A one-time signature using run-length encoding

Rainer Steinwandt and Viktória I. Villányi

Department of Mathematical Sciences, Florida Atlantic University,
777 Glades Road, Boca Raton, FL 33431, USA, {rsteinwa,vvillan}@fau.edu

Abstract

A one-time signature scheme using run-length encoding is presented, that in the random oracle model offers security against chosen-message attacks. For parameters of interest, the proposed scheme enables about 33% faster verification with a comparable signature size than a construction of Merkle and Winternitz. The public key size remains unchanged (1 hash value). The main price for the faster verification is an increase of the time for signing messages and for key generation.

Key words: cryptography, digital signature, one-time signature, hash function

1 Introduction

The design of digital signature schemes has received a lot of attention in the cryptographic literature. In recent years, there is an increasingly interest in signature schemes that can be implemented by means of symmetric primitives like a block cipher or a cryptographic hash function only, i.e., without relying on a number-theoretic assumption. One motivation is provided by Shor’s well-known quantum algorithms for factoring and discrete logarithms [21], but also specific efficiency needs in low power devices stimulate research along this line [20]. A common technique to obtain such signature schemes is to start with a one-time signature, which then—with a Merkle tree or a one-way chain—is augmented to a signature scheme that can handle a larger number of messages (cf., e.g., [15, 20, 6, 5]). Here the performance of the underlying one-time signature has significant impact on the efficiency of the obtained “multi-time” scheme, and the question for efficient one-time signatures naturally arises. Similarly, cryptographic constructions for password-authenticated key exchange as in [9] benefit from the availability of efficient one-time signatures.

Various approaches to materialize one-time signatures have been explored—including the work in [18, 11, 12, 13, 3, 1, 2, 17, 14, 19, 16]. Comparisons among different constructions can be found in [14, 7, 20, 16], and naming a single superior scheme seems non-trivial. Depending on the application requirements, preferences may differ, and subsequently, we focus on a construction of Merkle and Winternitz [12, 13], which is used in [8, 6, 5], for instance. Dods et al. in [7] recommend the use of this scheme, too.
After recalling the main ingredients of the Merkle-Winternitz construction, in Section 3 we describe our new scheme, which relies on a run-length encoding of a hash value. To analyze the security, we resort to a random oracle model. Section 4 discusses the efficiency and gives parameters of interest. It turns out that for relevant parameter choices the new scheme enables a significantly faster verification than the Merkle-Winternitz construction without sacrificing the attractive public key length of a single hash value. The main price we pay is a (moderate) increase in the expected cost for signature and key generation.

2 The Merkle-Winternitz one-time signature scheme

Subsequently we denote by $H : \{0, 1\}^* \rightarrow \{0, 1\}^s$ a cryptographic hash function, which will be modeled as a random oracle. We write $H^i$ for $i$-fold functional composition of $H$:

$$H^i(x) := H(H(\ldots H(x)\ldots)),$$

where $H^0(x) := x$.

The Merkle-Winternitz scheme is an improvement of a construction of Merkle [12, 13] and is parametrized by a positive integer $w$, which allows a trade-off between space and time. Figure 1 summarizes key generation, signature generation and signature verification of the Merkle-Winternitz construction, where

$$t := \left\lceil \frac{s}{w} \right\rceil + \left\lceil \log_2 \left\lceil \frac{s}{w} \right\rceil + 1 + w \right\rceil \frac{R}{w}$$

and $\rightarrow$ denotes the selection of an element uniform at random.

The public key size of only $s$ bit (one hash value) is quite attractive, and it can be derived by means of

$$t \cdot 2^w - 1 + 1$$

applications of $H$ in the key generation. If in a (message-independent) off-line phase we store the possible $H^i(x_j)$-values ($1 \leq i \leq 2^w - 1, 1 \leq j \leq t$) in a look-up table, the (on-line) computation of a signature requires only a single application of $H$. The exact number of applications of $H$ when verifying a (message, signature)-pair $(M, \sigma)$ depends on the Hamming weight of $H(M)$. For an attack aiming at a denial of service this could be of interest, and in the scheme proposed below the number of applications of $H$ in the verification does not depend on the particular (message, signature)-pair. Taking $H(M)$ for uniformly at random distributed, the expected number of applications of $H$ in the verification phase of the Merkle-Winternitz scheme computes to

$$t \cdot \frac{2^w - 1}{2} + 2$$

(1)
### Key generation:
**Input:** block width $w$
**Output:** verification key $pk$, signing key $sk$

\[
\begin{align*}
(x_1, \ldots, x_t) &\leftarrow \{0, 1\}^s \\
(y_1, \ldots, y_t) &\leftarrow (H^{2^w-1}(x_i))_{i=1}^t \\
sk &\leftarrow (x_1, \ldots, x_t) \\
(pk &\leftarrow H(y_1 \| \cdots \| y_t) \\
\text{return } (pk, sk)
\end{align*}
\]

### Signature generation:
**Input:** message $M$, signing key $sk = (x_1, \ldots, x_t)$
**Output:** a signature $\sigma$ for $M$ under $sk$

\[
\begin{align*}
m &\leftarrow H(M) \\
\text{pad } m \text{ with zeros from the left, so that its length is a multiple of } w \\
\text{split the (padded) bitstring } m \text{ into } w\text{-bit blocks } b_1, \ldots, b_{\lceil s/w \rceil} \\
treat each } b_i \text{ as binary representation of a } w\text{-bit integer and compute} \\
c &\leftarrow \text{binary representation of } \sum_{i=1}^{\lceil s/w \rceil} (2^w - b_i) \\
\text{pad } c \text{ with zeros from the left, so that its length is a multiple of } w \\
\text{split the (padded) bitstring } c \text{ into } w\text{-bit blocks } b_{\lceil s/w \rceil + 1}, \ldots, b_t \\
\sigma &\leftarrow (H^{b_1}(x_1), \ldots, H^{b_t}(x_t)) \\
\text{return } \sigma
\end{align*}
\]

### Signature verification:
**Input:** message $M$, verification key $pk$, signature $\sigma = (\sigma_1, \ldots, \sigma_t)$
**Output:** true, if $\sigma$ is a valid signature for $M$, false else

\[
\begin{align*}
\text{find } (b_i)_{i=1}^t \text{ as during signing} \\
(\gamma_1, \ldots, \gamma_t) &\leftarrow (H^{2^w-1-\sigma_i})_{i=1}^t \\
\text{if } H(\gamma_1 \| \cdots \| \gamma_t) &\neq pk \\
\text{then return true} \\
\text{else return false}
\end{align*}
\]

### 3 A scheme with faster verification

The Merkle-Winternitz scheme makes no assumption about the hash value $H(M)$. The construction discussed below assumes the hash value to have a particular structure, however. Similarly, as in the BiBa scheme [17], we will use a counter $r$ to ensure this: Starting with $r = 0$, we increase $r$ until $H(M \| r)$ has the desired structure. While this repeated hashing increases the cost for signing a message, it has no significant influence on the verification part: as in BiBa, the correct value $r$ will be part of the signature.

**Remark 1** For a hash function like SHA-1 which uses the Merkle-Damgård construction it is not necessary to process the complete (and potentially long) string $M \| r$ for each new counter value $r$ “from scratch”: Storing the internal state of the hash function after completing the “$M$-part”, it is enough to apply the compression function with the “$r$-part”.

---

<table>
<thead>
<tr>
<th>Key generation:</th>
<th>Signature verification:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> block width $w$</td>
<td><strong>Input:</strong> message $M$, verification key $pk$, signature $\sigma = (\sigma_1, \ldots, \sigma_t)$</td>
</tr>
<tr>
<td><strong>Output:</strong> verification key $pk$, signing key $sk$</td>
<td><strong>Output:</strong> true, if $\sigma$ is a valid signature for $M$, false else</td>
</tr>
<tr>
<td>$(x_1, \ldots, x_t) \leftarrow {0, 1}^s$</td>
<td>find $(b_i)_{i=1}^t$ as during signing</td>
</tr>
<tr>
<td>$(y_1, \ldots, y_t) \leftarrow (H^{2^w-1}(x_i))_{i=1}^t$</td>
<td>$(\gamma_1, \ldots, \gamma_t) \leftarrow (H^{2^w-1-\sigma_i})_{i=1}^t$</td>
</tr>
<tr>
<td>$sk \leftarrow (x_1, \ldots, x_t)$</td>
<td>if $H(\gamma_1 | \cdots | \gamma_t) \neq pk$</td>
</tr>
<tr>
<td>$pk \leftarrow H(y_1 | \cdots | y_t)$</td>
<td>then return true</td>
</tr>
<tr>
<td>\text{return } (pk, sk)</td>
<td>else return false</td>
</tr>
</tbody>
</table>

---

3
Key generation:

Input: parameters $R_{\text{min}}$, $R_{\text{max}}$, $\ell$

Output: verification key $pk$, signing key $sk$

$$(x_1, \ldots, x_{R_{\text{max}}+1}) \xleftarrow{\$} \{0, 1\}^s$$

$$(y_i)_{i=1}^{R_{\text{max}}} \leftarrow (H^{\ell-1}(x_i))_{i=1}^{R_{\text{max}}}$$

$$y_{R_{\text{max}}+1} \leftarrow H^{R_{\text{max}}-R_{\text{min}}}(x_{R_{\text{max}}+1})$$

$$sk \leftarrow (x_1, \ldots, x_{R_{\text{max}}+1})$$

$$pk \leftarrow H(y_1 \parallel \cdots \parallel y_{R_{\text{max}}+1})$$

return ($pk, sk$)

Signature verification:

Input: message $M$, verification key $pk$, signature $(\sigma, r)$ with $\sigma = (\sigma_1, \ldots, \sigma_{R_{\text{max}}+1})$

Output: true, if $(\sigma, r)$ is a valid signature for $M$, false else

$$m \leftarrow H(M \parallel r)$$

find $\Delta$, $(b_i)_{i=1}^{R_{\text{max}}}$ as during signing

if $\mu < 0$ or $b_1 + \cdots + b_{R_{\text{max}}-\Delta} \neq s$
then return false
else
$$\gamma_{R_{\text{max}}+1} \leftarrow H^{\mu}(\sigma_{R_{\text{max}}+1})$$

$$(\gamma_1, \ldots, \gamma_{R_{\text{max}}}) \leftarrow (H^{b_i-1}(\sigma_i))_{i=1}^{R_{\text{max}}}$$

if $H(\gamma_1 \parallel \cdots \parallel \gamma_{R_{\text{max}}+1}) = pk$
then return true
else return false

Signature generation:

Input: message $M$, signing key $sk = (x_1, \ldots, x_{R_{\text{max}}+1})$

Output: a signature $(\sigma, r)$ for $M$ under $sk$

$$r \leftarrow 0$$

repeat

$$r \leftarrow r + 1; m \leftarrow H(M \parallel r)$$

until $R_{\text{min}} \leq \text{NumberOfRuns}(m) \leq R_{\text{max}}$ and $\text{MaxRunLength}(m) \leq \ell$

$$\Delta \leftarrow R_{\text{max}} - \text{NumberOfRuns}(m)$$

let $b_i$ be the length of the $i^{\text{th}}$ run in $m$ ($1 \leq i \leq R_{\text{max}} - \Delta$)

$$(b_{R_{\text{max}}-\Delta+1}, \ldots, b_{R_{\text{max}}}) \leftarrow (1, \ldots, 1)$$

$$(\sigma_1, \ldots, \sigma_{R_{\text{max}}}) \leftarrow (H^{\ell-b_1}(x_1), \ldots, H^{\ell-b_{R_{\text{max}}}}(x_{R_{\text{max}}}))$$

$$\sigma_{R_{\text{max}}+1} \leftarrow H^{\Delta}(x_{R_{\text{max}}+1})$$

$$\sigma \leftarrow (\sigma_1, \ldots, \sigma_{R_{\text{max}}+1})$$

return $(\sigma, r)$

Fig. 2. our new one-time signature scheme

Specifically, for system parameters $R_{\text{min}}$, $R_{\text{max}}$ and $\ell$, in the signature generation we can only handle hash values $m = H(M \parallel r)$ satisfying the following: the hash value $m$ consists of at least $R_{\text{min}}$ and at most $R_{\text{max}}$ non-overlapping runs of consecutive 0s and 1s, and each of these runs is of length $\leq \ell$. We take $H$ for a random oracle with an output length of $s$ bit, and unless specified otherwise, subsequently we use $\ell := \lfloor \log_2(s) - 1 \rfloor$, which corresponds to the longest expected run.
3.1 Key generation

In terms of the mentioned parameters, the key generation of the proposed one-time signature scheme can be summarized as in Figure 2. It is similar to the construction of Merkle and Winternitz, the main difference being that in the new scheme there is only one “checksum block” and the number of applications of $H$ is chosen differently. Figure 3 illustrates the basic steps of the signature computation. The main parameters of the proposed scheme can be summarized as follows:

**Secret key size:** $(R_{\text{max}} + 1) \cdot s$ bit

**Public key size:** $s$ bit

**Applications of $H$:** $\ell \cdot R_{\text{max}} - R_{\text{min}} + 1$

3.2 Signature generation

As mentioned already, we have to assume that the hash value $m$ of the message we sign consists of at least $R_{\text{min}}$ and at most $R_{\text{max}}$ non-overlapping runs of consecutive 0s or consecutive 1s and that each run is of length $\leq \ell$. Therefore the signature generation shown in Figure 2 first determines a value $r$ with $m = H(M \parallel r)$ having the desired properties. At this, the functions $\text{NumberOfRuns}(\cdot)$ and $\text{MaxRunLength}(\cdot)$ are defined in the obvious way:
Definition 2 Let \( m \in \{0,1\}^+ \), and write \( m \) as concatenation of non-overlapping runs of consecutive 0s or 1s, i.e., \( m = B_1 \parallel \cdots \parallel B_v \) such that
- \( B_i \in \{0\}^+ \cup \{1\}^+ \), and
- \( B_i \) and \( B_{i+1} \) have no substring in common (\( 1 \leq i \leq v - 1 \)).

Denoting by \( |B_i| \) the length of the run \( B_i \), we have \( \text{NumberOfRuns}(m) := v \) and \( \text{MaxRunLength}(m) := \max\{|B_1|, \ldots, |B_v|\} \).

Example 3 Suppose \( R_{\min} = 2 \), \( R_{\max} = 4 \), \( \ell = 3 \) and \( m = 00011100 \). Then \( m \) consists of the \( \text{NumberOfRuns}(m) = 3 \) runs ‘000’, ‘111’, ‘00’. We obtain \( \text{MaxRunLength}(m) = 3 \), \( \Delta = 4 - 3 = 1 \), \( b_1 = 3 \), \( b_2 = 3 \), \( b_3 = 2 \).

Once an acceptable \( m \) has been found, essentially a run-length encoding is computed: We define values \( b_1, \ldots, b_{R_{\max}} \) such that \( b_i \in \{1, \ldots, \ell\} \) is the length of the \( i \)th run of 0s or 1s in \( m \). If \( m \) has less than \( R_{\max} \) runs, the “superfluous” \( b_i \)-values \( b_{R_{\max} - \Delta + 1}, \ldots, b_{R_{\max}} \) are set to 1. The value \( \Delta \in \{0, \ldots, R_{\max} - R_{\min}\} \) encodes the difference between \( R_{\max} \) and the actual number of runs in \( m \).

Remark 4 It is worth noting that the condition imposed on the runs occurring in \( m \) can be checked by a single pass through the bit representation of \( m \). In this pass, all the values \( \Delta \) and \( b_1, \ldots, b_{R_{\max} - \Delta} \) can be determined.

Together with \( \Delta \), the values \( b_1, \ldots, b_{R_{\max}} \) describe \( m \) uniquely up to complementing all bits in \( m \): As we store run lengths only, the bit-wise complement of \( m \) results in the same \( b_i \)-values and an identical \( \Delta \)-value. From a practical perspective, the (factor \( \sqrt{2} \)) loss in security resulting from this ambiguity seems to be outweighed by the gain in performance.

In a message-independent off-line phase we can precompute a look-up table with all \( R_{\max} \cdot \ell - R_{\min} \) possible \( H(x_j) \) values. In this case, no further applications of \( H \) are needed after a suitable value for \( m \) has been found. Without such a look-up table, we obtain the following number of

(OFFLINE) applications of \( H \): \( R_{\max} \cdot \ell - R_{\min} \)

To derive the expected number of hashes in the precomputation, we model \( H \) as a random oracle. Then the probability \( P(\text{NiceHash}) \) of an \( s \)-bit hash value to consist of at least \( R_{\min} \) and at most \( R_{\max} \) runs of length \( \ell \) is

\[
P(\text{NiceHash}) = 2 \cdot \sum_{d=R_{\min}}^{R_{\max}} \frac{n_d}{2^s},
\]

where \( n_d \) denotes the number of partitions of an \( s \)-bit string into exactly \( d \) runs, each of which has a length \( \leq \ell \). Each \( n_d \) is given as the coefficient of \( X^s \) in the polynomial \( \left( \sum_{i=1}^\ell X^i \right)^d \in \mathbb{Z}[X] \). For parameters of interest, \( n_d \) can thus be computed easily by expanding this product in a computer algebra system like Magma [4], for instance. We obtain
Expected number of (online) applications of $H$: $\frac{1}{P(\text{NiceHash})}$

3.3 Signature verification

As the verifier learns the correct value $r$ from the signature to be checked, there is no precomputation needed, and we count the following number of Applications of $H$: $s + 2 - R_{\text{min}}$

This number is independent of the (message, signature)-pair considered. Typically $R_{\text{min}}$ is close to $s/2$, so we expect $\approx s/2$ applications of $H$ for a signature verification. Approximating in (1) the value $t$ by $s/w$, for the verification in the Merkle-Winternitz scheme we obtain $\approx \frac{(2^w - 1)s}{2w}$ expected applications of $H$. So for $w = 2$, we expect to save about $33\%$ of the applications of $H$.

Before looking at the performance of the suggested scheme more closely, in the next section we discuss the correctness and the security under a chosen-message attack.

3.4 Correctness and security

It is straightforward to check that any signature computed by the signature generation passes the verification successfully: For $i = 1, \ldots, R_{\text{max}}$, signer and verifier together apply $H$ exactly $\ell - 1$ times to $x_i$, which allows the verifier to recover the values $y_1, \ldots, y_{R_{\text{max}}}$ from the key generation. Similarly, the correct value $y_{R_{\text{max}} + 1}$ is recovered, as together signer and verifier perform $R_{\text{max}} - R_{\text{min}}$ applications of $H$ to $x_{R_{\text{max}} + 1}$.

To verify existential unforgeability under chosen-message attack, we resort to a random oracle model with a random oracle $H : \{0, 1\}^* \rightarrow \{0, 1\}^*$. By existential unforgeability under chosen-message attack we mean that there is no probabilistic polynomial time algorithm $A$ satisfying the following: $A$ obtains an honestly generated public key $pk$ as input, can then query a signature for an arbitrary message $M$, and produces with non-negligible probability a valid (message, signature)-pair with a message $\tilde{M} \neq M$. This security requirement can be taken for an adaptation of the usual security requirement for “multi-time” signatures [10].

Assume we have found an adversary $A$ that violates the above security requirement: Without loss of generality, we may assume that before the output of a forgery, $A$ at some point queries the signing oracle with a message $\hat{M}$ to obtain a signature $(\sigma, r)$. Let $\hat{M} \neq M$ be the message for which $A$ has managed to forge a signature $(\tilde{\sigma}, \tilde{r})$. We write $\sigma_i$ respectively $\tilde{\sigma}_i$ ($1 \leq i \leq R_{\text{max}} + 1$) for the components of $\sigma$ respectively $\tilde{\sigma}$. We also assume that $A$ verifies the signatures $(\sigma, r)$ and $(\tilde{\sigma}, \tilde{r})$, and we write $q_H$ for the total number of random oracle
queries submitted by $A$ (including queries used for verifying a signature).

Let $\rho := (\Delta, b_1, \ldots, b_{R_{\max}})$ and $\tilde{\rho} := (\Delta, \tilde{b}_1, \ldots, \tilde{b}_{R_{\max}})$ be the values derived from $m := H(M \parallel r)$ and $\tilde{m} := H(\tilde{M} \parallel \tilde{r})$ in the signature verification. The values $\rho$ and $\tilde{\rho}$ determine these hash values up to a simultaneous negation of all bits, and we denote by $\text{CollisionOrComplement}$ the event that $A$ finds messages $M \neq \tilde{M}$ with $\rho = \tilde{\rho}$. To bound the probability of $\text{CollisionOrComplement}$, we double the upper bound $q_H^2/2^{s+1}$ for finding an “ordinary” collision:

$$P(\text{CollisionOrComplement}) \leq \frac{q_H^2}{2^s}.$$

Suppose now that $\rho \neq \tilde{\rho}$, and let $\delta := \text{NumberOfRuns}(\tilde{m}) - \text{NumberOfRuns}(m)$ be the difference in the number of runs between $\tilde{m}$ and $m$. We consider two cases:

- Suppose that $\delta$ is positive. If

  $$H^{\text{NumberOfRuns}(\tilde{m}) - R_{\min}}(\tilde{\sigma}_{R_{\max}+1}) \neq H^{\text{NumberOfRuns}(m) - R_{\min}}(\sigma_{R_{\max}+1}),$$

  then $A$ has come up with a 2nd preimage of the public verification key $pk$ under $H$—the 1st preimage being $(\gamma_1 \parallel \cdots \parallel \gamma_{R_{\max}+1})$ as derived from $\sigma$ in the verification procedure. The probability of finding such a second preimage of the $s$-bit value $pk$ is bounded by

  $$1 - (1 - 2^{-s})^{q_H} \leq q_H \cdot 2^{-s}.$$  (2)

  So we may assume there is a value $0 \leq i \leq \text{NumberOfRuns}(m) - R_{\min}$ with

  $$H^i(H^i(\tilde{\sigma}_{R_{\max}+1})) = H^i(\sigma_{R_{\max}+1}),$$

  and one of the following two conditions must hold:

  - $H^i(\tilde{\sigma}_{R_{\max}+1}) \neq \sigma_{R_{\max}+1}$ (and therefore $i > 0$): This means that $A$ has found the 2nd preimage $H^{i+\delta-1}(\tilde{\sigma}_{R_{\max}+1})$ of $H(H^{i-1}(\sigma_{R_{\max}+1}))$ under $H$. Again, as in Equation (2) we see that the probability of finding such a second preimage can be bounded by

    $$1 - (1 - 2^{-s})^{q_H} \leq q_H \cdot 2^{-s}. $$

  - $H^i(\tilde{\sigma}_{R_{\max}+1}) = \sigma_{R_{\max}+1}$: In this case, $A$ has found a preimage of $\sigma_{R_{\max}+1}$, namely $H^{i-1}(\tilde{\sigma}_{R_{\max}+1})$, under $H$. As in Equation (2), this probability can be bounded by

    $$1 - (1 - 2^{-s})^{q_H} \leq q_H \cdot 2^{-s}$$—for a random oracle $H$ the probability of finding a preimage can be bounded in the same way as the probability of finding a second preimage.

- Suppose $\delta \leq 0$, i.e., we have $\text{NumberOfRuns}(\tilde{m}) \leq \text{NumberOfRuns}(m)$. As $\rho \neq \tilde{\rho}$ and at the same time $\tilde{b}_1 + \cdots + \tilde{b}_{R_{\max} - \Delta} = s$, there exists an index $i_0 \in \{1, \ldots, R_{\max}\}$ with $\tilde{b}_{i_0} = b_{i_0} + \delta'$ and $\delta' > 0$. Now we can argue analogously as in the previous case:
If \( H^{b_0-1}(\tilde{\sigma}_{i_0}) \neq H^{b_0-1}(\sigma_{i_0}) \), then \( A \) has constructed a 2\(^{nd}\) preimage of the public verification key \( pk \) under \( H \). As in Equation (2), we can bound the probability of such an event by \( q_H \cdot 2^{-s} \), and we may assume there is a minimal number \( 0 \leq i \leq b_{i_0} - 1 \) with
\[
H^i(H^d(\tilde{\sigma}_{i_0})) = H^i(\sigma_{i_0}).
\]

- In case of \( H^d(\tilde{\sigma}_{i_0}) \neq \sigma_{i_0} \) (and therefore \( i > 0 \)), this means that \( A \) has found the 2\(^{nd}\) preimage \( H^{i+d^*-1}(\tilde{\sigma}_{i_0}) \) of \( H(H^{i-1}(\sigma_{i_0})) \) under \( H \).
- In case of \( H^d(\tilde{\sigma}_{i_0}) = \sigma_{i_0} \), the adversary \( A \) has found a preimage of \( \sigma_{i_0} \), namely \( H^{d-1}(\tilde{\sigma}_{i_0}) \), under \( H \).

With the same reasoning as above, we can bound the probability of each of these events by \( q_H \cdot 2^{-s} \).

Summarizing the above discussion, we see that for a successful forgery, one of the following events must occur: CollisionOrComplement, the adversary \( A \) finds a second preimage of \( pk \), or the adversary \( A \) finds a preimage or second preimage of an output of \( H \) obtained during signing \( M \). Consequently, the probability for a successful forgery is bounded by
\[
\frac{q_H^2}{2^s} + (R_{\max} \cdot \ell - R_{\min} + 1) \cdot \frac{q_H}{2^s},
\]
which for a polynomial number of queries \( q_H \) is negligible (in \( s \)). It is worth noting that the collision resistance of \( H \) is only used when arguing that the message digests \( H(M \parallel \hat{r}) \) and \( H(M \parallel \tilde{r}) \) differ, and this accounts for the quadratic term \( q_H^2 \) in the above bound. All the other arguments refer to the construction of preimages or 2\(^{nd}\) preimages under \( H \). Consequently, other than for the derivation of the message digest, one could consider the substitution of \( H \) by a hash function \( h \) with output length smaller than that of \( H \). Accordingly, the secret \( x_i \)-values would be chosen to be equal to the smaller output length of \( h \).

## 4 Performance for parameters of interest

The construction of Merkle and Winternitz is one of the most popular one-time signatures. In [7], Dods et al. compare this proposal with a construction of Bleichenbacher and Maurer [1], taking into account experimental results, and they recommend to use the Merkle-Winternitz construction with \( w = 2 \) “since it is very fast, easy to implement and provides short signatures”. The implementation complexity of the scheme proposed in the previous section seems comparable to the Merkle-Winternitz construction, and Table 1 compares the new proposal with the Merkle-Winternitz construction for \( w = 2 \) and \( w = 3 \). We also included Lamport’s scheme [11] and the HORS scheme from
The latter offers very attractive signature size, signing and verification cost, but suffers from a large public key size and key generation cost.

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
 & \text{signature size} & \text{signing cost} & \text{verification cost} & \text{key generation cost} & \text{public key size} \\
\hline
\text{Lamport} & 160s & 1 & 161 & 320 & 320s \\
\text{Merkle-Winternitz} & 85s & 1 & 129.5 (expected) & 256 & s \\
\text{Merkle-Winternitz} & 57s & 1 & 201.5 (expected) & 400 & s \\
\text{HORS} & 16s & 1 & 17 & 1024 & 1024s \\
\text{new scheme} & 84.4s & 8.8 (expected) & 86 & 423 & s \\
\text{new scheme} & 80.4s & 16.4 (expected) & 92 & 405 & s \\
\hline
\end{array}
\]

\(^1\)more precisely, here we had to multiply with the length of a secret key component \(x_i\), instead of \(s\)

Table 1

| performance of different one-time signature schemes with \(s = 160\); “cost” specifies the (expected) number of applications of \(H\) and “size” is given in bit; in the new scheme we reserve (conservatively) \(0.4s = 64\) bit in the signature for the counter \(r\). The table builds on a hash output length of \(s = 160\) bit and considers a scenario where arbitrary length messages are signed by hashing them first to a 160 bit value. This hashing of the message is counted as 1 application of \(H\). A possible optimization using a second hash function \(h \neq H\) with shorter output length as indicated in the previous section is not taken into account. For the signing we assume a look-up table to be available. This look-up table can be obtained by simply storing intermediate results when computing the hash chains in the key generation. For HORS we use parameters put forward in [19].

The table illustrates the main characteristics of the new scheme:

- As in the Merkle-Winternitz scheme, the public key is a single hash value.
- For a comparable signature size as the Merkle-Winternitz construction with \(w = 2\), signing and key generation cost increase. The key generation cost is still much lower than for HORS, however.
- Unlike for the Merkle-Winternitz construction and for HORS, signature sizes \(\ll 80\)s bit are not practical (the event \texttt{NiceHash} becomes too unlikely).
• The verification cost in the new scheme is message-independent and about 2/3 of the expected cost in the Merkle-Winternitz construction with \( w = 2 \).

**Remark 5** The overall picture does not change much when using a hash function with \( s = 256 \), and Table 2 gives a comparison of the Merkle-Winternitz construction with the proposed scheme for such a setting. The table also lists a parameter choice for the new scheme with \( \ell = 6 \) (instead of \( \lceil \log_2(256) - 1 \rceil = 7 \)): For the price of an increased signing cost, this choice of \( \ell \) enables a significant reduction in the cost for key generation.

<table>
<thead>
<tr>
<th></th>
<th>signature size</th>
<th>signing cost</th>
<th>verification cost</th>
<th>key generation cost</th>
<th>public key size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merkle-Winternitz</td>
<td>133s</td>
<td>1</td>
<td>201.5 (expected)</td>
<td>400</td>
<td>s</td>
</tr>
<tr>
<td>((w = 2))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Merkle-Winternitz</td>
<td>90s</td>
<td>1</td>
<td>317 (expected)</td>
<td>631</td>
<td>s</td>
</tr>
<tr>
<td>((w = 3))</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>new scheme ((R_{\text{min}} = 123, R_{\text{max}} = 131))</td>
<td>132.4s</td>
<td>7.1 (expected)</td>
<td>135</td>
<td>795</td>
<td>s</td>
</tr>
<tr>
<td>new scheme ((\ell = 6, R_{\text{min}} = 122, R_{\text{max}} = 131))</td>
<td>132.4s</td>
<td>24.1 (expected)</td>
<td>136</td>
<td>665</td>
<td>s</td>
</tr>
</tbody>
</table>

Table 2 comparison of the proposed scheme and the Merkle-Winternitz construction with \( s = 256 \); “cost” specifies the (expected) number of applications of \( H \) and “size” is given in bit; in the new scheme we reserve (conservatively) \( 0.4s = 64 \) bit in the signature for the counter \( r \).

Summarizing, the new scheme appears to be a viable alternative to the Merkle-Winternitz construction, if short public key size and fast verification have priority. In absolute terms, the increased signing cost and more expensive key generation look still acceptable.

### 5 Conclusion

The one-time signature scheme presented has some similarity with the Merkle-Winternitz scheme, but builds on a run-length encoding instead of blocks of fixed size. For parameters of interest, this enables a faster and message-independent signature verification cost without giving up the short public key size of a single hash value. The main price is an increased key generation and signing cost. In scenarios mandating a short public key and fast verification, the suggested scheme seems an interesting option.
Acknowledgments

We thank the anonymous referees for valuable comments and suggestions that helped to improve the original manuscript.

References


