A clipping technique for shorter hash-based signatures

Amos Y. C. L. Zheng[1], **Leonardo T. D. Ferraz**[1], **Marcos A. Simplicio Jr.**[1]

[1]Escola Politécnica da Universidade de São Paulo (EPUSP)
São Paulo – SP – Brazil

azheng@larc.usp.br, lferraz@larc.usp.br, mjunior@larc.usp.br

Abstract. *Hash-based signature schemes are a class of post-quantum algorithms that usually consist of hash-trees built upon One-Time Signature (OTS) solutions. These schemes have small key sizes, efficient processing and are simple to implement, while their security properties rely basically on the pre-image or collision resistance of the their underlying hash function. Despite such advantages, however, they have relatively large signature sizes compared to traditional signature algorithms. One way of tackling this issue is to reduce the sizes of their underlying OTS algorithms. In this article, we describe a probabilistic technique that, with negligible processing overhead, allows such reductions. Namely, up to 12.5% average size reduction can be achieved depending on the signature's parameters.*

1. Introduction

The area of post-quantum cryptography refers to algorithms that can be executed on classic computers and, at the same time, resist known attacks made by quantum computers [Bernstein et al. 2008]. This includes five main families of post-quantum algorithms and protocols: lattices [Goldreich et al. 1997], error correcting codes [McEliece 1978], multivariate quadratic systems [Ding and Schmidt 2005], isogenies on supersingular elliptic curves[Jao and De Feo 2011], and schemes based on symmetric cryptography primitives in general, such as hash functions [Dods et al. 2005].

The goal of researching such cryptosystems is to face the threat posed by quantum computing, an area that is evolving increasingly fast [Anthony 2017]. More precisely, if large-enough quantum computers become available, the security of many classical cryptographic solutions (e.g., RSA [Rivest et al. 1978] and ECC[Koblitz 1987][Miller 1986]) can be bypassed by attackers using Shor [Shor 1999] or Grover [Grover 1996] algorithms. In a smaller scale, symmetric primitives are also affected by the Grover algorithm: for example, the security level of symmetric ciphers drop by half, so brute-force attacks against k-bit keys take $2^{k/2}$ operations instead of 2^k ; similarly, the security of k-bit hash functions against pre-image attacks drop from k to $k/2$, and against collisions the fall is from $k/2$ to $k/3$ [Bernstein and Lange 2017]. Therefore, efficient schemes that can become drop-in replacements for conventional solutions are required. For symmetric primitives, there are already natural alternatives, such as replacing AES-128 with AES-256 [NIST 2001], or SHA-3-256 with SHA-3-512 [NIST 2015]. For asymmetric schemes, however, this is a challenging issue, in particular because the memory, bandwidth and/or processing requirements of asymmetric post-quantum cryptosystems are usually higher than what is observed in currently standardized solutions [Bernstein et al. 2008].

Among the aforementioned families of post-quantum algorithms, a particularly promising area are hash-based signatures [Bernstein et al. 2008]. Algorithms in this category are built using one-way hash functions as underlying cryptographic primitive, so their (quantum and classical) security relies basically on well-known properties of those primitives, such as collision and pre-image resistance [Dods et al. 2005]. That is because these algorithms all share the same idea of generating a list of random bit strings as the private key, then hash those strings one or more times to get the public key. After that, depending on the message to be signed, the corresponding pre-images are published with the signature. In addition, hash-based signatures are also quite efficient: besides leading to small public key sizes, their computational costs depend basically on performance of the underlying hashing functions, which are processing and memory-efficient algorithms.

Despite this interest, hash-based signatures still face some challenges. One of its main limitations is the reasonably large signature sizes when compared to classical digital signature schemes. For example, Winternitz One-Time Signature (W-OTS)[Merkle 1990], a one-time signature scheme proposed by Ralph Merkle in 1979, had signature sizes around 4 KiB, which is 8x the size of ECC-based multi-signature schemes such as ECDSA[ecd]. A more recent one-time signature proposal, W-OTS+ [Hülsing 2013], reduces this size in about half by requiring simply pre-image resistant hash functions rather than collision-resistant ones. On the other hand, many-times hash-based signature proposals built upon one-time schemes, such as XMSS[Buchmann et al. 2011] and SPHINCS[Bernstein et al.], has signature sizes of 10 KiB and 41 KiB respectively, which remain much larger than even not too compact classical schemes, such as RSA, for an equivalent classic security level.

Aiming to address this issue, in this paper we describe a novel probabilistic technique that can be combined with one-time schemes such as W-OTS and W-OTS+ for reducing the signature size under certain conditions. The proposed approach, called "clipping", relies on pseudo-random function (PRF) chaining, so the scheme's private keys are generated in two layers instead of a single one. As a result, two or more hash pre-images that need to be revealed for composing the signature do not need to be explicitly send: since they can be derived from a single seed, it suffices to reveal this seed. Even though this is a probabilistic approach whose efficiency depends on the data to be signed, its overhead in terms of processing cost or memory usage is negligible, so the (potentially small) gains obtained when it is employed come basically for free.

The rest of this paper is organized as follows. Section 2 gives an overview of the state of the art in the area of hash-based signature schemes. Section 3 describes our technique in details. We present the results of our technique in terms of signature size, processing time, storage required and security in Section 4. Finally, Section 5 concludes the discussion and presents our final considerations.

2. Hash-based signature schemes: state-of-the-art

As aforementioned, hash-based signatures rely on hard-to-invert hash functions, leading to quite computationally efficient digital signature schemes [Bernstein et al. 2008]. These functions map data of arbitrary size to a fixed-size output and, to be cryptographically secure, such outputs should be highly unpredictable. More formally, a cryptographic hash function H should satisfy three main properties:

- *Pre-image resistance*: given an image y of H, it is computationally unfeasible to find a pre-image x such that $H(x) = y$;
- *Second pre-image resistance*: given (x, y) such that $H(x) = y$, it is computationally unfeasible to find a second pre-image x' satisfying $H(x') = H(x) = y$;
- *Collision resistance*: it is computationally unfeasible to find x and x' such that $H(x) = H(x')$.

Even though these properties are required in a wide variety of scenarios, some hash-based digital signatures schemes (e.g., W-OTS+[Hülsing 2013] and XMSS[Buchmann et al. 2011]), are designed to eliminate the need for collision resistance in its internal construction. In other words, a collision-resistant hash function is usually still required to process the message itself, thus producing the hash value to be signed; this means that the size of the hash would be $2k$ bits for a classical security level of k bits, or $3k$ for the same security level in a quantum scenario [Grover 1996]. Internally, however, the signature scheme can use k-bit hash functions for a classical security level of k bits, and a $2k$-bit hash function for k bits of quantum security.

In the following subsections, we discuss one-time signature (OTS) schemes and many-times signature (MTS) schemes that can be built upon them, as well as some of the main optimization strategies proposed in the literature for both OTS and MTS.

2.1. One-time Signature (OTS) schemes

Originally, hash-based signature schemes were only One-Time Signature (OTS), i.e., for each public-private key pair, only one message could be signed. The first publicly known OTS scheme was the Lamport-Diffie (LD) OTS[Lamport 1979], which was basically a bit-signing scheme: each bit of the message was signed independently, so it required as many private and public key elements as the message length. As a consequence, it had large public and private key sizes, as well as large signatures. In addition, being able to sign a single message per key pair is not very practical in most real-world scenarios, since the management of disclosed public keys (e.g., by provisioning digital certificates) becomes cumbersome [Bernstein et al. 2008].

In 1979, Ralph Merkle proposed a new OTS called W-OTS[Merkle 1990], which introduces a trade-off between signature size and processing time. W-OTS works by using hash chains instead of single hashes over the secret key bit strings, and has a adjustable parameter w, called the Winternitz parameter. This parameter is useful for adjusting the hash chain length and, thus, the trade-off between signature size and processing time. More formally, the W-OTS scheme can be described as follows.

Let s be the length of the message digest d to be signed and w the Winternitz parameter. Compute the parameters L_1 and L_2 as:

$$L_1 = \left\lceil \frac{s}{w} \right\rceil, \quad L_2 = \left\lceil \frac{\lfloor \log_2 L_1 \rfloor + w + 1}{w} \right\rceil, \quad L = L_1 + L_2.$$

Let $f : \{0,1\}^n \to \{0,1\}^n$ be a one-way function and $g : \{0,1\}^* \to \{0,1\}^n$ a cryptographic hash function, where n is the output size of the two functions. Let PRF be a cryptographically secure pseudo-random function that, on input of a n-bit seed SEED_{in}, outputs a random number RAND and an updated seed SEED_{out}, both with bit length n.

The original W-OTS is a signature scheme $\textsc{Sign} = (\textsc{Gen}, \textsc{Sig}, \textsc{Ver})$, where:

Key generation ($\textsc{Gen}$): generates the private key $sk = X = (x_{L-1}, \cdots, x_1, x_0)$ (pseudo-) randomly, then calculates the public key $pk = Y = (y_{L-1}, \cdots, y_1, y_0)$, where $y_i = f^{2^w-1}(x_i)$. In other words, each part of the public key is linked to the corresponding parts of the private via a $(2^w - 1)$-long hash chain.

Signing ($\textsc{Sig}$): the signer hashes the message into a fixed-length digest d using the cryptographic hash function g and calculates the checksum c as

$$c = \sum_{i=0}^{L_1-1} 2^w - m_i$$

where m_i are the w-bit blocks of d (e.g. $d = m_{L_1}||m_{L_2}||\cdots||m_1||m_0$) interpreted as binary integers. Then, the checksum is appended to the digest and the resulting bit string is split into L blocks of length w:

$$d||c = b_{L-1}||b_{L-2}||\cdots||b_1||b_0$$

Each block b_i is seen by the algorithm as the binary representation of an integer. The signature is then $\sigma = (\sigma_{S-1}, \sigma_{S-2}, \cdots, \sigma_1, \sigma_0)$, where $\sigma_i = f^{b_i}(x_i)$.

Signature verification ($\textsc{Ver}$): With the message and the signature σ, the verifier computes the digest d and the checksum c, and calculates $d||c = b_{L-1}||b_{L-2}||\cdots||b_1||b_0$ just like in the signing operation. Then, the verifier reconstructs the public key $Y' = (y'_{L-1}, \cdots, y'_1, y'_0)$ where $y'_i = f^{2^w-1-b_i}(\sigma_i)$. The signature is accepted only if $Y' = Y$.

Figure 1 illustrates the W-OTS public and private keys, as well as its hash chains. A pseudo-random function is used to generate all x_i of the private key, so in practice the only information the signer needs to store as its private key is the seed of the generator.

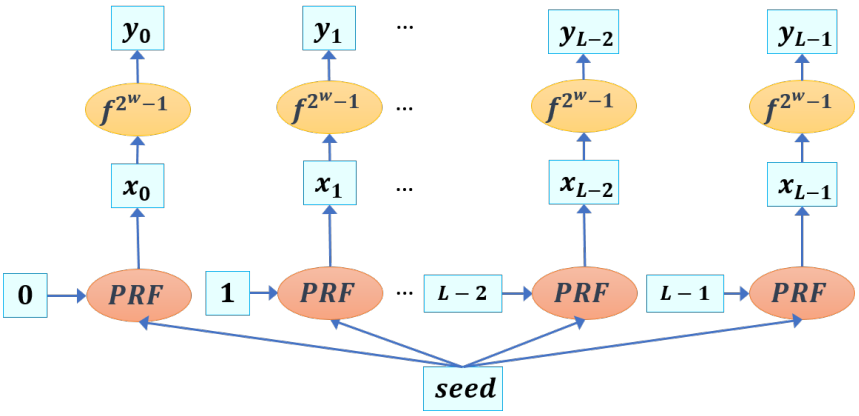

Figure 1. The conventional W-OTS with PRF.

When compared with Lamport-Diffie OTS (LD-OTS), W-OTS leads to smaller signature sizes at the cost of extra processing, using $(2^w - 1)$-long hash chains to sign w bits simultaneously. For example, for $w = 2$, the scheme signs pairs of bits, so processing each pair takes 0, 1, 2 or 3 hash operations depending on whether the bits to be signed are 11, 10, 01 or 00, respectively; for $w = 3$, signing 3 bits would then take 0 to 7 hash

operations for the bit strings 111 to 000, and so on. As a result, a linear decrease in the signature size results in an exponential growth on the hash chains and, thus, on the processing times of the signature and verification procedures. For this reason, in practice the recommended values of the w parameter usually ranges from 1 to 4.

After W-OTS, other authors have introduced many other OTS. One example is the proposal by BiBa [Perrig 2001], which was the first hash-based signature scheme that departed from the W-OTS design by incorporating collisions as part of its working principle. Its advantages included short signature sizes and fast verification times, at the cost of larger public keys and slower key generation. BiBa was subsequently superseded by HORS [Reyzin and Reyzin 2002], which is actually a Few-Times Signature (FTS) scheme: it can be used a few times without degrading too much its security, depending on the choice of parameters. HORS provides much better signature generation times than BiBa while maintaining comparable private key, public key and signature sizes, besides also having faster verification times. Compared to W-OTS, however, HORS received less attention until recently because, despite having smaller signatures, it leads to larger public keys and has one important disadvantage: its public key cannot be derived from the signature itself, which in principle makes it hard to use HORS in a many-times signature scheme using Merkle-trees, as discussed in what follows.

2.2. Many-times Signature (MTS) schemes

To be useful in practice, any digital signature solution needs to allow many messages to be signed with a single public/private key pair. For this reason, modern hash-based signature proposals build upon OTS schemes to create a Many-Times Signature (MTS) solution, capable of signing a large (albeit limited) number of messages under the same public key. The most common approach for this is to rely on Merkle-trees [Merkle 1990], which were originally proposed for use with W-OTS but can be adopted for use with other OTS schemes. Basically, a Merkle-tree is a binary hash-tree where the leaf nodes are the public keys of the one-time signature schemes and, in the inner nodes, the parent node is a concatenation of its left and right children nodes (Figure 2 shows a Merkle-tree of height $h = 3$). With this construct, given an OTS signature, its index and its authentication path, the root node of the tree can be reconstructed. This effectively reduces all OTS public keys to a single MTS public key: the root of a binary hash tree. Since the MTS's private keys can also be computed from a single seed and a suitable pseudo-random generator, the resulting public/private key pair requires very little storage.

An MTS scheme using a Merkle-tree works as follows: during key generation, the underlying OTS's private and public keys are generated first; then, the public keys are hashed together in a tree-like fashion, leading to the MTS public key, whereas the MTS private key is the collection of all OTS private keys. Whenever a message needs to be signed, the signer employs an OTS key pair that has never been used before. Verifiers can then check the OTS signature as long as they also receive the OTS's index and authentication path in the Merkle tree.

Even though a single Merkle-tree construction can be employed to allow the generation of an arbitrarily high number of signatures, the resulting scheme would not scale well because increasing the tree height increases the number of underlying OTS exponentially, which results in a costly private key generation process for the entire MTS. For

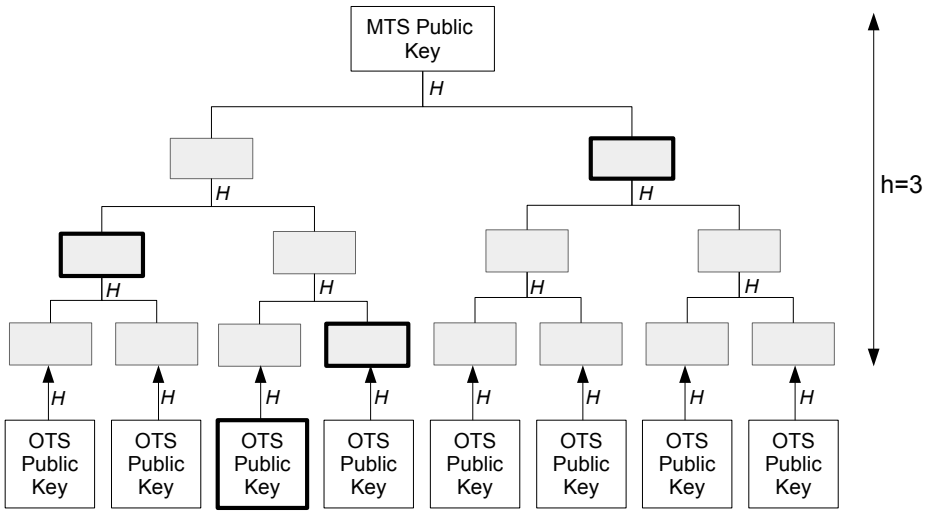

Figure 2. A Merkle-tree with $h = 3$ as a MTS: the individual OTS public keys are hashed ("*H*") together in a binary tree structure until the is obtained. Highlighted boxes represent the authentication path for the highlighted OTS public key (needed o reconstruct the MTS public key from the signature).

example, an electronic messaging application that requires generating millions of signatures every day would require an MTS with many millions of underlying OTS. All those OTS public keys would have to be built during the key generation phase, since the root of the Merkle tree can only be computed if all of its leaves are provided. Hence, the cost of generating the MTS public key supporting 2^n signatures would be $O(2^n)$: it corresponds to the generation of keys for 2^n individual OTS, plus approximately 2^n hash operations for building the Merkle-tree. In practice, that could lead to performance hits.

To remedy this, CMSS[Buchmann et al. 2006] proposes to decompose the original Merkle-tree into many layers of smaller trees, which are joined by intermediate OTS. In other words, the leaves of upper layer trees contain OTS for signing roots of lower layer trees, similarly to certificate chains in a Public Key Infrastructure (PKI). This starts from the single tree in the topmost layer and goes down until the bottom layer trees, whose leaves are the OTS for signing actual messages. This construction has the advantage of requiring only the topmost tree to be generated during the key generation phase. As a drawback, it increases the overall signature size, because the signature for messages need to contain the signatures that authenticate the intermediate tree roots as well. This approach is incorporated in many modern MTS designs aimed at practical applications, such as XMSS$^{\mathrm{MT}}$[Hülsing et al. 2013].

2.3. Optimizations: signature size, processing time, and statefulness

As mentioned in Section 1, one of the main problems with hash-based signature schemes is their signature sizes, which are usually much larger than ECC-bases schemes even when we consider only the OTS schemes (i.e., without considering the overhead from revealing intermediate nodes of the Merkle tree in MTS schemes). One proposal for addressing this issue is to eliminate the need for collision-resistance, but requiring only pre-image

resistance. In practice, to attain a security level of k-bits, this approach allows the adoption of k-bit hash functions (instead of $2k$ bits) in a classical setting, or $2k$-bit hash functions (instead of $3k$ bits) in a quantum setting, thus leading to more compact signatures. This is the case of W-OTS+ scheme, an improvement over W-OTS that uses different random bit-masks for each hash function call, thus removing the need for collision-resistant hash functions. To accomplish this, the hash functions in the hash chains linking private and public keys are replaced by a "chaining function", which consists basically in XORing the input with a random chain level mask before calculating those functions.

Another general optimization proposed over pioneering hash-based signature schemes refers to the encoding of the messages to be signed. For example, in [Steinwandt and Villányi 2008] the authors propose that messages should first be hashed together with a counter $r \geq 0$, similarly to what is suggested in [Perrig 2001]. The goal is to obtain a hash value d satisfying certain conditions, namely that there must be a maximum and minimum number of non-overlapping runs of consecutive 0s and 1s, each of which having a maximum length ℓ. Then, d is encoded into hash chains considering the run-lengths, not the integer values of each group of w bits. For example, if d starts with 11110011, the signature part corresponding to those bits would be given by $(H^{\ell-4}(x_0), H^{\ell-2}(x_1), H^{\ell-2}(x_2))$, where (x_0, x_1, x_2) are parts of the private key and H is a cryptographic hash function. The goal in this case is to speed up the verification time at the cost of longer signature generation times. This happens for two reasons: first, the signer needs to find a suitable counter r for satisfying the scheme's constraints; second, since short runs appear more often than long runs, the signer ends up computing a larger portion of the hash chain that links the private key x_i to the public key y_i (i.e., the number of hash calls at the signer is closer to ℓ, whereas at the verifier it is closer to 0).

Finally, a somewhat orthogonal concern with hash-based signatures is that they lead to stateful systems, i.e., the signer must keep track of used keys to prevent reuse [Bernstein et al.]. After all, since most MTS schemes are built over OTS schemes, their security relies on the fact that no key is used twice, since that would fatally break the security of the system (e.g., allowing signatures to be forged). One of the few schemes in the literature that addresses this problem is SPHINCS [Bernstein et al.], a stateless hash-based signature solution that relies on a large hyper-tree structure with HORST, a variant of the HORS few-times signature scheme. The price paid by SPHINCS for its statelessness, however, is having even larger signatures than its stateful counterparts: about 41 KB for SPHINCS-256, which provides 128 bits of quantum security.

3. Compacting signatures with the Clipping-tree

Aiming to allow shorter signatures, in this section we describe a novel technique that builds upon the manner by which private keys are usually generated from a seed in hash based schemes. Specifically, most OTS hash-based signature schemes use a single layer of pseudo-random numbers generated from a seed, using the resulting x_i as the private key (see Figure 1, which shows this approach for the W-OTS scheme). Then, during signature generation, x_i is sometimes revealed directly, whereas other times what is revealed is $f^j(x_i)$ for some j. The basic idea of the hereby proposed "clipping technique" is to leverage on situations where consecutive parts of the private key (say, x_i and x_{i+1}) would be naturally revealed. In such cases, we can reveal a single piece of information that allow the generation of both x_i and x_{i+1}, thus compressing the total signature size. This is ac-

complished by using two layers of pseudo-random functions to generate when computing the scheme's private keys. As such, the technique applies to W-OTS+, W-OTS and any other OTS that relies on pseudo-random generators for deriving secret values that may be revealed as part of the signature itself.

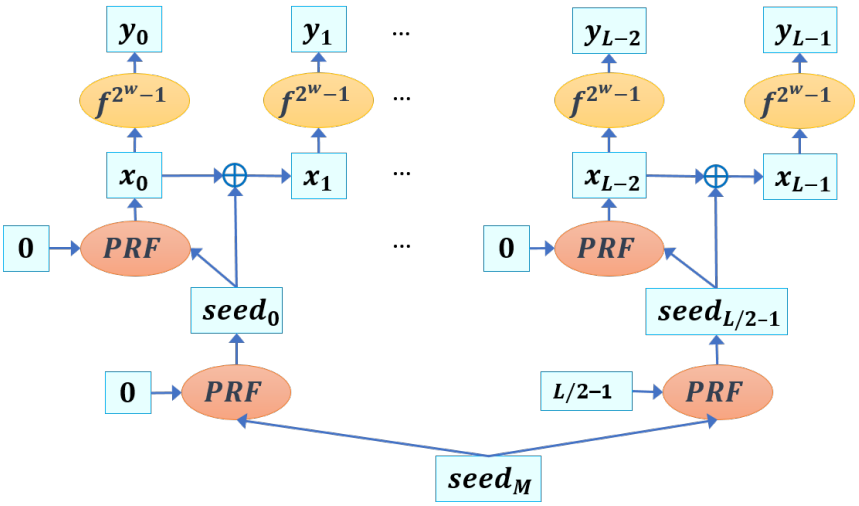

Figure 3. Our improved W-OTS scheme with PRF, where $x_{2i+1} = seed_i \oplus x_{2i}$. For example, x_1 is generated by calculating $seed_0 \oplus x_0$.

Figure 3 illustrates the proposed approach, using W-OTS as the base scheme (since it is simpler to explain). In a nutshell, the main modification on the scheme lies in the fact that the OTS's private values x_i are not derived directly from a seed. Instead, a master seed (denoted $seed_M$) is employed in the generation of a single layer of intermediate seeds $seed_i$, each of which is subsequently employed for generating pairs of private values x_{2i} and x_{2i+1}. Then, whenever a signature should reveal those pair of private values sharing the same lower layer seed, this seed is revealed in their place.

More formally, still using W-OTS as basis, the hereby proposed clipping technique can be described as follows. Given n, w, f, g and PRF as defined in the W-OTS scheme, calculate L_1, L_2 and L also like in the original W-OTS. The resulting signature scheme is the triple $\text{SIGN} = (\text{GEN}, \text{SIG}, \text{VER})$, where:

Key generation (GEN): Use two layers of pseudo-random sequences to generate the L pseudo-random integers $(x_0, \cdots, x_{L-1})$. For the first layer, the user selects an n-bit integer as the master seed $seed_M$ (the private key of the OTS). Using that seed, generate the first layer of n-bit pseudo-random values $seed_i$, for $0 \leq i < L/2$. Then, for the second layer, generate the private value x_{2i} as $PRF(seed_i)$, and its companion private value $x_{2i+1} = seed_i \oplus x_{2i}$. As a result, the second layer contains the private values $X = (x_{L-1}, \cdots, x_1, x_0)$, from which each part of the public key $pk = Y = (y_{L-1}, \cdots, y_1, y_0)$ are computed as $y_i = f^{2^w-1}(x_i)$, as usual.

Signing (SIG): The signer calculates

$$d||c = b_{L-1}||b_{L-2}|| \cdots ||b_1||b_0$$

and each block b_i is interpreted as the binary representation of an integer, just like in the W-OTS scheme. The difference to W-OTS, however, is that instead of processing each

block of $d||c$ separately, they are processed two blocks at a time, either sequentially or in parallel. More precisely, for each pair of blocks b_{2i} and b_{2i+1}, where $0 \leq i \leq L/2$, if either b_i or b_{i+1} is non-zero, the usual hashing chain is used: $f^{b_{2i}}(x_{2i})$ and $f^{b_{2i+1}}(x_{2i+1})$ are published with the signature, and the signer then proceeds with the next two blocks b_{2i+2} and b_{2i+3}. However, if both b_{2i} and b_{2i+1} are sequences of zero bits, the shared seed in the lower layer is published instead of $f^{b_{2i}}(x_{2i})$ and $f^{b_{2i+1}}(x_{2i+1})$ and the signer proceeds with the next two blocks b_{2i+2} and b_{2i+3}. The signer repeats the same procedure for all pairs of blocks until the entire $d||c$ is processed. The resulting signature $\sigma = (\sigma_{S-1}, \sigma_{S-2}, \cdots, \sigma_1, \sigma_0)$ is sent alongside the message. Note that, with this technique, the signature can have variable length depending on the message to be signed.

Signature verification (VER): With the message and the signature σ, the verifier computes the digest d and the checksum c, and calculates $d||c = b_{L-1}||b_{L-2}||\cdots||b_1||b_0$ just like in the original verification operation. The verifier then proceeds considering pairs of blocks, which can be processed either sequentially or in parallel. For a sequential verification, let $i = 0$ and $k = 0$ be indices for the blocks from $d||c$ and from the signature σ, respectively. For each pair of blocks b_{2i} and b_{2i+1} in $d||c$, where $0 \leq i \leq L/2$, if either block is non-zero the verification is done as usual, by evaluating $y'_{2i} = f^{2^w-1-b_{2i}}(\sigma_k)$ and $y'_{2i+1} = f^{2^w-1-b_{2i+1}}(\sigma_{k+1})$; the verifier then proceeds to the next two blocks, updating the i and k indices by making $i = i + 2$ and $k = k + 2$. However, if both b_{2i} and b_{2i+1} are zeros, the verifier first calculates $x'_{2i} = PRF(\sigma_k)$ and $x'_{2i+1} = x'_{2i} \oplus \sigma_k$, then calculates $y'_{2i} = f^{2^w-1}(x'_{2i})$ and $y'_{2i+1} = f^{2^w-1}(x'_{2i+1})$; then, i and k are updated by $i = i + 2$ and $k = k + 1$, since a single block from the signature was used to process two blocks from $d||c$. The verifier repeats this process until all blocks have been processed; if this process results in exactly L values $(y'_0, \cdots, y'_{L-1})$ and $Y' = (y'_0, \cdots, y'_{L-1}) = Y$, the signature is valid; otherwise, it is invalid.

A parallel verification follows an analogous process: (1) start by identifying pairs of blocks b_{2i} and b_{2i+1} from $d||c$ that are both zeros, thus learning the indices k of blocks from the signature that cover those pairs of blocks; (2) compute the corresponding $x'_{2i} = PRF(\sigma_k)$ and $x'_{2i+1} = x'_{2i} \oplus \sigma_k$; and finally (3) proceed with the usual verification process, checking whether $y'_i = f^{2^w-1-b_i}(\sigma_i) = y_i$ for the blocks not affect by the previous steps, and whether $y'_i = f^{2^w-1}(x'_i) = y_i$ for the affected blocks.

As a simplified example, suppose the digest of a message to be signed is $d = 00101100$ — here the checksum is omitted for the sake of simplicity, but it would need to be included in the message and signed as well. Using $w = 1$, the private key would be $X = (x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. In the traditional W-OTS scheme, the signature would include all eight x_i hashed zero or one time, thus leading to $\sigma = (f^0(x_0), f^0(x_1), f^1(x_2), f^0(x_3), f^1(x_4), f^1(x_5), f^0(x_6), f^0(x_7))$. With proposed clipping technique, however, there are two pairs of blocks that are both zeros: the first (b_0 and b_1) and the last (b_6 and b_7) one. Therefore, the corresponding seeds are published, so the resulting signature becomes $\sigma = (seed_0, f^1(x_2), f^0(x_3), f^1(x_4), f^1(x_5), seed_3)$. In this particular case, the signature becomes 25% shorter compared to the original W-OTS scheme (again, without taking the checksum into account for the sake of simplicity).

4. Analysis

In this section we evaluate the proposed clipping technique considering the resulting signature size, processing time, and underlying security, assuming it is applied to W-OTS+ or W-OTS. In summary, we show that the (probabilistic) reduction in signature size is attained with negligible performance overheads and no security impacts to those schemes.

4.1. Signature size

As described for W-OTS in Section 2.1, we assume that the hash-based signature scheme uses a PRF for generating the secret n-bit values x_i from a seed, which plays the role of the private key. The message to be signed is converted into the bit-string $d||c$, with L blocks of w bits, where each block is interpreted as an integer from zero to $2^w - 1$. In this scenario, the message digest d follows a uniform distribution, since this is a necessary condition for any secure cryptographic hash function; similarly, the checksum c obtained from the uniformly distributed blocks from d is also expected to be uniformly distributed. Therefore, all block values (from 0 to 2^{w-1}) are equally likely, and the probability of one w-bit block from $d||c$ being filled with zeros is $\frac{1}{2^w}$, whereas the probability of two adjacent blocks being both zero is $\frac{1}{2^{2w}}$.

In our scheme, due to the way the pseudo-random number layers are created, any two adjacent blocks b_{2i} and b_{2i+1} in $d||c$ are signed using secret values x_{2i} and x_{2i+1} that share the same seed. Therefore, the whole message to be signed can be seen as a concatenation of two-block units, and the analysis of how much data is omitted from the signature when the seed is revealed instead of each x_{2i} and x_{2i+1} gives the overall signature size reduction. For each two-block unit, there are two possibilities: if both blocks contain a zero value (i.e., with probability $1/2^{2w}$), then the contribution to the signature is a single n bit integer, the lower layer seed; otherwise (i.e., with probability $1 - 1/2^{2w}$), two n-bit strings are published, adding $2n$ bits to the signature. The average bit-length of each two-block unit when the clipping technique is applied is, thus:

$$\text{AvgLen} = 2n \times \frac{2^{2w} - 1}{2^{2w}} + n \times \frac{1}{2^{2w}} = \frac{2^{2w+1} - 1}{2^{2w}} n$$

The savings when compared with a regular, $2n$-bit unit can be calculated as:

$$\text{SAVINGS} = \frac{2n - \text{AvgLen}}{2n} = \frac{1}{2^{2w+1}}$$

Finally, since the whole signature is just concatenations of independent two-block units, the savings in terms of signature size is equivalent to the savings for each unit. For $n = 256$ bits and $w = 1$, for example, we have $\text{SAVINGS} = 12.5\%$. Note that the savings decrease exponentially with w, so smaller values of w lead to better compression. This is illustrated in Table 1, which shows the average savings for different values of w. Given its probabilistic nature, however, in practice the clipping technique may be useful for compressing signatures even for $w \geq 3$, depending on the actual data to be signed.

4.2. Processing time

In terms of processing, the proposed clipping technique affects basically the generation of the secret values x_i from a private key $seed_M$, either during the GEN and VER operations.

Table 1. Average signature size for the original W-OTS+ and the "clipped" W-OTS+, obtained with the proposed approach, depending on the w parameter. We assume a 256-bit (classical) security level.

w	Signature Size		Average Savings (s)
	Original W-OTS+	"Clipped" W-OTS+	
1	8192 bytes	7168 bytes	12.50%
2	4096 bytes	3968 bytes	3.13%
3	2731 bytes	2710 bytes	0.78%
4	2048 bytes	2044 bytes	0.20%

Even though the two-layer construction may appear to lead to a less efficient GEN than the original, single-layered approach, the fact is that both perform the same number of PRF calls per x_i. More precisely, as observed in Figure 1, each x_i takes one call to the PRF, whereas the clipping technique shown in Figure 1 takes two calls to the PRF plus one XOR operation for each *pair* (x_{2i}, x_{2i+1}). Since XOR operations have a negligible cost, both approaches end up being similar in terms of processing time.

The VER operation, on the other hand, would require one extra PRF call for each pair of blocks that can be "clipped". Assuming the same cost for PRF calls and hash function calls, the gains in terms of signature size translate to an equivalent overhead in terms of processing for $w = 1$: where two calls to the hash function would be made to compute (y'_{2i}, y'_{2i+1}) from $(\sigma_{2i}, \sigma_{2i+1})$, there is an extra call to the PRF (i.e., a 50% overhead) to recover (x'_{2i}, x'_{2i+1}); in this case, a pair of n-bit blocks is replaced by the corresponding n-bit seed (i.e., a 50% saving). For larger values of w, on the other hand, this extra PRF call becomes less pronounced when compared to the $2 \cdot (2^w - 1)$ hash function calls required to compute y'_{2i} and y'_{2i+1}. Hence, albeit the signature size gains obtained drop exponentially with w, so does the processing overhead, thus allowing interesting trade-offs.

As a final remark, another aspect of the proposed scheme refers to parallelism. Specifically, W-OTS+ and W-OTS is such that, with L or more cores, one could compute all values of x_i in parallel using a PRF. In comparison, in our proposal only half of the values of x_i can be computed simultaneously (the even ones), since the other half (the odd ones) is dependent on the former. Still, the performance impact is relatively minor, as it only adds a constant processing overhead (namely, one PRF call) in the processing flow when compared to the fully parallel approach.

4.3. Security

The overall security of the clipping technique is inherited from the hash-based signature scheme to which it is applied. After all, except for the manner by which the private key $seed_M$ is processed for generating the secret x_i, our proposal does not change anything on the signature scheme itself. Therefore, the main security aspect that needs to be assessed is whether the proposed two-layer construction reveals any information it should not reveal to attackers. In what follows, we show that this is not the case.

First, note that $seed_i$ is only explicitly revealed when both x_{2i} and x_{2i+1} need to be public anyway, i.e., when b_{2i} and b_{2i+1} are zeros. If only $x_{2i} = PRF(seed_i)$ is revealed, on the other hand, its companion $x_{2i+1} = seed_i \oplus x_{2i}$ cannot be obtained by attackers; after all, $seed_i$ acts as a one-time pad to protect the secrecy of x_{2i+1}, whereas computing

$seed_i$ from x_{2i} requires inverting the PRF. An analogous argument applies to the task of computing x_{2i} when x_{2i+1} is disclosed, since by symmetry $seed_i$ acts as a one-time pad for $x_{2i} = x_{2i+1} \oplus seed_i$. Finally, when neither x_{2i} or x_{2i+1} are disclosed, but rather $f^{\alpha}(x_{2i})$ and $f^{\beta}(x_{2i})$ for some $\alpha, \beta > 0$, their secrecy remains protected by the pre-image resistance of the hash function f, as usual. Therefore, we can conclude that the clipping technique does not bring any negative impact in terms of security.

4.4. Proof of concept

A proof of concept implementation was developed in C language using the OpenSSL library for the SHA256 hash function ($n = 256$ bits and $d = 256$ bits). It was compiled with the usual -O2 flag without any special hardware optimizations. Ten thousand KeyGen-Sign-Verify tests were performed in an Intel Core i5 750 machine with 6GB of RAM, which produced the results summarized in Figure 4. This figure shows that, for $w = 1$, signing was about 6% faster when using the technique, although the verification was about 14% slower. This occurs because, with the proposed technique, some PRF calls are moved from the signing operation to the verification operation. Other than that, all differences were well within their standard deviations, which shows that the smaller signature sizes obtained with the technique have a low impact in terms of performance.

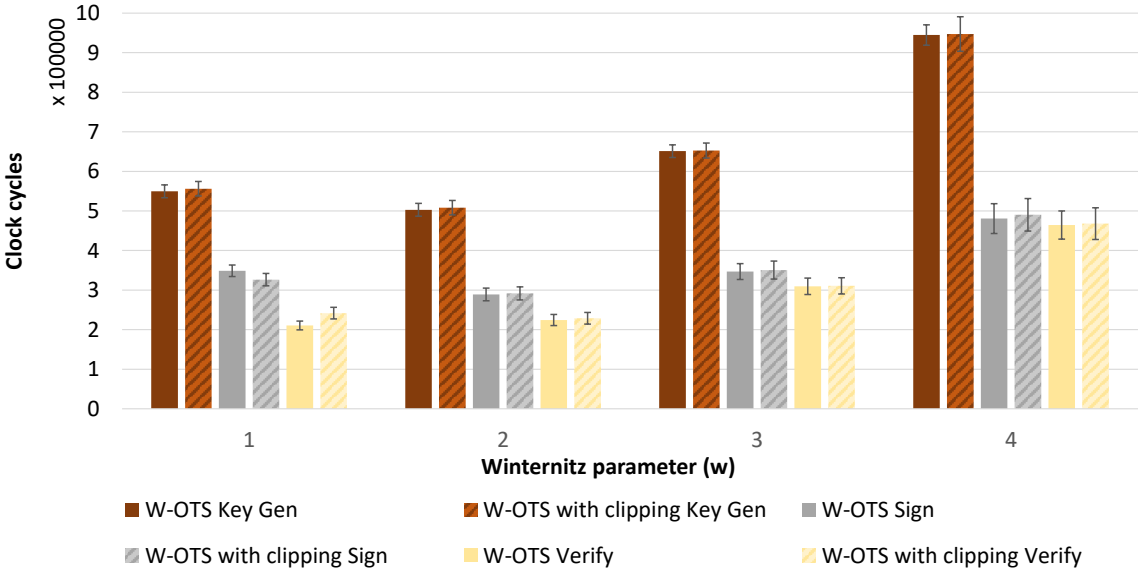

Figure 4. Summary of the performance comparison between W-OTS and W-OTS with the clipping technique. The standard deviation is shown in the error bars.

5. Conclusion

With the growing interest and investments in quantum technology by governments and private corporations, large scale quantum computers are growing closer to reality every year. This motivates the search for post-quantum cryptographic algorithms that can replace traditional schemes based on the (elliptic) discrete logarithm or number factorization, which would be vulnerable to attacks in a quantum scenario. Among the many existing proposals, hash-based digital signature schemes are quite promising, in particular due to their small public/private key pairs and to the fact that their security builds upon well-known properties of hash functions.

In this article, we propose a novel approach that uses two layers of PRFs when generating public keys from private keys. The resulting clipping technique enables the compression of signatures generated from state-of-the- art hash-based OTS schemes, such as W-OTS and W-OTS+, as well any MTS built upon them (e.g., XMSS and SPHINCS). The technique is probabilistic, meaning that the actual compression factor depends on the data to be signed; namely, one can obtain better compression rates for data where pairs of w-bit blocks processed by the OTS are filled with zeros, where w is the scheme's Winternitz parameter. On average, our proposal can save roughly 12.5% in the signature size when $w = 1$, whereas the savings are lower for larger values of w. Nevertheless, since the resulting processing overheads drop for larger w, in practice it provides flexibility for signers, who can choose whether or not to compress the signature depending on the most constrained resource (bandwidth or processing) at verifiers. Therefore, the proposed clipping technique further advances the practical adoption of hash-based signatures in the post-quantum era. To the best of our knowledge, this is the first proposal in the literature that takes advantage of the private/public key generation process to reduce hash-based signature sizes.

Acknowledgements: This work was supported by Intel Corporation and FAPESP (grant 2015/50520-6), and in part by CNPq (research productivity grant 301198/2017-9).

References

Anthony, S. (2017). IBM will sell 50-qubit universal quantum computer "in the next few years". `https://arstechnica.co.uk/gadgets/2017/03/ibm-q-50-qubit-quantum-computer`. Accessed: 2017-09-28.

Bernstein, D. J., Buchmann, J., and Dahmen, E. (2008). *Post-Quantum Cryptography*. Springer, Heidelberg, Deutschland.

Bernstein, D. J., Hopwood, D., Hülsing, A., Lange, T., Niederhagen, R., Papachristodoulou, L., Schwabe, P., and WilcoxO'Hearn, Z. SPHINCS: practical stateless hash-based signatures. LNCS.

Bernstein, D. J. and Lange, T. (2017). Post-quantum cryptography – dealing with the fallout of physics success. Cryptology ePrint Archive, Report 2017/314. `https://eprint.iacr.org/2017/314`.

Buchmann, J., Dahmen, E., and Hülsing, A. (2011). XMSS – A practical forward secure signature scheme based on minimal security assumptions. *Post-Quantum Cryptography*, 7071.

Buchmann, J., García, L. C. C., Dahmen, E., Döring, M., and Klintsevich, E. (2006). Cmss – an improved merkle signature scheme. In *Progress in Cryptology - INDOCRYPT 2006*, pages 349–363, Berlin, Heidelberg. Springer.

Ding, J. and Schmidt, D. (2005). Rainbow, a new multivariable polynomial signature scheme. *Int. Conf. on Applied Cryptography and Network Security (ACNS'05)*, 3531 of LNCS:164–175.

Dods, C., Smart, N. P., and Stam, M. (2005). Hash based digital signature schemes. *Cryptography and Coding*, 3796 of LNCS:96–115.

Goldreich, O., Goldwasser, S., and Halevi, S. (1997). Public-key cryptosystems from lattice reduction problems. *Advances in Cryptology – CRYPTO'97*, 1294:112–131.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proc. of the 28th annual ACM symposium on Theory of computing*, pages 212–219. ACM.

Hülsing, A. (2013). W-OTS+ – shorter signatures for hash-based signature schemes. In *Progress in Cryptology – AFRICACRYPT 2013*, pages 173–188, Berlin, Heidelberg. Springer.

Hülsing, A., Rausch, L., and Buchmann, J. (2013). Optimal parameters for xmss mt. In *Security Engineering and Intelligence Informatics*, pages 194–208, Berlin, Heidelberg. Springer.

Jao, D. and De Feo, L. (2011). Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *International Workshop on Post-Quantum Cryptography*, pages 19–34. Springer.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209.

Lamport, L. (1979). Constructing digital signatures from a one way function. Technical report, SRI International.

McEliece, R. J. (1978). A public-key cryptosystem based on algebraic coding theory. *DSN Progress Report*, 42:114–116.

Merkle, R. C. (1990). A certified digital signature. *Advances in Cryptology — CRYPTO'89*, pages 218–238.

Miller, V. S. (1986). Use of elliptic curves in cryptography. In *Advances in Cryptology — CRYPTO '85*, pages 417–426, Berlin, Heidelberg. Springer.

NIST (2001). *Federal Information Processing Standard (FIPS 197) – Advanced Encryption Standard (AES)*. National Institute of Standards and Technology.

NIST (2015). *Federal Information Processing Standard (FIPS 202) – SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. National Institute of Standards and Technology. DOI:10.6028/NIST.FIPS.202.

Perrig, A. (2001). The biba one-time signature and broadcast authentication protocol. In *Proc. of the 8th Conf. on Computer and Communications Security*, CCS '01, pages 28–37, New York, NY, USA. ACM.

Reyzin, L. and Reyzin, N. (2002). Better than biba: Short one-time signatures with fast signing and verifying. In *Information Security and Privacy*, pages 144–153, Berlin, Heidelberg. Springer.

Rivest, R., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126.

Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.*, 41(2):303–332.

Steinwandt, R. and Villányi, V. I. (2008). A one-time signature using run-length encoding. *Information Processing Letters*, 108(4):179–185.