One-Time Signatures Revisited: Practical Fast Signatures Using Fractal Merkle Tree Traversal

Dalit Naor* Amir Shenhav† Avishai Wool‡

***Abstract*— One-time signatures have been known for more than two decades, and have been studied mainly due to their theoretical value. Recent works motivated us to examine the practical use of one-time signatures in high-performance applications. In this paper we describe FMTseq — a signature scheme that merges recent improvements in hash tree traversal into Merkle's one-time signature scheme. Implementation results show that the scheme provides a signature speed of up to 35 times faster than a 2048-bit RSA signature scheme, for about one million signatures, and a signature size of only a few kilobytes. We provide an analysis of practical parameter selection for the scheme, and improvements that can be applied in more specific scenarios.**

I. Introduction

A. Motivation

Commonly used digital signature algorithms, like RSA, are not sufficiently fast for many applications. As an example application, consider a financial news service that wishes to sign all its messages: such a service requires both low latency and high bandwidth. An efficient alternative to a digital signature is a Message Authentication Code (MAC). However, a MAC does not provide the asymmetry between the signer and the verifier which digital signatures provide — anyone who can verify the signature can also sign.

One-time signatures [Mer89], on the other hand, are digital signatures that are based on one-way functions without a trapdoor, thus they are much faster. One-time signatures have been known for more than two decades, but are usually considered to be impractical. This is due to their complex key management problems, and the fact that their signature length is significantly larger.

Over the last few years, there has been some increased interest in one-time signatures, mainly due to the multicast authentication problem (see [Per01]). New signature schemes have been presented, with key management solutions for some limited scenarios. These techniques and other efficient algorithms that improved key management aspects led us to re-examine whether one-time signatures can be made practical for high-performance tasks.

*IBM Haifa Research labs, Tel Aviv, Israel. dalit@il.ibm.com.
†School of Electrical Engineering, Tel Aviv University, Ramat Aviv 69978, Israel. amir.shenhav@gmail.com.
‡School of Electrical Engineering, Tel Aviv University, Ramat Aviv 69978, Israel. Supported in part by an IBM faculty award. yash@acm.org.

B. Related Work

One-time signatures have been known for a relatively long time. They were first presented by Lamport [Lam79] and Rabin [Rab78], but are mostly known in the form presented by Merkle and Winternitz [Mer89]. These signatures are based on one-way functions, as opposed to trapdoor functions that are used in public key signatures like RSA and ElGamal.

Although one-way functions are simpler to implement and are typically more computationally efficient than trapdoor functions, one-time signatures have been considered to be impractical for two main reasons: First, their "one-timed-ness", i.e., key generation is required for each usage, thus implying a complicated key management scheme; Second, the signature size is relatively long in comparison with common public-key signatures and MACs.

Recently, this area was revisited due to the need for fast, low computation, authentication solutions for IP-multicast and sensor networks. Perrig [Per01] presents a new one-time signature scheme that aims for very fast verification at the cost of signature time and key size. Reyzin and Reyzin [RR02] introduce a different scheme that has faster signature and verification times (for a single signature). This scheme is further improved by Pieprzyk et al. [PWX03]. These new schemes suggest the reuse of the same keys more than once. These "several times signatures" have a known compromise probability after a given number of reuses. Nevertheless, even if these schemes can sign several messages with the same keys, with reasonable security, the number of repeated uses is still small for all practical scenarios. Therefore, efficient key generation and management for a large sequence of one-time signatures still remains on open problem.

Merkle [Mer89] addressed the problem of key management, introducing the method of *tree authentication*. In [Mer89], originally published in 1979, the concept of a *hash tree* is presented, which provides efficient key management for a large number of one-time signatures. The recent works of [JLMS03], [Szy04a], [BKN04] improve Merkle's hash-tree method [Mer89] and give the ability to handle large hash trees more efficiently. The focus of [JLMS03] was to obtain space-time trade off, the focus of [Szy04a] was very low space solution, and [BKN04] combined the two. As an example, Jakobsson et al.'s suggested to use their algorithm for one-time signature. A proof of concept code for their idea was implemented in [Col03] with some basic performance results.

Recently, a work of Seys and Preneel [SP05] provided a power consumption estimation of one-time signatures schemes, for low power mobile platforms. Their work estimates the overall power consumption of the signature schemes of Winternitz and Reyzin, with the key management techniques of Jakobsson et al. [JLMS03] and Perrig [Per01].

The work closest to ours was recently suggested independently by Coronado [Cor05]. The work used a different traversal technique by [Szy04a] and focused on forward security of the scheme rather than on achieving fast signatures; therefore, we cannot directly compare Coronado's results with ours.

C. Contributions

Our first contribution is a design of a signature scheme we call FMTseq - Fractal Merkle Tree sequential signatures. FMTseq combines Merkle's one-time signatures with Jakobsson et al.'s algorithm for hash tree traversal. We refine [JLMS03] construction and complete the details for a practical scheme to provide many one-time signatures with the same hash tree. In contrast with [JLMS03] our work follows Merkle's original suggestion for hash tree construction which is more efficient and conceptually natural.

Next, we provide an efficient implementation of a scheme that significantly improves the preliminary implementation [Col03] (see also [Szy04b]). Our experimental results show that FMTseq is one- or two-orders of magnitude faster than RSA, with low signature sizes and signer storage requirements.

We consider applications that wish to obtain fast signature rates while keeping the run-time space and signature size reasonable. We show that when selecting the parameter values for the scheme, rather than using the low space solution of [JLMS03], we can trade a few additional kilobytes of signer run-time storage to obtain faster signature rates.

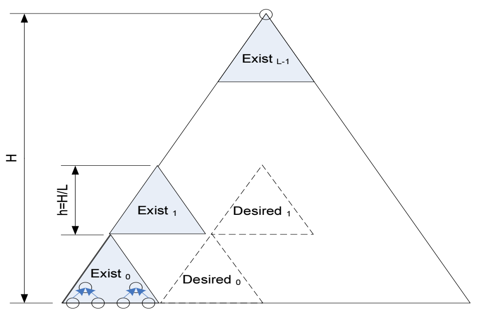


Fig. 1. Fractal merkle tree notations

Full details can be found in [NSW05].

II. Fractal Merkle Tree Sequential One-Time Signatures

In this section we describe our scheme for sequential one-time signatures using fractal Merkle tree algorithm [JLMS03]. The fractal Merkle tree algorithm is a scheme for sequential traversal of a Merkle hash tree, i.e., providing the authentication path for each leaf when the leaves are used one after the other. The scheme requires a computational effort of $2\log N/\log\log N$ and a run-time space of $1.5\log^2 N/\log\log N$ nodes.

Notation: A hash tree T of height H is divided into L *levels*, each of height h. The leaves of the hash tree are indexed $\{0, 1, ..., 2^H - 1\}$ from left to right. The *altitude* of a node n is defined as the height of the maximal subtree for which it is the root and ranges from 0 (for the leaves), to H (for the root). An *h-subtree* is "at level i" when the altitude of its root is ih for some $i \in \{1, 2, ..., L\}$. For each i there are 2^{H-ih} such h-subtrees at level i. A series of h-subtrees $\{Tree_i\}_{i=1}^{L}$ is a *stacked series* if for all $i < L$ the root of $Tree_i$ is a leaf of $Tree_{i+1}$. The notations are illustrated in Figure 1.

In our scheme, the secrets of each one-time signatures are generated by a pseudo-random number generator. The value of each leaf of the fractal Merkle tree is a hash over all the commitments of a single one-time signature. Therefore, *each leaf serves as a public commitment* *to a one-time signature*[1]. For each one-time signature, the signer regenerates the next unused leaf, reveals the required secrets, and outputs the commitments of the unrevealed secrets and the authentication path. We call this scheme FMTseq: Fractal Merkle Tree Sequential One-Time Signature.

The FMTseq scheme consists of three algorithms: key generation, signing and verification. The scheme is described in Figure 2.

The performance in FMTseq is dominated by the leaf calculation time, which is defined as the time to generate all the secrets, commit each one of them with a one-way hash function, and hash all these commitments to one value. Therefore, to minimize the leaf calculation time, we chose the basic Merkle's one-time signature scheme [Mer89].

III. Experimental Results

A. Implementation

We implemented the FMTseq scheme in C and tested its performance on a Pentium IV, 1.7GHz, running Microsoft Windows XP. The implementation of RC4, MD5, SHA-1 and SHA-256 was based on code from [Dev05]. The MD5 code was slightly optimized to achieve a more efficient hash for the committing function[2]. The RC4 stream cipher was used as a pseudo-random number generator. The program ran with real-time priority, measuring time using operating system time functions.

We chose to compare with the popular RSA signature using the Crypto++ code library [Dai04]. Comparing with a wider variety of algorithms is left for a future work, however Crypto++ provides benchmarks [Dai04] that can be used as a reference for a relative compare.

B. Selecting Hash Functions for FMTseq

The hash functions that are used in our scheme determine its performance (both speed and signature size) as well as security. We selected the hash functions so that they provide a level of security that is comparable with RSA signatures. According to [Len01], 2048-bit RSA is approximately equivalent to a 128-bit key of a symmetric cipher, and 1024-bit RSA is approximately equivalent to a 75-bit symmetric key.

[1]In contrast to the suggestion of Jakobsson et al. [JLMS03] that each leaf serves as a commitment to a single secret, i.e., to a single bit.

[2]Since we hash short bit strings we could remove some code that supports arbitrarily long strings.

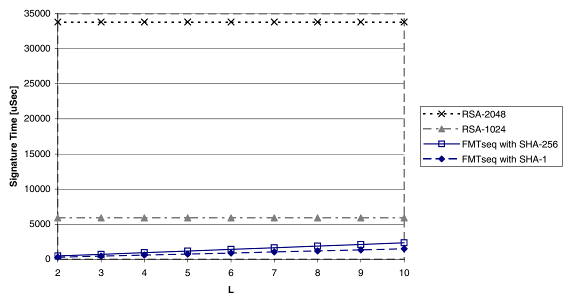


Fig. 3. Average signature time as a function of the number of levels L in the FMTseq tree for $L = 1$ to 10, in comparison with RSA signature times. Each point is an average of 128 runs, each consisting of 128 sequentially generated signatures over randomly chosen messages.

TABLE I
Average Verification Time for $H \leq 20$

Signature Scheme	Average Verification Time
FMTseq with SHA-1	76 uSec
FMTseq with SHA-256	114 uSec
RSA-1024	240 uSec
RSA-2048	600 uSec

For the FMTseq message digest hash function, $g()$, collision resistance is required. Thus, a fair comparison should match SHA-1 to the 1024-bit RSA, and SHA-256 to the 2048-bit RSA. The number of secrets for each one-time signature will be 168 or 264 respectively. For the one-way hash function, $h()$, we select 128-bit MD5 to provide the required security level. For more details the reader may refer to [NSW05].

C. Signature Time

Figure 3 shows the average signature time of FMTseq versus RSA signatures. The figure shows that even for the worst choice of the number of levels, L ($L = 10$), FMTseq-SHA-1 is roughly 4 times faster than 1024-bit RSA, and FMTseq-SHA-256 is more than 14 times faster than 2048-bit RSA. Moreover, other choices of L provide even better performance, e.g., for $L = 4$ we get a 10 times speedup over 1024-bit RSA and more than 35 times speedup over 2048-bit RSA.

Table I shows the average signature verification time for FMTseq and RSA. The FMTseq verification time comprises of a one-time signature and authentication path verification. We found that the verification time is practically invariant to the tree parameters. Therefore, we provided only the average of all the results.

For a practical number of one million signatures (using $L = 4$), comparing FMTseq with 2048-bit RSA

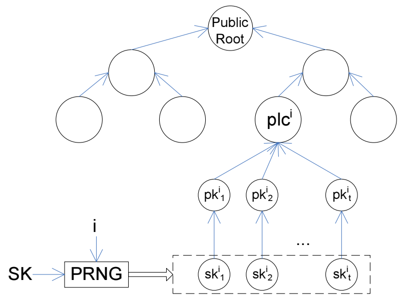
Key generation and initialization

Definitions:

1. SK - a k-bit secret key.
2. sk_j^i - the secrets of each one-time signature where i is the signature number, and j is the index of the secret, $j = 1, \ldots, t$.
3. $sk_j^i = R(SK, i, j)$
   R is a pseudo-random number generator (PRNG).
4. $pk_j^i = h(sk_j^i)$ - the commitments for each sk_j^i.
5. *Public leaf commitment* - $plc^i = h(pk_1^i| \ldots |pk_t^i)$, the hash of all the commitments of a single one-time signature.

Computation:

1. Input: a secret, k-bit key, SK.
2. Initialize a Fractal Merkle hash tree of height H, with the plc^i values as its leaves ($i = 1, \ldots, 2^H$).
3. Publish the tree root
   (or provide it to the verifiers securely).
4. Set Signature number $i = 0$.



FMTseq key generation phase.

Signing a message

Input: a message M.

1. Increment i.
2. Calculate a message digest $md = g(M)$
   and C=number of $'0'$-bits in md.
3. Sign m with Merkle's one-time signature scheme:
   - Let $m = md||C$,
   - Let $J = \{j|m_j = 1\}$ and $\bar{J} = \{j|m_j = 0\}$
   - Generate $\{sk_j^i\}_{j=1}^t$ using the PRNG R.
   - Output $S = \left\{ sk_j^i \; \forall j \in J \, , pk_j^i \; \forall j \in \bar{J} \right\}$.
4. Output $\{ap^i\}$, the authentication path of leaf i, using fractal Merkle tree algorithm [JLMS03].
5. Perform the update operations of [JLMS03].

Verifying a message

Input: a message M and a signature S, $\{ap^i\}$.

1. Calculate a message digest $md = g(M)$.
   and C=number of $'0'$-bits in md.
2. Verify the Merkle one-time signature:
   - Let $m' = md||C$.
   - Let $J = \left\{ j|m'_j = 1 \right\}$.
   - Set $S' = S$, and denote the members of S' by $S' = \{s'_j\}_{j=1}^t$.
   - Calculate and update $s'_j \leftarrow h(s'_j)$, $\forall j \in J$.
   - Calculate $plc' = h(s'_1| \ldots |s'_t)$
3. Iteratively hash the result, plc' with the authentication path.
4. Compare the result to the public root of the tree.

Fig. 2. The FMTseq scheme algorithms and data structure.

(using SHA-256 as a message digest) shows that the verification time is about 5 times faster, and signing time is approximately 35 times faster. Compared with 1024-bit RSA (using SHA-1 as a message digest), FMTseq verification is 3 times faster, and signing is about 10 times faster.

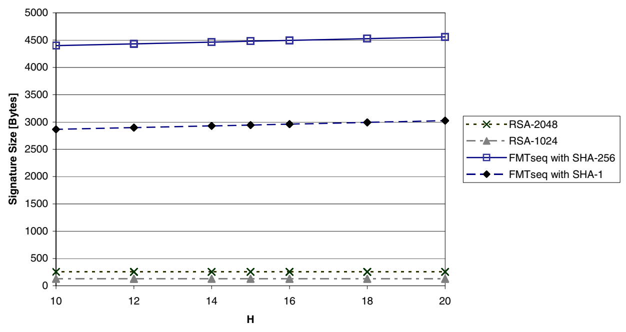


Fig. 4. Signature size as a function of the tree height H in comparison with RSA signatures.

D. Signature Size

The signature size depends on the one-time signature parameters and on the length of the authentication path. The latter only slightly affects the overall size in practical constructions since it is logarithmic with the number of leaves. The one-time signature size is defined by the hash output length as shown in Figure 4.

For about one million signatures (H=20), we compared the signature size of FMTseq to that of RSA. Using SHA-256 as a message digest, the signature size is approximately 18 times larger than 2048-bit RSA, but is still reasonable for many applications: about 4.5 KBytes.

E. A Memory versus Signature Time Trade-off

In [JLMS03] a low-space solution is presented to reduce the memory requirements for running the fractal Merkle tree algorithm. The authors show that minimal space is achieved with subtree height of $h_{opt} \approx \ln H$.

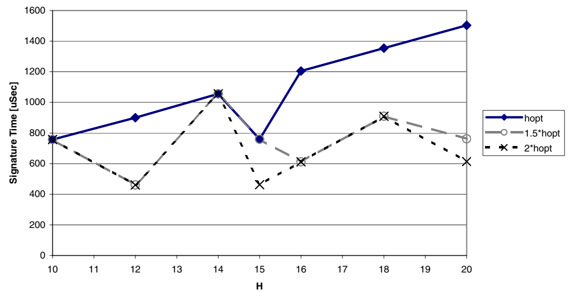


Fig. 5. The improvement in performance that can be achieved, by allowing up to twice the optimal run-time space that was suggested by [JLMS03].

However, h and L must be integers, and we argue that signature time is more important than run-time space in many applications.

Figure 5 demonstrates the improvements in total signature time when allowing a larger space allocation relative to the space that is required when $h = h_{opt}$. For example, when $H = 20$, $h_{opt} = 2$, (since 3 does not divide 20). If we allow up to twice the run-time space compared with h_{opt} (about 4.6 KBytes instead of 3KBytes), the signature time becomes more than twice as fast. The "dip" at $H = 15$ is due to the fact that $h_{opt} \approx 2$, but for $H = 15$, we must use $h = 3$, since h must divide H. For all other choices of H we have $h_{opt} = 2$.

IV. Conclusions

In this work we defined FMTseq - a one-time signature scheme that combines Merkle's one-time signatures with the efficient hash tree traversal of [JLMS03] to a complete detailed signature scheme. Benchmarking FMTseq against RSA signatures, shows that a speedup of up to 35 times can be achieved if signature size of a few kilobytes is acceptable. The FMTseq scheme differs from the [JLMS03] suggestion for a signature scheme, in the method by which the leaves of the hash tree represent the one-time signatures. FMTseq followed Merkle's suggestion [Mer89] for using the hash tree, which is more efficient and conceptually natural.

We demonstrated that when speed is the bottleneck factor, a different selection of the fractal traversal parameters leads to a significant improvement in performance at the cost of a small increase in memory consumption.

We believe that when fast signatures are required, our results makes a strong case for practical feasibility of using one-time signatures as an alternative to the public-key signatures like RSA.

References

[BKN04] Piotr Berman, Marek Karpinski, and Yakov Nekrich. Optimal trade-off for Merkle tree traversal. *Electronic Colloquium on Computational Complexity (ECCC)*, (049), 2004.

[Col03] Dominic F. Coluccio. C++ implementation of a hash-based digital signature scheme using fractal Merkle tree representation. `http://cs1.cs.nyu.edu/~dfc218/hashsig.html`, 2003.

[Cor05] Carlos Coronado. On the security and the efficiency of the merkle signature scheme. Cryptology ePrint Archive, Report 2005/192, 2005. `http://eprint.iacr.org/`.

[Dai04] Wei Dai. Crypto++ library 5.2.1. `http://www.eskimo.com/~weidai/cryptlib.html`, 2004.

[Dev05] Christophe Devine. Crypto source code, GNU public license. http://www.cr0.net:8040, 2001-2005.

[JLMS03] Markus Jakobsson, Frank Thomson Leighton, Silvio Micali, and Michael Szydlo. Fractal Merkle tree representation and traversal. In *Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003*, pages 314–326. Springer, 2003.

[Lam79] Leslie Lamport. Constructing digital signatures from a one-way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, October 1979.

[Len01] Arjen K. Lenstra. Unbelievable security. Matching AES security using public key systems. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, pages 67–86. Springer, 2001.

[Mer89] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89*, pages 218–238. Springer, 1989.

[NSW05] Dalit Naor, Amir Shenhav, and Avishai Wool. One-time signatures revisited: Have they become practical? Cryptology ePrint Archive, Report 2005/442, 2005. `http://eprint.iacr.org/`.

[Per01] Adrian Perrig. The BiBa one-time signature and broadcast authentication protocol. In *ACM Conference on Computer and Communications Security*, pages 28–37, 2001.

[PWX03] Josef Pieprzyk, Huaxiong Wang, and Chaoping Xing. Multiple-time signature schemes against adaptive chosen message attacks. In *Selected Areas in Cryptography, SAC 2003*, pages 88–100. Springer, 2003.

[Rab78] M. O. Rabin. Digitalized signatures. In Richard A. DeMillo, David P. Dobkin, Anita K. Jones, and Richard J. Lipton, editors, *Foundations of Secure Computation*, pages 155–168. Academic Press, 1978.

[RR02] Leonid Reyzin and Natan Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In *Information Security and Privacy, 7th Australian Conference, ACISP 2002*, pages 144–153. Springer, 2002.

[SP05] Stefaan Seys and Bart Preneel. Power consumption evaluation of efficient digital signature schemes for low power devices. In *Proceedings of the 2005 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (IEEE WiMob 2005)*, pages 79–86. IEEE, 2005.

[Szy04a] Michael Szydlo. Merkle tree traversal in log space and time. In *Advances in Cryptology - EUROCRYPT 2004*, pages 541–554. Springer, 2004.

[Szy04b] Michael Szydlo. Recent improvements in the efficient use of Merkle trees: Additional options for the long term. RSA Laboratories: Technical Notes and Reports, 2004.