

XMSS – A Practical Forward Secure Signature Scheme based on Minimal Security Assumptions

First Version, 08. September 2011

Johannes Buchmann, Erik Dahmen, and Andreas Hülsing*
{buchmann,dahmen,huelsing}@cdc.informatik.tu-darmstadt.de

Cryptography and Computeralgebra
Department of Computer Science
TU Darmstadt

Abstract. We present the hash-based signature scheme XMSS. It is the first provably (forward) secure and practical signature scheme with minimal security requirements: a pseudorandom and a second preimage resistant (hash) function family. Its signature size is reduced to less than 25% compared to the best provably secure hash based signature scheme.

Keywords digital signature, practical, minimal security assumptions, hash-based signatures, forward security, provable security

1 Introduction

Digital signatures are a very important cryptographic tool. The signature schemes currently used in practice are RSA, DSA, and ECDSA. Their security depends on the security of certain trapdoor one-way functions which, in turn, relies on the hardness of factoring integers and computing discrete logarithms, respectively. However, it is unclear whether those computational problems remain hard in the future. In fact, it has been shown by Shor [28] that quantum computers can solve them in polynomial time. Other algorithmic breakthroughs are always possible in the future. In view of the importance of digital signatures it is necessary to come up with alternative practical signature schemes that deliver maximum security. In particular, quantum computers must not be able to break them. They are called post-quantum signature schemes.

In this paper we propose the hash-based signature scheme XMSS (eXtended Merkle Signature Scheme). It is based on the Merkle Signature Scheme [24] and the Generalized Merkle Signature Scheme (GMSS) [10].

* Supported by grant no. BU 630/19-1 of the German Research Foundation (www.dfg.de).

We show that XMSS is an efficient post-quantum signature scheme with minimal security assumptions.

This is done as follows. XMSS requires a hash function family \mathcal{H} and another function family F . We prove:

(Security) XMSS is existentially unforgeable under adaptively chosen message attacks in the standard model, provided \mathcal{H} is second preimage resistant and F is pseudorandom.

(Efficiency) XMSS is efficient, provided that \mathcal{H} and F are efficient. This claim is supported by experimental results.

The first assertion shows that the security requirements for XMSS are minimal. This follows from [27], [26], [19] and [17] where the existence of a secure signature scheme is proved to imply the existence of a second preimage resistant hash function family and a pseudorandom function family (see Section 3).

The second assertion shows that XMSS is practical as there are many ways to construct very efficient (hash) function families that are believed to be second preimage resistant or pseudorandom, respectively, even in the presence of quantum computers. For example, cryptographic hash functions and block ciphers can be used to construct such families. In particular, there are such constructions based on hard problems in algebra or coding theory. The huge number of instantiations of XMSS guarantees the long-term availability of secure and efficient signature schemes.

The idea of hash-based signatures was introduced by Merkle [24]. The results in [8,9,10,11,12,13,14,15,16,20,21,29] improve the Merkle idea in many respects by providing new algorithmic ideas and security proofs. XMSS incorporates many of those ideas and goes one step further. It is the first practical (forward) secure signature scheme with minimal security requirements in the above sense. All other constructions, but MSS-SPR [14], require a collision resistant hash function family whose existence is not known to follow from the existence of a secure digital signature scheme. Compared to MSS-SPR, which is the currently best hash-based signature scheme that carries a proof of security, we can reduce the signature size by more than 25 % for the same level of security. Furthermore MSS-SPR is not forward-secure. The improved signature size is very important as the signature size is considered the main drawback of hash-based signatures.

In this paper we show only how to sign fixed length messages. The scheme can easily be extended to sign messages of arbitrary length using TCR hash and sign as proposed in [14]. This requires a target collision resistant hash function family. Target collision resistant hash function

families are known to exist if any one-way function exists [27]. Therefore this preserves the minimal security assumptions property.

The paper is organized as follows. In Section 2 we describe the construction of XMSS. Its security and forward security is discussed in Sections 3 and 4. The XMSS-efficiency is shown in Section 5. Section 6 contains a description of our implementation and a presentation of the performance results.

2 The eXtended Merkle Signature Scheme XMSS

In this section we describe XMSS. Like the Merkle signature scheme [24] it uses a one-time signature scheme (OTS) that can only sign one message with one key. To overcome the limitation to one message per key, a hash tree is used to reduce the authenticity of many OTS verification keys to one public XMSS key. To minimize storage requirements, pseudorandom generators (PRG) are used. They generate the OTS signature keys as needed.

The parameters of XMSS are the following:

- $n \in \mathbb{N}$, the security parameter,
- $w \in \mathbb{N}, w > 1$, the Winternitz parameter,
- $m \in \mathbb{N}$, the message length in bits,
- $F(n) = \{f_K : \{0, 1\}^n \rightarrow \{0, 1\}^n | K \in \{0, 1\}^n\}$ a function family,
- $H \in \mathbb{N}$, the tree height, XMSS allows to make 2^H signatures using one keypair,
- h_K , a hash function, chosen randomly with the uniform distribution from the family $\mathcal{H}(n) = \{h_K : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n | K \in \{0, 1\}^n\}$,
- $x \in \{0, 1\}^n$, chosen randomly with the uniform distribution. The string x is used to construct the one-time verification keys.

Those parameters are publicly known.

We keep the following description of XMSS and its components short by including references to more detailed descriptions. We write \log for \log_2 .

Winternitz OTS As OTS we use the Winternitz OTS (W-OTS) first mentioned in [24]. We use a slightly modified version proposed in [9]. For $K, x \in \{0, 1\}^n$ and $e \in \mathbb{N}$ we define $f_K^e(x)$ as follows. We set $f_K^0(x) = K$ and for $e > 0$ we define $K' = f_K^{e-1}(x)$ and $f_K^e(x) = f_{K'}(x)$. In contrast to previous versions of W-OTS this is a (random) walk through the function

family instead of an iterated evaluation of a hash function. This modification allows to eliminate the need for a collision resistant hash function family.

Also, define

$$\ell_1 = \left\lceil \frac{m}{\log(w)} \right\rceil, \quad \ell_2 = \left\lceil \frac{\log(\ell_1(w-1))}{\log(w)} \right\rceil + 1, \quad \ell = \ell_1 + \ell_2.$$

The secret signature key of W-OTS consists of ℓ n -bit strings sk_i , $1 \leq i \leq \ell$ chosen uniformly at random. The public verification key is computed as

$$\text{pk} = (\text{pk}_1, \dots, \text{pk}_\ell) = (f_{\text{sk}_1}^{w-1}(x), \dots, f_{\text{sk}_\ell}^{w-1}(x)),$$

with f^{w-1} as defined above.

W-OTS signs messages of binary length m . They are processed in base w representation. They are of the form $M = (M_1 \dots M_{\ell_1})$, $M_i \in \{0, \dots, w-1\}$. The checksum $C = \sum_{i=1}^{\ell_1} (w-1 - M_i)$ in base w representation is appended to M . It is of length ℓ_2 . The result is (b_1, \dots, b_ℓ) . The signature of M is

$$\sigma = (\sigma_1, \dots, \sigma_\ell) = (f_{\text{sk}_1}^{b_1}(x), \dots, f_{\text{sk}_\ell}^{b_\ell}(x)).$$

It is verified by constructing $(b_1 \dots, b_\ell)$ and checking

$$(f_{\sigma_1}^{w-1-b_1}(\text{pk}_0), \dots, f_{\sigma_\ell}^{w-1-b_\ell}(\text{pk}_0)) \stackrel{?}{=} (\text{pk}_1, \dots, \text{pk}_\ell).$$

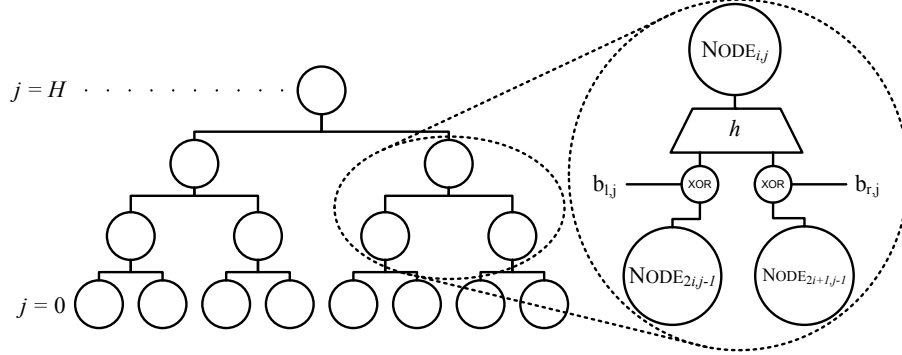
The sizes of signature, public, and secret key are ℓn . For more detailed information see [9].

XMSS Tree The XMSS tree is a modification of the Merkle Hash Tree proposed in [14]. It utilizes the hash function h_K . The XMSS tree is a binary tree. Denote its height by H . It has $H + 1$ levels. The leaves are on level 0. The root is on level H . The nodes on level j , $0 \leq j \leq H$, are denoted by $\text{NODE}_{i,j}$, $0 \leq i < 2^{H-j}$. The construction of the leaves is explained below. Level j , $0 < j \leq H$, is constructed using a bitmask $(b_{l,j} || b_{r,j}) \in \{0, 1\}^{2n}$ chosen uniformly at random. The nodes are computed as

$$\text{NODE}_{i,j} = h_K((\text{NODE}_{2i,j-1} \oplus b_{l,j}) || (\text{NODE}_{2i+1,j-1} \oplus b_{r,j}))$$

for $0 < j \leq H$. The usage of the bitmasks is the main difference to the other Merkle tree constructions. It is borrowed from [5] and allows to

Fig. 1. The XMSS tree construction



replace the collision resistant hash function family. Figure 1 shows the construction of the XMSS tree.

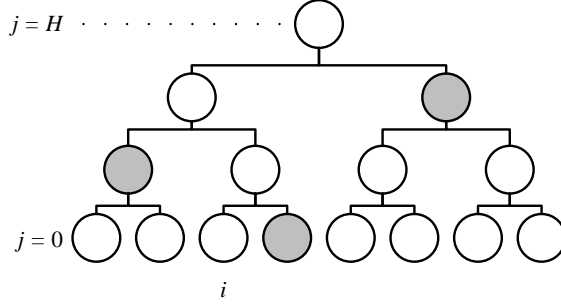
We explain the computation of the leaves of the XMSS tree. The XMSS tree is used to authenticate 2^H W-OTS verification keys, each of which is used to construct one leaf of the XMSS tree. The construction of the keys is explained at the end of this section. In the construction of a leaf another XMSS tree is used. It is called L-tree. The first ℓ leaves of an L-tree are the ℓ bit strings (pk_0, \dots, pk_ℓ) from the corresponding verification key. As ℓ might not be a power of 2 there are not sufficiently many leaves. Therefore the construction is modified. A node that has no right sibling is lifted to a higher level of the L-tree until it becomes the right sibling of another node. In this construction, the same hash function as above but new bitmasks are used. The bitmasks are the same for each of those trees. As L-trees have height $\lceil \log \ell \rceil$, additional $\lceil \log \ell \rceil$ bitmasks are required.

The XMSS public key PK contains the bitmasks and the root of the XMSS tree.

To sign the i th message, the i th W-OTS key pair is used. The signature $SIG = (i, \sigma, AUTH)$ contains the index i , the W-OTS signature σ , and the authentication path for the leaf $NODE_{0,i}$. It is the sequence $AUTH = (AUTH_0, \dots, AUTH_{H-1})$ of the siblings of all nodes on the path from $NODE_{0,i}$ to the root. Figure 2 shows the authentication path for leaf i . To compute the authentication path we use the tree traversal algorithm from [11] as it allows for optimal balanced runtimes using very little memory.

To verify the signature $SIG = (i, \sigma, AUTH)$, the string (b_0, \dots, b_ℓ) is computed as described in the W-OTS signature generation. Then the i th

Fig. 2. The authentication path for leaf i



verification key is computed using the formula

$$(\mathbf{pk}_1, \dots, \mathbf{pk}_\ell) = (f_{\sigma_1}^{w-1-b_1}(x), \dots, f_{\sigma_\ell}^{w-1-b_\ell}(x)).$$

The corresponding leaf $\text{NODE}_{0,i}$ of the XMSS tree is constructed using the L-tree. This leaf and the authentication path are used to compute the path (p_0, \dots, p_H) to the root of the XMSS tree, where $p_0 = \text{NODE}_{0,i}$ and

$$p_j = \begin{cases} h_K((p_{j-1} \oplus b_{l,j}) || (\text{AUTH}_{j-1} \oplus b_{r,j})), & \text{if } \lfloor i/2^j \rfloor \equiv 0 \pmod{2} \\ h_K((\text{AUTH}_{j-1} \oplus b_{l,j}) || (p_{j-1} \oplus b_{r,j})), & \text{if } \lfloor i/2^j \rfloor \equiv 1 \pmod{2} \end{cases}$$

for $0 \leq j \leq H$. If p_H is equal to the root of the XMSS tree in the public key, the signature is accepted. Otherwise, it is rejected.

Signature key generation The W-OTS secret signature keys are computed using a seed $\text{SEED} \in \{0, 1\}^n$, the pseudorandom function family $F(n)$, and the pseudorandom generator GEN which for $\lambda, \mu \in \{0, 1\}^n$ yields

$$\text{GEN}_\lambda(\mu) = f_\mu(1) || \dots || f_\mu(\lambda).$$

For $i \in \{1, \dots, 2^H\}$ the i -th W-OTS signature key is

$$\text{sk}_i \leftarrow \text{GEN}_\ell(f_{\text{SEED}}(i)).$$

The XMSS secret key contains SEED and the index of the last signature i .

The bit length of the XMSS public key is $(2(H + \lceil \log \ell \rceil) + 1)n$, an XMSS signature has length $(\ell + H)n$, and the length of the XMSS secret signature key is $< 2n$.

3 Security

In this section we discuss the security of XMSS. We use the notations of Section 2 and sketch the proof of the following theorem.

Theorem 1. *If $\mathcal{H}(n)$ is a second preimage resistant hash function family and $F(n)$ a pseudorandom function family, then XMSS is existentially unforgeable under chosen message attacks.*

Before sketching the proof of Theorem 1 we show that it implies the minimality of the security requirements of XMSS. For this purpose, we show how to construct second preimage resistant hash function families and pseudorandom function families from secure signature schemes. Those constructions imply that there is a secure instance of XMSS if there is a secure signature scheme. This implies that the security requirements for XMSS are minimal. Now we present the constructions.

In [27] it is shown that a one-way function can be constructed from any secure signature scheme. Also the construction of a target collision resistant hash function family from a one-way function is presented. Since target collision resistant hash function families are second preimage resistant (see [26]), this implies that second preimage resistant hash function families can be constructed from secure digital signature schemes. In [19] the construction of a pseudorandom generator from a one-way function is presented. In [17] pseudorandom function families are obtained from pseudorandom generators. It follows that secure signature schemes yield pseudorandom function families.

Next, we sketch the proof of Theorem 1. First the notion of existentially unforgeability under chosen message attacks (EU-CMA) [18] has to be adapted to the given setting. XMSS is a signature scheme where the private signature key is not constant as in RSA, but changes over time, whereas the definition of EU-CMA assumes a constant signature key. The definition of EU-CMA security can be described by an experiment, where the adversary has access to a signature oracle, initialized with the constant signature key. The adversary is allowed to send messages to the oracle. The oracle returns the corresponding signatures under the secret signature key. At the end of the experiment, the adversary has to come up with a signature for a message she did not send to the oracle before. For one-time signature schemes like W-OTS the number of queries is limited to one. In the XMSS-experiment the adversary is given access to an XMSS oracle which changes the secret signature key after each signature according to the XMSS construction. The adversary may query this oracle 2^H times.

The proof of Theorem 1 proceeds in two steps. First, it is shown that W-OTS, which is known to be EU-CMA-secure from [9] remains EU-CMA secure in the XMSS construction, where the secret key is generated using a pseudorandom function family.

Second, we modify the security proof in [14]. This is necessary since in contrast to the signature scheme in [14] XMSS generates its signature keys pseudorandomly.

In fact, both steps use the following result. Denote by DSS one of the signature schemes XMSS and W-OTS. In both schemes the key generation algorithm uses n bits of random input and expands them to the λn bits using the pseudorandom function family $F(n)$. Denote by DSS^* the modification of DSS, that is obtained by choosing the λn bits randomly. Denote by $\text{InSec}^{\text{EU-CMA}}(\text{DSS}; t, q)$ the maximum success probability over all adversaries running in time $\leq t$ and making at most q_{Sign} oracle queries in the EU-CMA experiment. $\text{InSec}^{\text{EU-CMA}}(\text{DSS}^*; t, q)$ is defined analogously. Also, define by $\text{InSec}^{\text{PRF}}(F(n); t, q)$ the maximum success probability over all adversaries in distinguishing a random element from $F(n)$ from a random function, when the runtime is bounded by t and she can query an oracle for up to q function values. Then the following is true

$$\begin{aligned} \text{InSec}^{\text{EU-CMA}}(\text{DSS}; t, q) &= \text{InSec}^{\text{PRF}}(F(n); (t' + \lambda), \lambda) \\ &\quad + \text{InSec}^{\text{EU-CMA}}(\text{DSS}^*; t, q) \end{aligned}$$

$$t' = t + t_{\text{Kg}} + qt_{\text{Sign}} + t_{\text{Vf}}.$$

This is shown by contradiction. Assume there exists an adversary A returning a valid forgery for DSS with success probability

$$\text{Succ}^{\text{EU-CMA}}(\text{DSS}; A) > \text{InSec}^{\text{EU-CMA}}(\text{DSS}; t, q)$$

in time t . We build a distinguisher Dis for $F(n)$ that runs A and outputs 1, if A returns a valid forgery. Then we show that either Dis can distinguish $F(n)$ with success probability

$$\text{Succ}^{\text{PRF}}(F(n); \text{Dis}) > \text{InSec}^{\text{PRF}}(F; t', q)$$

or A is a forger for DSS^* with success probability

$$\text{Succ}^{\text{EU-CMA}}(\text{DSS}^*; A) > \text{InSec}^{\text{EU-CMA}}(\text{DSS}^*; t, q).$$

Both lead to a contradiction of the initial assumptions.

Applying this result to W-OTS shows that W-OTS as used in the XMSS construction is EU-CMA-secure if $F(n)$ is pseudorandom. Together with the assumption that $\mathcal{H}(n)$ is second preimage resistant it

follows from [14] that XMSS is EU-CMA-secure if the seeds for the W-OTS signature keys are chosen at random. Finally, applying the above result again, we obtain the following formula which is an exact version of Theorem 1 and therefore concludes the proof.

Denote by $\text{InSec}^{\text{SPR}}(\mathcal{H}(n); t)$ the maximum success probability over all adversaries running in time $\leq t$ for finding a second preimage in $\mathcal{H}(n)$. Furthermore, denote the number of key collisions of $F(n)$ by κ according to [9]. Then the maximum success probability over all adversaries running in time $\leq t$, making at most 2^H oracle queries, against the XMSS EU-CMA security is bounded by

$$\begin{aligned} & \text{InSec}^{\text{EU-CMA}}(\text{XMSS}; t, q = 2^H) \\ & \leq \text{InSec}^{\text{PRF}}(F(n); (t' + 2^H), q = 2^H) \\ & \quad + 2 \cdot \max \left\{ \begin{array}{l} (2^{H+\log \ell} - 1) \cdot \text{InSec}^{\text{SPR}}(\mathcal{H}(n); t'), \\ 2^H \left(\text{InSec}^{\text{PRF}}(F(n); (t' + \ell), q = \ell) \right. \\ \left. + (\ell^2 w^2 \kappa^{w-1} \frac{1}{(\frac{1}{\kappa} - \frac{1}{2^n})}) \cdot \text{InSec}^{\text{PRF}}(F(n); (t'), q = 2) \right) \end{array} \right\} \end{aligned}$$

where $t' = t + 2^H \cdot t_{\text{Sign}} + t_{\text{Vf}} + t_{\text{Kg}}$.

Note that, assuming only generic attacks on $\mathcal{H}(n)$ and $F(n)$ the symmetric bit security of XMSS is

$$\begin{aligned} b &= \log \left(\frac{t}{\text{InSec}^{\text{EU-CMA}}(\text{XMSS}; t, q = 2^H)} \right) \\ &\leq \min \{n - 1, n - H - 2 - w - 2 \log(\ell w)\} - 1 \end{aligned}$$

4 Forward Security

Given the above result we can go even further. In [1] Anderson introduced the idea of forward security for signature schemes (FSSIG) which was later formalized in [4]. It says that even after a key compromise all signatures created before remain valid. Obviously, this notion is only meaningful for signature schemes that change their secret signature key over time. From an attacker based point of view this translates to: If an attacker learns the actual secret key sk_i , she is still not able to forge a signature under a secret key sk_j , $j < i$. This is a desirable property, especially in the context of long term secure signatures, as it allows to remove the need for timestamps and an online trusted third party.

In this section we show that XMSS is forward secure if we slightly modify the key generation process based on an idea from [22]. We describe

the modifications. To make XMSS forward secure we use a forward secure PRG FsGen when generating the seeds for the W-OTS secret keys. A forward secure PRG is a stateful PRG that starts from a random initial state. Given a state, it outputs a new state and some output bits. Even if an adversary manages to learn the secret state of a forward secure PRG, she is not able to distinguish the former outputs from random bit strings. In the modified XMSS, the W-OTS seeds are generated by FsGen . Starting from a random input $\text{SEED} = \text{STATE}_0$ of length n , FsGen uses $F(n)$ and the previous state STATE_{i-1} to generate n bits of pseudorandom output OUT_i and a new state STATE_i of length n :

$$(\text{STATE}_i || \text{OUT}_i) = (f_{\text{STATE}_{i-1}}(0) || f_{\text{STATE}_{i-1}}(1))$$

The generation of the W-OTS secret keys from the seeds still utilizes GEN_ℓ . The secret key of the resulting forward secure XMSS contains the actual state STATE_i instead of SEED . In contrast to the construction from Section 2, the seeds for the W-OTS signature keys are not easily accessible from STATE_i using one evaluation of $F(n)$. To compute the authentication path, the tree traversal algorithm needs to compute several W-OTS keys before they are needed. This is very expensive using FsGen . This problem is already addressed in [11]. We use their solution that requires to store $2H$ states of FsGen . This results in a secret signature key size of $2Hn$.

For this modified XMSS we proof the following security theorem.

Theorem 2. *If $\mathcal{H}(n)$ is a second preimage resistant hash function family and $F(n)$ a pseudorandom function family, then XMSS with a modified key generation described below is a forward secure digital signature scheme.*

Again we will only sketch the proof and refer the reader to the full version for more details and the formal definitions. We keep the notions from the last two Sections.

We sketch the proof. Our proof is a modification of the proof from [22]. In the proof of Theorem 1 it is shown that XMSS is EU-CMA-secure if the seeds for the W-OTS signature keys are chosen at random. In [6] it is shown that if $F(n)$ is a pseudorandom function family, then $\text{InSec}^{\text{FSPRG}}(\text{FsGen}; t)$, the maximum success probability of any adversary running in time $\leq t$, attacking the forward security of FsGen is:

$$\text{InSec}^{\text{FSPRG}}(\text{FsGen}; t) = 2\tilde{n} \cdot \text{InSec}^{\text{PRF}}(F(n); (t + 2\tilde{n}), 2)$$

where \tilde{n} denotes the maximum number of outputs produced by FsGen . We use these results in the proof from [22] to conclude the following formula which is an exact version of Theorem 2.

The maximum success probability over all adversaries running in time $\leq t$, making at most 2^H oracle queries, in attacking the forward security of the modified XMSS, $\text{InSec}^{\text{FSSIG}}(XMSS; t, q = 2^H)$, is bounded by

$$\begin{aligned} & \text{InSec}^{\text{FSSIG}}(XMSS; t, q = 2^H) \\ & \leq 2^{2H+1} \cdot \text{InSec}^{\text{PRF}}(F(n); (t' + 2), q = 2) \\ & \quad + 2 \cdot \max \left\{ \begin{array}{l} (2^{H+\log \ell} - 1) \cdot \text{InSec}^{\text{SPR}}(\mathcal{H}(n); t'), \\ 2^H \left(\text{InSec}^{\text{PRF}}(F(n); (t' + \ell), q = \ell) \right. \\ \left. + (\ell^2 w^2 \kappa^{w-1} \frac{1}{(\frac{1}{\kappa} - \frac{1}{2^n})}) \cdot \text{InSec}^{\text{PRF}}(F(n); (t'), q = 2) \right) \end{array} \right\} \end{aligned}$$

$$t' = t + 2^H \cdot t_{\text{Sign}} + t_{\text{Vf}} + t_{\text{Kg}}.$$

5 Efficiency

In this Section we discuss the efficiency of XMSS. We will show that XMSS and its the forward secure variant are efficient if $\mathcal{H}(n)$ is an efficient second preimage resistant hash function family and $F(n)$ an efficient pseudorandom function family. Efficiency here refers to the runtimes and space requirements for sufficiently secure parameters. It is expressed as a function of the security parameter n . In the Section 6 we will propose parameters that are secure according to [23] and present experimental results that support the efficiency of XMSS.

The runtime of all three algorithms of XMSS is dominated by the number $\#call_F$ of calls to $F(n)$ and the number $\#call_{\mathcal{H}}$ of calls to $\mathcal{H}(n)$. We ignore the negligible computational overhead for adding the bitmasks, control flow and computing the base w representation of the message. Using a simple counting argument we obtain the following result:

For one call to the XMSS signature algorithm, the number of calls to $\mathcal{H}(n)$ and $F(n)$ is bounded by

$$\#call_{\mathcal{H}} \leq \frac{H+2}{2} * (H + \ell), \quad \#call_F \leq \frac{H+2}{2} * (\ell(w+1)) + 4H.$$

For one call to the XMSS signature verification algorithm, the number of calls to $\mathcal{H}(n)$ and $F(n)$ is bounded by

$$\#call_{\mathcal{H}} \leq H + \ell, \quad \#call_F \leq \ell w.$$

For one call to the XMSS key generation algorithm, the number of calls to $\mathcal{H}(n)$ and $F(n)$ is bounded by

$$\#call_{\mathcal{H}} \leq 2^H(\ell + 1), \quad \#call_F \leq 2^H(2 + \ell(w + 1)).$$

The space requirements for the internal state of `Sign` and `Kg` (including `sk`) are at most $6H * n$ bits. `Vf` needs no internal state. Hence, the space used by XMSS is at most $6H * n$ bits.

6 Implementation

We have implemented XMSS to evaluate its practical performance. The implementation was done in C, using the AES and SHA-2 implementation of OpenSSL¹. The implementation is straightforward, except for the construction of $\mathcal{H}(n)$ and $F(n)$ for which we implemented constructions based on hash functions and block ciphers.

First we discuss the hash function based constructions. In our implementation any hash function from the OpenSSL library can be used that uses the Merkle-Darmgard (M-D) construction [25]. The family $F(n)$ is constructed as follows.

Given a hash function `Hash` with block length b and output size n that uses the M-D construction we build the function family $F(n)$ as

$$f_K(M) = \text{Hash}(\text{Pad}(K) || \text{Pad}(M)),$$

for key $K \in \{0, 1\}^n$, message $M \in \{0, 1\}^n$ and $\text{Pad}(x) = (x || 10^{b-|x|-1})$ for $|x| < b$.

We show that this is a pseudorandom function family if `Hash` is a good cryptographic hash function. In [2] it is assumed, that the compression function of a good M-D hash function is a pseudorandom function family if keyed using the input. In [3], it is assumed, that the compression function of a good M-D hash function is a pseudorandom function family if keyed on the chaining input. Further it is shown, that a fixed input length M-D hash function, keyed using the initialization vector (IV) is a pseudorandom function family for fixed length inputs. In our construction the internal compression function of `hash` is evaluated twice: First on the IV and the padded key, second on the resulting chaining value and the padded message. Due to the pseudorandomness of the compression function when keyed on the message input, the first evaluation works as a pseudorandom key generation. As we have a fixed message length the second iteration is a pseudorandom function family keyed using the IV input.

For $\mathcal{H}(n)$ we use `Hash` without modifications, as we only need a randomly chosen element of $\mathcal{H}(n)$ and not the whole family. We follow the

¹ <http://www.openssl.org/>

standard assumption for the security of keyless hash functions. It assumes that a keyless hash function is an element of a family of hash functions, chosen uniformly at random.

Next we present the constructions using a block cipher $E(K, M)$ with block and key length n bit. This is of special interest in case of AES, because many smartcard crypto co-processors and also most actual Intel processors provide hardware acceleration for AES. For $F(n)$ we use E without modification, as a standard assumption states that a good block cipher can be modelled as pseudorandom permutation. $\mathcal{H}(n)$ is constructed as $h_K(M) = C_2$ for $M = M_1 || M_2$, with

$$C_i = E_{C_{i-1}}(M_i) \oplus M_i, \quad C_0 = K, \quad 0 \leq i \leq 2$$

in M-D mode. In [7] the authors give a black box proof for the security of this construction. We do not use M-D strengthening, as our domain has fixed size.

Table 1 shows our results on an Intel(R) Core(TM) i5 CPU M540 @ 2.53GHz with Infineon AES-NI² for XMSS. For the forward secure construction the signature key size grows to 10.240 bits (5.120 bits) for SHA-256 (AES-128), respectively. We used a tree height $H = 20$. This leads to instances usable for about one million signatures. Further we assumed a message length of $m = 256$ bit. The last column of the table shows the bit security of the configuration. Following the heuristic of Lenstra and Verheul [23] the AES configuration with bit security 82 is secure until 2015. The SHA-256 configurations with bit security 100 (146, 196, 210) are secure until 2039 (2099, 2164, 2182). According to [23], RSA as well as DSA using a 2048-bit key are assumed to be secure until 2022. The timings for RSA and DSA were taken using the OpenSSL `speed` command. As this does not provide timings for key generation, we had to leave this field blank. The results show that XMSS is comparable to existing signature schemes. Only the key generation takes a lot of time. But as key generation is an offline task, it can be scheduled.

The last row of table 1 shows the signature size and public key size for MSS-SPR [14]. To make the results from [14] comparable, we computed the signature and public key size for message length $m = 256$ bit, using their formulas. [14] does not provide runtimes, therefore we had to leave these fields blank. Comparing XMSS using SHA-256 and $w = 108$ with MSS-SPR shows that even for a slightly higher bit security we achieve a signature length of less than 25 % of the signature length of MSS-SPR.

² <http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni>

Table 1. XMSS performance for $H = 20$, $m = 256$. b denotes the bit security. * Using AES-NI. ** Although the authors of [14] mention the possibility to generate the secret key using a pseudorandom generator, this is not covered by their security proof. For the provided values a secret key of size $2^H \cdot n$ is assumed. A secret key size of 152 bits is possible, slightly reducing the bit security. Hence we exclude this value from the comparison for fairness.

Function	w	Timings (ms)			Sizes (bit)			b
		Sign	Verify	Keygen	Signature	Public key	Secret key	
AES-128*	4	1.72	0.11	109,610.45	19,608	7,296	152	82
AES-128	4	2.87	0.22	158,208.49	19,608	7,296	152	82
SHA-256	4	6.30	0.51	408,687.43	39,192	13,568	280	210
SHA-256	16	7.00	0.52	466,236.55	22,296	13,568	280	196
SHA-256	64	15.17	1.02	1,099,377.18	16,664	13,568	280	146
SHA-256	108	33.47	2.34	2,288,355.24	15,384	13,568	280	100
RSA 2048		3.08	0.09	-	≤ 2048	≤ 4096	≤ 4096	87
DSA 2048		0.89	1.06	-	≤ 2048	≤ 4096	≤ 4096	87
MSS-SPR (n=128)					68,096	7680	-**	98

We also tried to compare XMSS with GMSS [10], but as the authors do not provide a security proof, a fair comparison is not possible without presenting a security proof for GMSS.

References

1. Ross Anderson. Two remarks on public key cryptography. In *Manuscript. Relevant material presented by the author in an invited lecture at the 4th ACM Conference on Computer and Communications Security, CCS*, pages 1–4. Citeseer, 1997.
2. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg, 1996.
3. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *Proceedings of 37th Annual Symposium on Foundations of Computer Science*, pages 514–523. IEEE, 1996.
4. Mihir Bellare and Sara Miner. A forward-secure digital signature scheme. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 786–786. Springer Berlin / Heidelberg, 1999.
5. Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In Burton Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484. Springer Berlin / Heidelberg, 1997. 10.1007/BFb0052256.
6. Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *Topics in Cryptology — CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin / Heidelberg, 2003.

7. John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 103–118. Springer Berlin / Heidelberg, 2002.
8. Daniel Bleichenbacher and Ueli M. Maurer. Optimal tree-based one-time digital signature schemes. In *STACS '96: Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, pages 363–374, London, UK, 1996. Springer-Verlag.
9. Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hülsing, and Markus Rückert. On the security of the Winternitz one-time signature scheme. In A. Nitaj and D. Pointcheval, editors, *Africacrypt 2011*, volume 6737 of *Lecture Notes in Computer Science*, pages 363–378. Springer Berlin / Heidelberg, 2011.
10. Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume. Merkle signatures with virtually unlimited signature capacity. In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security*, volume 4521 of *Lecture Notes in Computer Science*, pages 31–45. Springer Berlin / Heidelberg, 2007.
11. Johannes Buchmann, Erik Dahmen, and Michael Schneider. Merkle tree traversal revisited. In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 63–78. Springer Berlin / Heidelberg, 2008.
12. Johannes Buchmann, Erik Dahmen, and Michael Szydło. Hash-based digital signature schemes. In Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, editors, *Post-Quantum Cryptography*, pages 35–93. Springer Berlin Heidelberg, 2009.
13. Johannes Buchmann, L. C. Coronado García, Erik Dahmen, Martin Döring, and Elena Klintsevich. CMSS - an improved Merkle signature scheme. In *INDOCRYPT*, volume 4329 of *Lecture Notes in Computer Science*, pages 349–363. Springer, 2006.
14. Erik Dahmen, Katsuyuki Okeya, Tsuyoshi Takagi, and Camille Vuillaume. Digital signatures out of second-preimage resistant hash functions. In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography*, volume 5299 of *Lecture Notes in Computer Science*, pages 109–123. Springer Berlin / Heidelberg, 2008.
15. Chris Dods, Nigel Smart, and Martijn Stam. Hash based digital signature schemes. In *Cryptography and Coding*, pages 96–115. Springer Verlag LNCS 3796, November 2005.
16. L. C. Coronado García. On the security and the efficiency of the Merkle signature scheme. Technical Report Report 2005/192, Cryptology ePrint Archive - Report 2005/192, 2005. Available at <http://eprint.iacr.org/2005/192/>.
17. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.
18. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
19. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28:1364–1396, March 1999.
20. Alejandro Hevia and Daniele Micciancio. The provable security of graph-based one-time signatures and extensions to algebraic signature schemes. In Yuliang Zheng, editor, *Advances in Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 191–196. Springer Berlin / Heidelberg, 2002.

21. Markus Jakobsson, Tom Leighton, Silvio Micali, and Michael Szydlo. Fractal Merkle tree representation and traversal. In Marc Joye, editor, *Topics in Cryptology — CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 314–326. Springer Berlin / Heidelberg, 2003.
22. Hugo Krawczyk. Simple forward-secure signatures from any signature scheme. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 108–115, New York, NY, USA, 2000. ACM.
23. Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14:255–293, 2001.
24. Ralph Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO' 89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer Berlin / Heidelberg, 1990.
25. Ralph Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer Berlin / Heidelberg, 1990.
26. Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *FSE*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.
27. John 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*, pages 387–394, New York, NY, USA, 1990. ACM Press.
28. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1994)*, pages 124–134. IEEE Computer Society Press, 1994.
29. Michael Szydlo. Merkle tree traversal in log space and time. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 541–554. Springer Berlin / Heidelberg, 2004.