2018 IEEE Symposium on Computers and Communications (ISCC)

# Tuning the Winternitz hash-based digital signature scheme

Lucas Pandolfo Perin, Gustavo Zambonin, Douglas Marcelino Beppler Martins,
Ricardo Custódio, Jean Everson Martina
Departamento de Informática e Estatística
Universidade Federal de Santa Catarina
Florianópolis, SC, 88040-900, Brasil
E-mail: lucas.perin@posgrad.ufsc.br

***Abstract*—It is known that, for a given set of parameters, the overall complexity for generating and verifying a signature is constant and independent of the document being signed, for the Winternitz one-time signature scheme (WOTS). These costs are due to the number of chained iterations of a function $f$. However, the cost for signature generation alone is slightly different from signature verification, and these depend directly on the document. We introduce a new variant for WOTS, which allows the adjustment of these costs, 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 $w$. Additionally, our experiments show that these proposals have substantial impact on Merkle-based signature schemes, such as XMSS.**

***Index Terms*—cryptography, digital signatures, public key**

## I. Introduction

Traditional digital signature schemes, widely used in situations requiring secure and tamper-free communication (e.g. software distribution and SSL tunneling), are derived from cryptographic public key algorithms based on the computational hardness assumption of problems in number theory. Some examples are the factorization of large composite integers, closely related to the RSA cryptosystem, and the discrete logarithm, associated with the DSA and ECDSA algorithms. However, with the advent of quantum computers, these problems can be solved more efficiently with the aid of Shor's algorithm [1]. Ergo, such digital signature schemes will become fragile and may no longer guarantee the authenticity, integrity and non-repudiation properties required for digital signatures. The alternative is to use post-quantum cryptography — algorithms that are thought to be resistant against such techniques.

There are various methods for quantum-resilient digital signatures being proposed in the literature [2]. Some of the most popular ones make use of code-based cryptography, cryptographic hash functions, lattice structures and multivariate polynomials over finite fields. However, all these algorithms still need to be improved. According to [3], code-based cryptographic systems feature impractical key sizes, more security analysis on lattice-based signatures and multivariate cryptography are required, and hash-based signatures force the signer to preserve a state of which keys were used, featuring large signature sizes if this behavior is undesired. Moreover, for the latter system, we believe that the cost difference between signature generation and verification is not exploited to tailor specific needs, such as in the case when signatures are repeatedly verified.

Notwithstanding these limitations, the robust flexibility and the abundant availability of cryptographic hash functions in all kinds of computational systems make hash-based signature schemes potential candidates to replace traditional digital signature schemes. For example, the inclusion of the SHA algorithm to the x86 SSE instruction set allows a possible improvement on the performance of these schemes with fine-tuned implementations and carefully chosen parameters. Additionally, it has been shown in the literature that one-way functions (the theoretical foundation of a hash function) are the only constructions needed to build a secure digital signature scheme [4], [5]. These reasons are strong indicators that hash-based signature schemes are presumably safe, and extremely fast in practice.

It is also important to note that organizations such as the U.S. National Institute of Standards and Technology (NIST) and the Internet Engineering Task Force (IETF) have shown great interest in the research and standardization of these algorithms. The SPHINCS+ proposal [6], a variant of [7] that reduces the signature size and execution time, was submitted to NIST as a candidate for this process. Besides, there are other examples of submissions [8] and Internet Drafts [9], [10] for hash-based schemes. These circumstances have motivated the proposition of several post-quantum digital signature schemes based on the use of hash functions [11], [2], [12].

The idea of using only hashes to sign digital messages is not new. Lamport [13] proposed, in 1979, a very simple and secure scheme to generate and verify digital signatures. It considers a pair of pseudorandom words as the private key and their hashes as the public key. The signer signs a single bit of information by choosing to distribute one string from the private key, on whether the bit is 0 or 1, as the signature. The verifier then computes the hash of this string, comparing it with part of the public key. By extending this idea to sign messages of arbitrary sizes, some drawbacks emerge, such as large sizes for the cryptographic key pair and signature.

These drawbacks have been the subject of research and sev-

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

eral improvements have been proposed since its initial design. Perhaps the most promising method is the Winternitz one time signature scheme (WOTS) due to Merkle [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 sizes 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, various hash-based signature schemes have been proposed based on, or using WOTS as part of the scheme [15], [16], [17], [18], [7], [6].

Most hash-based one-time signature schemes feature the following construction: the private key is composed of words generated by a pseudorandom function, and the public key is the result of recursively applying a hash function to the each element of the private key. To sign a document, a fingerprint function is used, which defines how many times the hash function should be applied to the private key to produce the signature. The result of this evaluation is the signature of the message. In order to perform the verification step, it is necessary to compute more hashes for the signature, until it coincides with the public key of the signer. Also, by applying a data structure known as Merkle tree, one can sign many documents with multiple one-time key pairs, and a single public key.

In the case of WOTS, the total number of hashes to sign a document and verify a signature depends solely on the size of the parameter $w$. However, these quantities are not distributed evenly, since they are derived from the result of the application of the fingerprint function to a message. Hence, these can be tailored to a faster signature generation or verification by carefully choosing a message derived from the original.

We propose two methods to improve the signature generation or verification steps for WOTS. Our first method changes the checksum computation slightly, by padding unused bits with ones in the place of zeros. This last proposal can be used to improve signature verification run times, while padding with zeros is already optimal for signature generation. Our main approach is based on [19], where we append a cryptographic 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. 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. This is actually a trade-off choice, where improving one step results in more computations for the other. However, there is always an additional fixed cost to the signature generation. Both proposals can be applied independently or together for better results.

### A. Related Works

Signatures are generated only once and frequently on a targeted device, which can be properly configured to accommodate their required resources. However, the resources available for the signature verification are unlikely to be predictable in advance, since this step can be performed by any other kind of devices. For this reason, in practice, it is common to choose parameters that make it easier for a third party to verify signatures. An example is the use of the public exponent $65{,}537$ of the RSA signature scheme. This choice makes signature verification faster than RSA signature generation. This is no different with hash-based signature schemes. Several papers proposed in the literature aim to improve the performance of the verification step for WOTS, such as [20], [21], [19]. While these works modify the nature of the Winternitz scheme, our main proposal consists of a preprocessing of the message, keeping the scheme intact. Another advantage of our scheme is the possibility of using larger values for the parameter $w$ (such as $w = 16$) substantially reducing the size of the WOTS signature and, even so, with small verification time at the expense of a slight increase in signature generation time.

### B. 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". Finally, let $\Delta^{(x_1,x_2)}$ be "a $x_2$-tuple of words with length $x_1$ from alphabet $\Delta$".

*Organization.* The remaining of the paper is organized as follows: Section II briefly states the classical WOTS scheme; in Section III, we review some definitions of the scheme, present bounds for the variables used during the signature generation and verification steps and propose the modification of the checksum padding; Section IV contains the main proposal of this work; finally, Section V contains results obtained by experimenting with WOTS and XMSS, followed by our final considerations.

## II. CLASSICAL WINTERNITZ

In its original proposal [14], 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

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

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 following.

*Key Generation.* Let $X = (x_{t-1}, \ldots, x_0) \xleftarrow{\$} \{0,1\}^{(m,t)}$ be the private key. The public key $Y = (y_{t-1}, \ldots, y_0)$ is 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 with zeroes 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 assert the correctness of the signature $\alpha$, each of its blocks $\alpha_i$ needs 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 whoever has the secret signature key $X$.

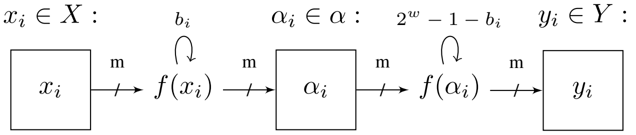

Fig. 1. WOTS signature and verification steps for a index $i \in \{0, \ldots, t-1\}$.

## III. Checksum Padding

In this section, we propose to pad unused bits, reserved for $c$, with 1. Recall that $c = \sum_{i=t-t_1}^{t-1}(2^w - 1 - b_i)$. Define $c_{max}$ and $c_{min}$ as the greatest and smallest possible values of $c$. These situations happen when, $\forall b \in \mathcal{B}_1$, $b = 0$ or $b = 2^w - 1$, respectively. 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 II.

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 $n_u = t_2 w - n_c$.

We note that $w$ and $n_u$ grow together, as seen in Table I. It groups several parameters for WOTS, how they affect $n_u$, and presents how these parameter groups benefit from this optimization. Odd values for $w$ show no abnormal behavior and are suppressed for simplicity, although there are some combinations where no padding is needed, such as $w = 3, m = 128$ and $w = 7, m = 512$. Finally, we redefine $c$ to

$$c^p = c + (2^{n_u} - 1)2^{n_c} = c + 2^{t_2 w} - 2^{n_c}.$$

In other words, during the signature generation step, we fill the unused $n_u$ bits with ones. We call the modified scheme WOTS-B.

TABLE I
Unused bits on $\mathcal{B}_2$ for various combinations of $w$ and $m$.

| $w$ | $m$ | $n_c$ | $n_u$ | $t_2 w$ | $w$ | $m$ | $n_c$ | $n_u$ | $t_2 w$ |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 128 | 8 | 2 | 10 | 10 | 128 | 14 | 6 | 20 |
| | 192 | 9 | 1 | | | 192 | 15 | 5 | |
| | 256 | 9 | 1 | | | 256 | 15 | 5 | |
| | 512 | 10 | 2 | 12 | | 512 | 16 | 4 | |
| 4 | 128 | 9 | 3 | 12 | 12 | 128 | 16 | 8 | 24 |
| | 192 | 10 | 2 | | | 192 | 16 | 8 | |
| | 256 | 10 | 2 | | | 256 | 17 | 7 | |
| | 512 | 11 | 1 | | | 512 | 18 | 6 | |
| 6 | 128 | 11 | 1 | 12 | 14 | 128 | 18 | 10 | 28 |
| | 192 | 11 | 1 | | | 192 | 18 | 10 | |
| | 256 | 12 | 0 | | | 256 | 19 | 9 | |
| | 512 | 13 | 5 | 18 | | 512 | 20 | 8 | |
| 8 | 128 | 12 | 4 | 16 | 16 | 128 | 19 | 13 | 32 |
| | 192 | 13 | 3 | | | 192 | 20 | 12 | |
| | 256 | 13 | 3 | | | 256 | 20 | 12 | |
| | 512 | 14 | 2 | | | 512 | 21 | 11 | |

### A. Security Considerations

We recall that, in Section II, the classical WOTS implementation already uses a padding of zeroes in the signature generation algorithm. Our proposal based on flipping the padding bits from zeroes to ones moves this fixed amount of iterations of $f$ from the verification process to the signature generation. These computations have no impact in security, since they have no checksum purpose, but must be evaluated nevertheless.

## IV. Tuning $\mathcal{B}_1$

We propose a method to speed up the WOTS signature generation or verification without any modification to the original scheme. Our idea is to append a cryptographic nonce to the signed message, before generating a signature. We show in Section V that, by repeating this process and searching for a suitable nonce, we can significantly reduce the cost of the signature verification, in exchange of an increased cost of the signature generation, or vice-versa. In the following, we explain the method and give statistical thresholds for the searching process.

Let $M$ be any message, $R \in \mathbb{N}^*$ and $\lambda = (\lambda_0, \ldots, \lambda_{R-1})$ an $R$-tuple of nonces. We compute all $d_r$ such that $d_r = \mathcal{H}(M \mid\mid \lambda_r)$ 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 of the integer representations of the elements in 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 $\mathcal{B}_1 = d_r$ or, simply by appending $\lambda_r$ to the initial message before running the signature algorithm.

We call the method WOTS-R. Furthermore, this proposal is inspired by [19], and therefore we observe that $\lambda$ could be replaced by the trivial set $\{0, \ldots, R\}$.

### A. Finding a threshold for $R$

In this proposal, $R$ is a statistical parameter that represents the sufficient number of hashes needed to find an adequate summation in $S$. Intuitively, larger $R$ should produce better results. However, this choice translates to a higher cost for signature generation. We show that there is a suitable threshold for $R$, depending on the required optimization.

Consider $\mu$ as the function that calculates the arithmetic mean of a set of integers. If we assume that $\forall b \in \mathcal{B}_{r,1}, 0 \leq b \leq 2^w - 1$ follows a uniform distribution, then by repeatedly calculating $\mu(\mathcal{B}_{r,1})$, we expect the average $\mu' = \mu(\mu(\mathcal{B}_{r,1})) = 2^{w-1}$, for $0 \leq r \leq R$. Hence, the distribution of the averages $\mu(\mathcal{B}_{r,1})$ should follow a normal distribution.

We experiment with $w = 16$, $m = 256$ and $\mathcal{H} =$ SHA-256, by computing $\mu(\mathcal{B}_{r,1})$ with $0 \leq r < 2^{16}$, and indeed verify that the distribution of the averages follows a normal distribution. Figure 2 is a graphical representation of this behavior. Then, by using the standard normal table (Z-table) with standard deviation $\sigma$, we have

$$P(\mu(\mathcal{B}_{r,1}) > \mu' + \sigma) = 0.1587,$$
$$P(\mu(\mathcal{B}_{r,1}) > \mu' + 2\sigma) = 0.0228,$$
$$P(\mu(\mathcal{B}_{r,1}) > \mu' + 3\sigma) = 0.0013.$$

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 as suggestions for values of R: $\{25, 200, 3500\}$. Each value yields a probability of $99.9\%$ of finding $\mu(\mathcal{B}_{r,1})$ in the intervals previously mentioned, respectively.

### B. Security Considerations

Our proposal makes no attempt to modify the underlying classical WOTS mechanisms. Hence, we discuss the impact of appending a cryptographic nonce $\lambda_r$ to a message $M$ before hashing it. Recall that we must find a suitable hash $d_r = \mathcal{H}(M \mid\mid \lambda_r)$ such that it produces a maximized or minimized sum of $\mathcal{B}_{r,1}$. In other words, high-order bits of various blocks in $\mathcal{B}_{r,1}$ have high probability of being fixed, making partial hash collision attacks more susceptible.

This behavior may be exploited through differential cryptanalysis on the fixed bits for the Merkle-Damgård construction, recently put in practice to generate the first practical collision for SHA-1 [22]. Such an attack could present a threat to our proposal, thus requiring the use of cryptographic hash functions which are second preimage resilient, such as SHA-256. We leave the remaining security considerations for the scheme to be taken from [18].

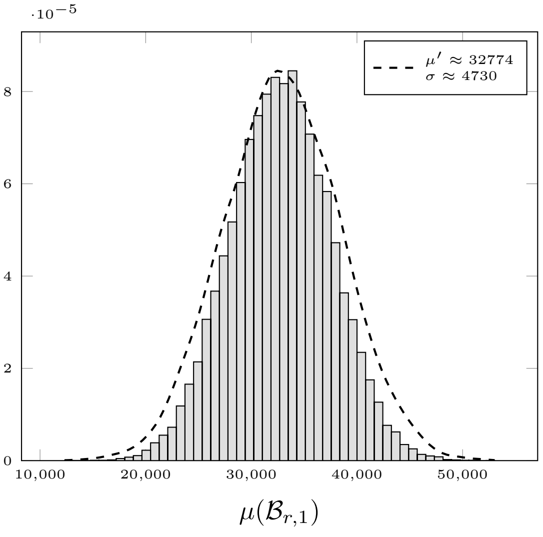

Fig. 2. Normalized histogram of $\mu(\mathcal{B}_{r,1})$, with 50 bins.

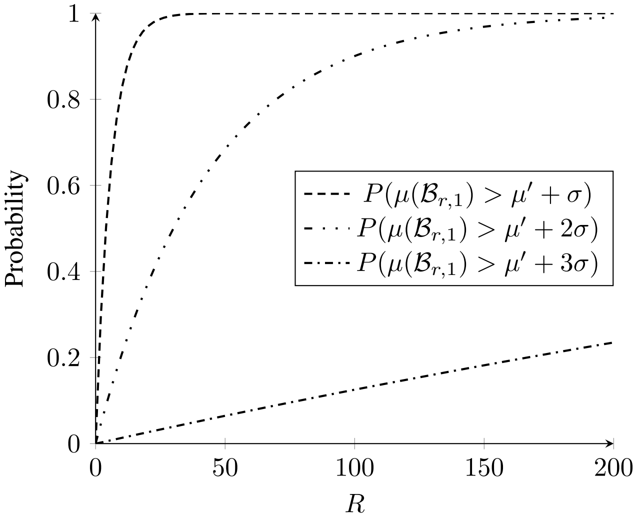

Fig. 3. Thresholds for $R$.

## V. EXPERIMENTS

As a proof-of-concept, 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 II, WOTS-B for the variant described on Section III, WOTS-R is the scheme described on Section IV and WOTS-BR merges the characteristics of the latter two. Considering the discussion on Subsection IV-A, sufficient values of $R$ were chosen according to the usual values of the Winternitz parameter $w$.

In Table II we give the average number of iterations of $f$ needed to verify a signature. We experiment with $2^{14}$ executions of the verification step for the proposed schemes, with $\mathcal{H} =$ SHA-256, $m = 256$ and base-16 messages of $2^{10}$ length generated through `/dev/urandom`. To better understand the effect of each proposal individually, we also

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

compute the average of $\mathcal{B}_1$ and $\mathcal{B}_2$ separately.

Note that $\mu(\mathcal{B}_2)$ is affected negatively by WOTS-R, since maximizing the sum of elements in $\mathcal{B}_1$ has a direct impact on the calculation of the checksum. However, this difference does not heavily impact the overall gains achieved in $\mu(\mathcal{B}_1)$. Furthermore, by using WOTS-BR, this behavior is greatly mitigated and may speed up the signature generation or verification steps up to a factor of half in a best-case scenario.

TABLE II
NUMBER OF ITERATIONS OF $f$ ON THE VERIFICATION STEP FOR THE PROPOSED SCHEMES WHEN $max(S)$ IS CHOSEN.

| $w$ | $R$ | SCHEME | $\mu(\mathcal{B}_1)$ | $\mu(\mathcal{B}_2)$ | $\mu(\mathcal{B})$ |
|---|---|---|---|---|---|
| 4 | - | WOTS | 479.93 | 25.88 | 505.80 |
| | | WOTS-B | | 12.25 | 492.18 |
| | 25 | WOTS-R | 407.81 | 27.49 | 435.29 |
| | | WOTS-BR | | 13.49 | 421.29 |
| | 200 | WOTS-R | 379.08 | 29.23 | 408.31 |
| | | WOTS-BR | | 15.23 | 394.31 |
| | 3500 | WOTS-R | 348.88 | 31.14 | 380.02 |
| | | WOTS-BR | | 17.14 | 366.02 |
| 8 | - | WOTS | 4081.84 | 368.43 | 4450.27 |
| | | WOTS-B | | 136.25 | 4218.08 |
| | 25 | WOTS-R | 3262.39 | 370.56 | 3632.95 |
| | | WOTS-BR | | 130.56 | 3392.95 |
| | 200 | WOTS-R | 2940.63 | 372.13 | 3312.76 |
| | | WOTS-BR | | 132.13 | 3072.76 |
| | 3500 | WOTS-R | 2604.49 | 374.75 | 2979.24 |
| | | WOTS-BR | | 134.75 | 2739.24 |
| 16 | - | WOTS | 525120.63 | 98231.81 | 623352.44 |
| | | WOTS-B | | 32707.82 | 557828.45 |
| | 25 | WOTS-R | 376550.24 | 98225.54 | 474775.78 |
| | | WOTS-BR | | 32697.52 | 409247.76 |
| | 200 | WOTS-R | 319490.02 | 98850.06 | 418340.08 |
| | | WOTS-BR | | 33321.91 | 352811.94 |
| | 3500 | WOTS-R | 262301.92 | 101022.36 | 363324.28 |
| | | WOTS-BR | | 35492.57 | 297794.48 |

In general, we observe a reduction of roughly $2^w$ iterations of $f$ with WOTS-B alone. In the case of WOTS-R, we obtain a reduction of up to 25% for $w = 4$, 33% for $w = 8$ and 42% for $w = 16$. By merging both schemes together, we improve these results to $28\%, 39\%$ and $52\%$, respectively.

The aforementioned reductions translate to an increase of similar magnitude on the signature generation. For example, according to Section II and by letting $w = 4$, then $t = 67$ and the total number of iterations of $f$ for signature generation and verification is equal to $t \times 2^w = 1072$. Our results show that, when $R = 25$ with WOTS-BR, we can decrease the number of iterations of $f$ during the signature verification from approximately 506 to 421, in average.

Avoiding these 85 iterations of $f$ during the signature verification means that we must now calculate these on the signature generation. In other words, this amounts to a $16.8\%$ speedup for the verification step at the cost of a $15\%$ slowdown during the signature generation, without taking the computation of $R$ into account.

We can substantially increase this trade-off by letting $R = 200$ or $R = 3500$, when signature generation time is not constrained. Otherwise, for small values of $w$, the number of hashes used for WOTS-R might not be an attractive choice. Hence, such values can be better used with greater values of $w$, where computing hundreds of hash functions is negligible compared to $t \times 2^w$.

### A. Impact on Merkle signature schemes

Our proposal has significant results for hash-based schemes that make use of Merkle tree structures. We test WOTS-B, WOTS-R and WOTS-BR with the public domain[1] reference implementation of XMSS for the IETF Internet Draft [9].

We patch the reference implementation by modifying the padding process inside the WOTS+ signing algorithm, according to Section III, denoting this modification as XMSS-B. Furthermore, by choosing $R$ with the method described in Section IV, we achieve up to 32% of speedup when benchmarking the verification step for XMSS. We call this variant XMSS-R. Additionally, when these optimizations are used together, we call the resulting scheme XMSS-BR.

Table III shows the average run time of $2^{14}$ signatures for each scheme, including the computation of $R$ for XMSS-R. We use the recommended value of $w = 4$, and additionally, $w = 8$. For greater values of $w$ (e.g. 16), it is widely known that the XMSS key generation algorithm is too slow and unpractical. Hence, this value is omitted from the results. Furthermore, we experiment with both $max(S)$ and $min(S)$ to demonstrate the impact of our schemes when choosing to optimize signature verification or generation, respectively. In the case of $min(S)$, XMSS-B and XMSS-BR are not considered, since WOTS' default padding already optimizes signature generation.

Our results in Table III show that, for $max(S)$, any value of $R$ improves the signature verification run time, with the associated cost on signature generation. Evidently, this behavior is suppressed with greater values of $w$. However, for $min(S)$, not every value of $R$ may be chosen. In the case of $w = 4$ and $R = 25$, we obtain a speedup of $9.4\%$ for the signature generation process, while when $R = 200$ or $R = 3500$, both processes present a slower run time. The same reasoning applies to $w = 8$, where one should only choose $R = 25$ or $R = 200$.

## VI. FINAL REMARKS

In this paper, we propose methods to adjust WOTS for faster signature verification at the cost of a slower signature generation, or vice-versa. Our first proposal, WOTS-B, consists of flipping the checksum padding bits, thus improving signature verification run time. In our second proposal, WOTS-R, we present a choice to to maximize or minimize the base-$w$ blocks constructed from the message digest. The former is used to speed up signature verification, whereas the latter speeds up signature generation or verification.

We experiment with the classical Winternitz scheme and a state-of-the-art Merkle-based signature scheme, XMSS. We obtain an improvement factor of up to two when $w = 16$, for the WOTS signature verification process, with regards to the number of iterations of $f$. Also, by applying our proposals

[1] `https://github.com/joostrijneveld/xmss-reference/`

978-1-5386-6950-1/18/$31.00 ©2018 IEEE

TABLE III
Signature generation and verification run times (in ms) for the proposed schemes.

| | $w$ | $R$ | Scheme | Sig. time | Ver. time |
|---|---|---|---|---|---|
| $max(S)$ | 4 | - | Xmss | 0.953 | 0.734 |
| | | | Xmss-b | 0.975 | 0.724 |
| | | 25 | Xmss-r | 1.059 | 0.652 |
| | | | Xmss-br | 1.073 | 0.637 |
| | | 200 | Xmss-r | 1.222 | 0.620 |
| | | | Xmss-br | 1.240 | 0.616 |
| | | 3500 | Xmss-r | 3.724 | 0.588 |
| | | | Xmss-br | 3.730 | 0.573 |
| | 8 | - | Xmss | 7.709 | 5.676 |
| | | | Xmss-b | 7.908 | 5.361 |
| | | 25 | Xmss-r | 8.597 | 4.637 |
| | | | Xmss-br | 8.992 | 4.415 |
| | | 200 | Xmss-r | 9.045 | 4.245 |
| | | | Xmss-br | 9.460 | 4.052 |
| | | 3500 | Xmss-r | 11.760 | 3.861 |
| | | | Xmss-br | 12.193 | 3.664 |
| $min(S)$ | 4 | - | Xmss | 0.971 | 0.746 |
| | | 25 | Xmss-r | 0.879 | 0.832 |
| | | 200 | | 1.006 | 0.885 |
| | | 3500 | | 3.393 | 0.898 |
| | 8 | - | Xmss | 7.819 | 5.731 |
| | | 25 | Xmss-r | 6.672 | 6.553 |
| | | 200 | | 6.472 | 6.982 |
| | | 3500 | | 8.488 | 7.435 |

Relevant computer specifications are as follows: 8 GB of DDR3 RAM @ 1333MHz, Intel Core i5-4570 @ 3.2GHz and `gcc` 7.3.0. Base commit for the modifications: 05dac989c40349ad5f4dfee3b563b85131b95332.

to Xmss, we reduce the run time for the verification step by 22% when $w = 4$ and 36% when $w = 8$. Finally, when improving signature generation, for $w = 4$ and $w = 8$, we obtain speedups of 9.4% and 17.2%, respectively.

### A. Future Works

Throughout our work, we consider using $min(S)$ and $max(S)$ to find cryptographic nonces that speed up the Wots scheme. However, this may not be the best approach. We believe that the criteria for choosing $\lambda_r$ can be improved, by searching for $\mathcal{B}_{r,1}$ with elements of similar magnitude, in addition to the original proposal of $\mu(\mathcal{B}_{r,1}) > \mu'$. Parallel implementations can exploit this trait and allow faster computations of $f$.

## References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997. [Online]. Available: https://arxiv.org/abs/quant-ph/9508027v2

[2] D. J. Bernstein, J. Buchmann, and E. Dahmen, *Post Quantum Cryptography*, 1st ed. Springer Publishing Company, Incorporated, 2008.

[3] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, Sep. 2017.

[4] J. Rompel, "One-way functions are necessary and sufficient for secure signatures," in *STOC '90: Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, H. Ortiz, Ed., May 1990, pp. 387–394.

[5] J. Katz and C.-Y. Koo, "On constructing universal one-way hash functions from arbitrary one-way functions," Cryptology ePrint Archive, Report 2005/328, Sep. 2005. [Online]. Available: https://eprint.iacr.org/2005/328

[6] D. J. Bernstein, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hülsing, P. Kampanakis, S. Klbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, and P. Schwabe, "SPHINCS$^+$ – Submission to the NIST post-quantum project," Dec. 2017. [Online]. Available: https://sphincs.org/data/sphincs+-specification.pdf

[7] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M. Schneider, P. Schwabe, and Z. Wilcox-O'Hearn, "SPHINCS: Practical stateless hash-based signatures," in *Advances in Cryptology – EUROCRYPT 2015*, ser. LNCS, E. Oswald and M. Fischlin, Eds., vol. 9056, Apr. 2015, pp. 368–397. [Online]. Available: https://eprint.iacr.org/2014/795

[8] G. Endignoux, "Design and implementation of a post-quantum hash-based cryptographic signature scheme," Master's thesis, École polytechnique fédérale de Lausanne, Jul. 2017. [Online]. Available: https://gendignoux.com/assets/pdf/2017-07-master-thesis-endignoux-report.pdf

[9] A. Hülsing, D. Butin, S.-L. Gazdag, J. Rijneveld, and A. Mohaisen, "XMSS: Extended Hash-Based Signatures," Internet Engineering Task Force, Internet-Draft draft-irtf-cfrg-xmss-hash-based-signatures-12, Jan. 2018, Work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-xmss-hash-based-signatures-12

[10] D. A. McGrew, M. Curcio, and S. Fluhrer, "Hash-Based Signatures," Internet Engineering Task Force, Internet-Draft draft-mcgrew-hash-sigs-08, Oct. 2017, Work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-mcgrew-hash-sigs-08

[11] C. Dods, N. P. Smart, and M. Stam, "Hash based digital signature schemes," in *Proceedings of the 10th IMA International Conference on Cryptography and Coding*, ser. LNCS, N. Smart, Ed., vol. 3796, Dec. 2005, pp. 96–115.

[12] P. Lafrance, "Digital signature schemes based on hash functions," Master's thesis, University of Waterloo, Apr. 2017. [Online]. Available: https://uwspace.uwaterloo.ca/handle/10012/11679

[13] L. Lamport, "Constructing digital signatures from a one-way function," SRI International Palo Alto, Technical Report CSL-98, Oct. 1979. [Online]. Available: https://www.microsoft.com/en-us/research/publication/constructing-digital-signatures-one-way-function/

[14] R. C. Merkle, "A certified digital signature," in *Advances in Cryptology – CRYPTO '89*, ser. LNCS, G. Brassard, Ed., vol. 435, Aug. 1989, pp. 218–238.

[15] J. Buchmann, L. C. C. García, E. Dahmen, M. Döring, and E. Klintsevich, "CMSS: An improved merkle signature scheme," in *Progress in Cryptology – INDOCRYPT 2006*, ser. LNCS, R. Barua and T. Lange, Eds., vol. 4329, Dec. 2006, pp. 349–363. [Online]. Available: https://eprint.iacr.org/2006/320

[16] J. Buchmann, E. Dahmen, E. Klintsevich, K. Okeya, and C. Vuillaume, "Merkle signatures with virtually unlimited signature capacity," in *Applied Cryptography and Network Security*, ser. LNCS, J. Katz and M. Yung, Eds., vol. 4521, Jun. 2007, pp. 31–45.

[17] J. Buchmann, E. Dahmen, and A. Hülsing, "XMSS - a practical forward secure signature scheme based on minimal security assumptions," in *Post-Quantum Cryptography*, ser. LNCS, B.-Y. Yang, Ed., vol. 7071, Nov. 2011, pp. 117–129. [Online]. Available: https://eprint.iacr.org/2011/484

[18] A. Hülsing, "W-OTS$^+$ – shorter signatures for hash-based signature schemes," in *Progress in Cryptology – AFRICACRYPT 2013*, ser. LNCS, A. Youssef, A. Nitaj, and A. E. Hassanien, Eds., vol. 7918, Jun. 2013, pp. 173–188. [Online]. Available: https://eprint.iacr.org/2017/965

[19] R. Steinwandt and V. I. Villányi, "A one-time signature using run-length encoding," *Information Processing Letters*, vol. 108, no. 4, pp. 179–185, Oct. 2008.

[20] J. P. Cruz, Y. Yatani, and Y. Kaji, "Constant-sum fingerprinting for Winternitz one-time signature," in *2016 International Symposium on Information Theory and Its Applications (ISITA)*, Oct. 2016, pp. 703–707.

[21] G. C. C. F. Pereira, C. Puodzius, and P. S. L. M. Barreto, "Shorter hash-based signatures," *Journal of Systems and Software*, vol. 116, pp. 95–100, Jun. 2016.

[22] M. Stevens, E. Bursztein, P. Karpman, A. Albertini, and Y. Markov, "The first collision for full SHA-1," in *Advances in Cryptology – CRYPTO 2017*, ser. LNCS, J. Katz and H. Shacham, Eds., vol. 10401, Jul. 2017, pp. 570–596. [Online]. Available: https://eprint.iacr.org/2017/190