristics of the latter two. Considering the discussion on Subsection 4.1, sufficient values of $R$ were chosen according to the usual values of the Winternitz parameter $w$.

In Table 1 we give the average number of iterations of $f$ needed to verify a signature. To better understand the effect of each proposal individually, we also compute the average of $\mathcal{B}_1$ and $\mathcal{B}_2$ separatly. We observe a reduction of close to $2^w$ hashes with WOTS-B, and ranging from 15% considering $w = 4, R = 25$, up to 45% considering $w = 16, R = 3500$, with WOTS-R. Using WOTS-BR may speed up the signature generation or verification steps up to a factor of half in a best-case scenario.

## 6 Final Remarks

In this work, we propose two methods for improving WOTS signature verification time, by spending extra computations during the signature generation or vice-versa. Our first proposal is to pad the extra – unused – bits of the checksum blocks. Our experiments demonstrate that for $w = 16$, this method can be used to increase the number of iterations of $f$, during the signature generation step, by 10% in average. Our second proposal is a method to find nonces that, when appended to a message, produce distinct and suitable hash outputs. We analyze the blocks of size $w$ of these hash outputs and choose one in which the sum of the blocks (represented as integers) is the highest or lowest. Picking the highest means that the WOTS signature generation will be slower. However, the signature verification is much faster. Furthermore, we emphasize that both proposals can be used together for better results.

### 6.1 Future Works

Finally, we propose a few thoughts for future works. We consider using $min(S)$ and $max(S)$ to find nonces to speed up the scheme. However, when considering parallel computation, the minimum or maximum may not be the best choice. We believe that by choosing $\lambda_r$ such that $b_{r,i}$ have similar magnitude and $\mu(\mathcal{B}) > \mu'$, then we can take advantage of parallel computation for faster computations. In addition, we consider finding fast implementations for computing $S$, using SIMD instructions so that this fixed cost can be mitigated and more viable for speeding up signature generations with small $w$.

**Table 1.** $2^{10}$ executions of the verification step for the aforementioned schemes, with $|\lambda_r| = 16$, $f$ defined as BLAKE2s with $m = 256$ and base-16 messages of $2^{11}$ length generated through `/dev/urandom`.

| $w$ | R | Algorithm | $\mu(\mathcal{B}_1)$ | $\mu(\mathcal{B}_2)$ | total |
|---|---|---|---|---|---|
| 4 | - | Wots | 479.04 | 25.73 | 504.77 |
| | | Wots-b | 479.12 | 12.11 | 491.22 |
| | 25 | Wots-r | 404.78 | 27.42 | 432.20 |
| | | Wots-br | 405.87 | 13.76 | 419.64 |
| | 200 | Wots-r | 389.86 | 28.39 | 418.24 |
| | | Wots-br | 388.39 | 14.43 | 402.82 |
| | 3500 | Wots-r | 360.59 | 30.23 | 390.82 |
| | | Wots-br | 360.74 | 16.34 | 377.08 |
| 8 | - | Wots | 4067.86 | 368.91 | 4436.77 |
| | | Wots-b | 4067.72 | 136.52 | 4204.24 |
| | 25 | Wots-r | 3222.71 | 371.67 | 3594.38 |
| | | Wots-br | 3230.16 | 128.45 | 3358.61 |
| | 200 | Wots-r | 3049.73 | 375.48 | 3425.21 |
| | | Wots-br | 3053.72 | 127.01 | 3180.73 |
| | 3500 | Wots-r | 2734.27 | 373.24 | 3107.52 |
| | | Wots-br | 2752.24 | 136.66 | 2888.90 |
| 16 | - | Wots | 521580.75 | 98220.51 | 619801.26 |
| | | Wots-b | 521691.68 | 32713.40 | 554405.07 |
| | 25 | Wots-r | 369860.18 | 98219.56 | 468079.74 |
| | | Wots-br | 370980.98 | 32337.98 | 403318.95 |
| | 200 | Wots-r | 338532.85 | 98409.78 | 436942.62 |
| | | Wots-br | 337185.57 | 32438.73 | 369624.30 |
| | 3500 | Wots-r | 287021.07 | 98772.29 | 385793.36 |
| | | Wots-br | 284147.68 | 33942.88 | 318090.56 |

## References


1. Bellare, M., Kohno, T.: Hash function balance and its impact on birthday attacks. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 401–418. Springer (2004)
2. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post Quantum Cryptography. Springer Publishing Company, Incorporated, 1st edn. (2008)
3. Bernstein, D.J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schneider, M., Schwabe, P., Wilcox-O'Hearn, Z.: Sphincs: practical stateless hash-based signatures. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 368–397. Springer (2015)
4. Buchmann, J., Dahmen, E., Hülsing, A.: XMSS - A practical forward secure signature scheme based on minimal security assumptions. Post-Quantum Cryptography pp. 117–129 (2011)
5. Cruz, J.P., Yatani, Y., Kaji, Y.: Constant-sum fingerprinting for winternitz one-time signature. In: Information Theory and Its Applications (ISITA), 2016 International Symposium on. pp. 703–707. IEEE (2016)
6. Dods, C., Smart, N.P., Stam, M.: Hash based digital signature schemes. Lecture notes in computer science 3796, 96–115 (2005)
7. Endignoux, G.: Design and implementation of a post-quantum hash-based cryptographic signature scheme. Ph.D. thesis, Master's thesis, EPFL (2017)
8. Huelsing, A., Butin, D., Gazdag, S.L., Rijneveld, J., Mohaisen, A.: XMSS: Extended Hash-Based Signatures. Internet-Draft draft-irtf-cfrg-xmss-hash-based-signatures-10, Internet Engineering Task Force (Jul 2017), https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-xmss-hash-based-signatures-10, work in Progress
9. Hülsing, A.: W-OTS$^+$ Shorter signatures for hash-based signature schemes. Africacrypt 7918, 173–188 (2013)
10. Katz, J., Koo, C.Y.: On constructing universal one-way hash functions from arbitrary one-way functions. IACR Cryptology ePrint Archive 2005, 328 (2005)
11. Lafrance, P.: Digital Signature Schemes Based on Hash Functions. Master's thesis, University of Waterloo (2017)
12. Lamport, L.: Constructing digital signatures from a one-way function. Tech. rep., Technical Report CSL-98, SRI International Palo Alto (1979)
13. McGrew, D.D.A., Curcio, M., Fluhrer, S.: Hash-Based Signatures. Internet-Draft draft-mcgrew-hash-sigs-08, Internet Engineering Task Force (Oct 2017), https://datatracker.ietf.org/doc/html/draft-mcgrew-hash-sigs-08, work in Progress
14. Merkle, R.C.: A certified digital signature. In: Conference on the Theory and Application of Cryptology. pp. 218–238. Springer (1989)
15. Pereira, G.C., Puodzius, C., Barreto, P.S.: Shorter hash-based signatures. Journal of Systems and Software 116, 95–100 (2016)
16. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing. pp. 387–394. ACM (1990)
17. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM review 41(2), 303–332 (1999)
18. Steinwandt, R., Villányi, V.I.: A one-time signature using run-length encoding. Information Processing Letters 108(4), 179–185 (2008)