ON THE EFFICIENCY OF THE LAMPORT SIGNATURE SCHEME

Daniel ZENTAI
Óbuda University, Budapest, Hungary
zentai.daniel@bgk.uni-obuda.hu

ABSTRACT

Post-quantum (or quantum-resistant) cryptography refers to a set of cryptographic algorithms that are thought to remain secure even in the world of quantum computers. These algorithms are usually considered to be inefficient because of their big keys, or their running time. However, if quantum computers became a reality, security professionals will not have any other choice, but to use these algorithms. Lamport signature is a hash based one-time digital signature algorithm that is thought to be quantum-resistant. In this paper we will describe some simulation results related to the efficiency of the Lamport signature.

KEYWORDS: digital signature, post-quantum cryptography, hash functions

1. Introduction

Although reasonable sized quantum computers do not exist yet, post quantum cryptography (Bernstein, 2009) became an important research field recently. Indeed, we have to discover the properties of these algorithms before quantum computers become a reality, namely suppose that we use the current cryptographic algorithms for x more years, we need y years to change the most widely used algorithms and update our standards, and we need z years to build a quantum-computer. In this case we face serious problems if x+y>z.

Lamport signature (1979) is a hash based one-time digital signature algorithm that is thought to be quantum-resistant. In this work we will describe some simulation results related to the efficiency of the Lamport signature. We implemented the Lamport algorithm with four different hash functions: MD5, SHA1, SHA256, and SHA512.

This paper is organized as follows. After this introduction, in chapter 2 we describe some basic concepts and definition related to the security of hash functions. Also, we describe the Lamport one-time signature algorithm. In chapter 3 we expound our simulation results related to the efficiency of the Lamport signature. The last chapter summarizes our work.

2. Preliminaries

In this chapter the basic concepts and definitions will be described, that are needed to understand the following results. First of all we have to define cryptographic hash functions.

Definition. A hash function is a mapping from a set of arbitrary length bitstrings to a set of fixed length bitstrings i.e. a hash function is a function of the following form:

$$h: \{0,1\}^* \rightarrow \{0,1\}^n$$

for some positive integer n.

DOI: 10.2478/raft-2020-0033

Hash functions play a very important role in cryptography. The most important applications are password storing, and digital signatures.

Most of the time, computers do not store our password as a plaintext, instead they store a so called message digest, which is the value of a hash function on the password. Namely, if the password is x, the value h(x) appears in the database, and during the login procedure, upon receiving a password x' from the user, the system calculates h(x'), and if h(x)= h(x'), the user can log in.

On the other hand, if we sign a message x, for efficiency reasons, we will sign h(x) instead. Then the receiver also calculates h(x), and verifies the signature on this hash value.

In both cases we will face some problems, if an attacker can construct an output value y, such that y=h(x), because this way an attacker can log in with our credentials, or forge a fake signature on our message. Therefore the most important requirement of a secure hash function is the one-way property. Informally a one-way hash function is easy to calculate, but hard to invert i.e. an attacker cannot compute an *y* such that *y=h(x)* with non-negligible probability. A more precise definition is as follows (Katz & Lindell, 2007).

Definition. A function $h: \{0,1\}^* \rightarrow \{0,1\}^n$ is one-way, if *h* can be calculated in polynomial time, but for any randomized polynomial time algorithm A, and any input *x* with sufficiently large length, we have

$$\Pr\left[h\left(\mathrm{A}(\mathrm{h}(\mathrm{x}))\right) = h(x)\right] < |x|^{-c}$$

for all *c* positive integer.

2.1. The Lamport Signature

The Lamport signature scheme is an algorithm for constructing one-time digital signatures (Lamport, 1979). The algorithm is not secure if we generate multiple signatures with the same key, but with some modification we can do so (Merkle, 1988). The algorithm works as follows.

Key generation: Let h be a one-way function and suppose that we want to sign a message m with length |m|=k. For all $\mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{k}$ and $\boldsymbol{j} \in \{\mathbf{0}, \mathbf{1}\}$ the signer chooses $\boldsymbol{y_{i,j}}$ from the domain of h, and calculates $\boldsymbol{z_{i,j}} = \boldsymbol{h}(\boldsymbol{y_{i,j}})$. The private key consists of the values $\boldsymbol{y_{i,j}}$ and the public key consists of the values $\boldsymbol{z_{i,j}}$.

Signature: Let m be a k-bit message with digits $\boldsymbol{m_1}, \boldsymbol{m_2}, \ldots, \boldsymbol{m_k}$. The signature of m calculated by the sender is $\boldsymbol{sig}(\boldsymbol{m}) = (\boldsymbol{y_{1,m_1}}, \boldsymbol{y_{2,m_2}}, \cdots, \boldsymbol{y_{k,m_k}})$.

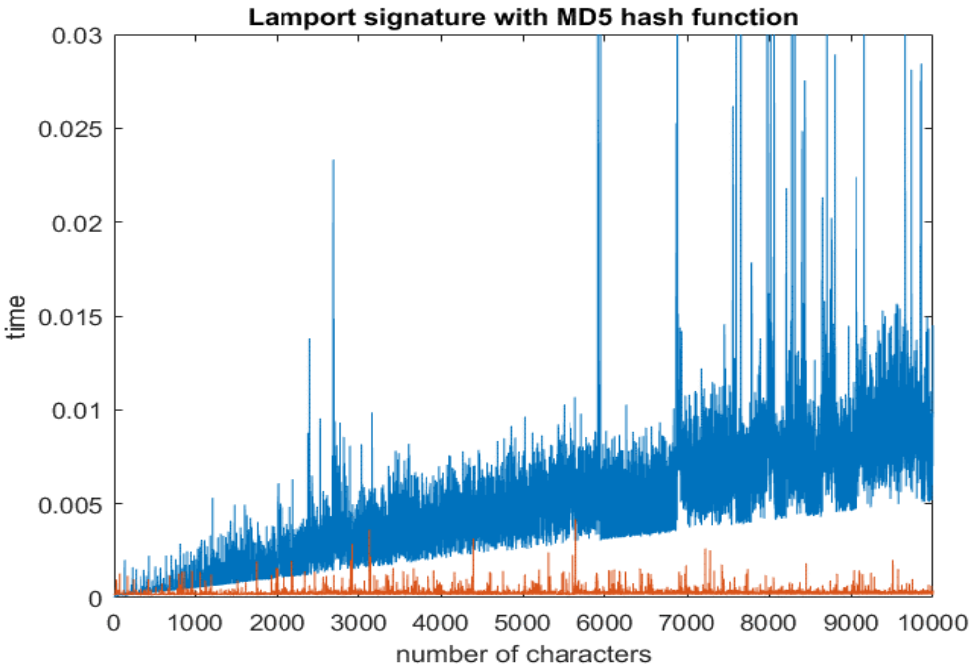
Verification: Upon receiving the message with a signature, the receiver accepts the signature if $\boldsymbol{f}(\boldsymbol{y_{i,m_i}}) = \boldsymbol{z_{i,m_i}}$ for all $\mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{k}$.

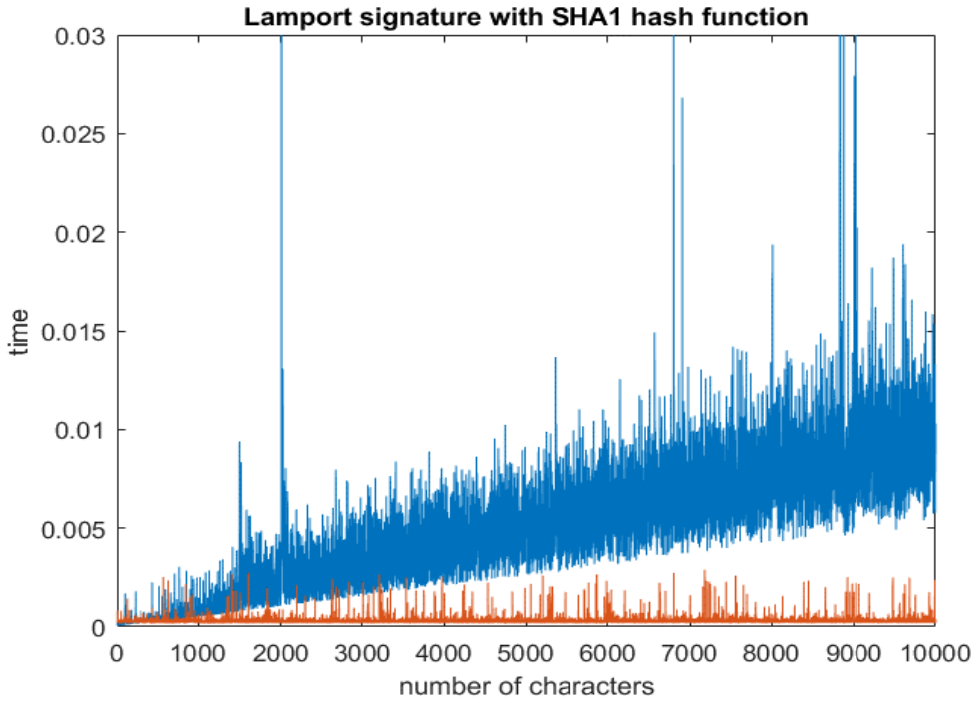
As we already mentioned, the Lamport signature is not the most practical digital signature scheme, because we have to generate a new pair of keys for each messages. However, the algorithm has a serious advantage. Namely, the Lamport signature scheme is secure against quantum computer attacks in contrast with the most popular digital signatures like Rivest, Shamir & Adleman, (1978) and DSS (National Institute of Standards and Technology, 1994). In other words, we do not know any polynomial time algorithms that can forge a fake Lamport signature with non-negligible probability, even if we have a quantum computer.

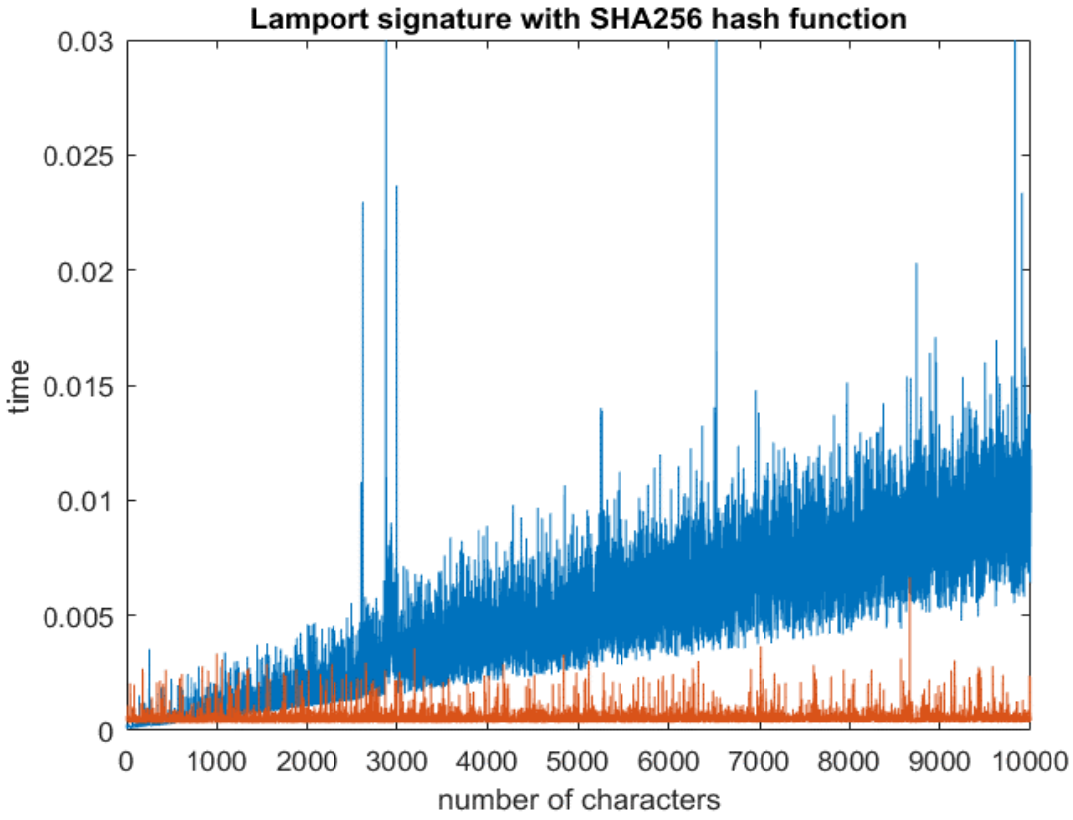
3. Efficiency Results

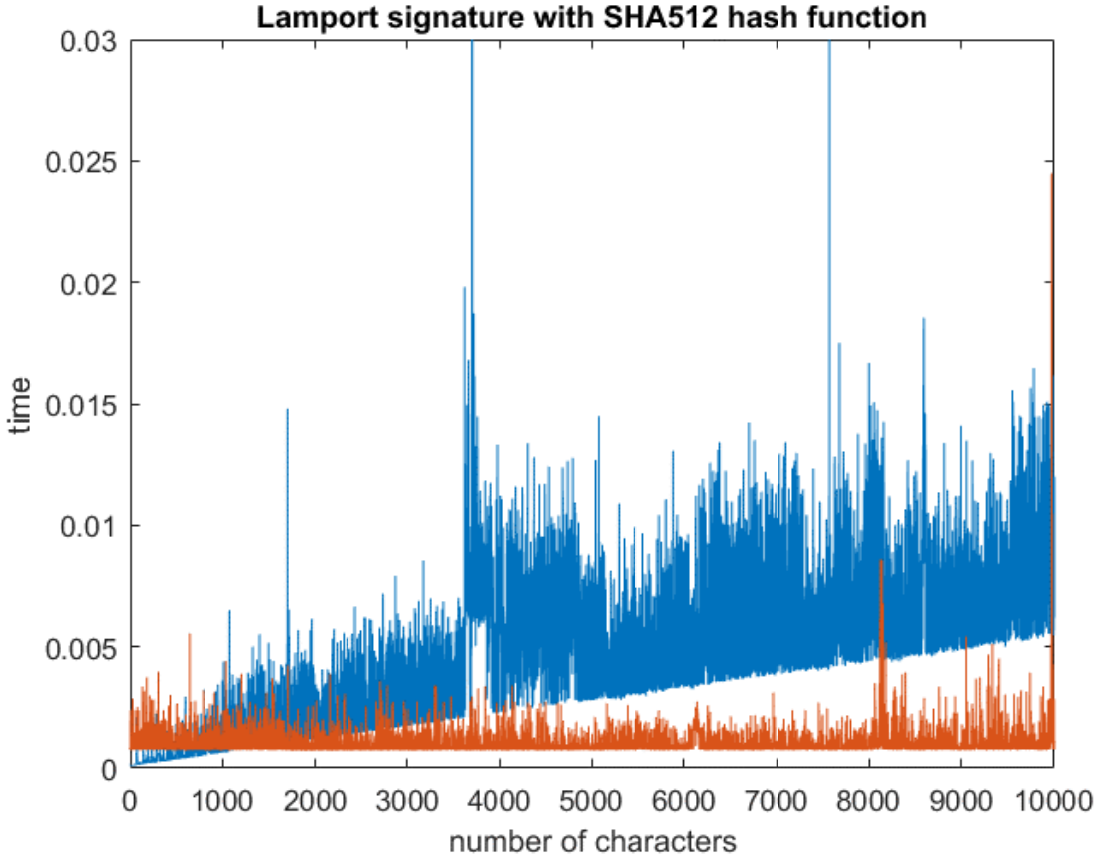
In this chapter we will describe our simulation results. Although originally the Lamport signature scheme can be implemented with any one-way functions, not only one-way hash functions, in practice the most widely used one-way functions are the hash functions, therefore we will compare four different hash functions in our analysis. Our approach was the following. We implemented the Lamport signature scheme in Python, and

we generated random messages of length 1 to 10000, and we measured the running time of the algorithm both in case of the signature generation and verification. These random messages consisted of upper- and lowercase ASCII characters. We also measured the running time using different hash functions such as MD5, SHA1, SHA256 and SHA512. Our results appear on the following diagrams. The blue lines correspond to the signature generation, and the orange lines correspond to the verification.






Lamport signature with SHA256 hash function
time
number of characters
0
0.005
0.01
0.015
0.02
0.025
0.03
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000


Lamport signature with SHA512 hash function
time
number of characters
0
0.005
0.01
0.015
0.02
0.025
0.03
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

We can conclude from these diagrams, that the running time is increasing about linearly on average in the number of characters in the message. Although the Lamport signature scheme is not secure for multiple messages with the same key, for simulation purposes it is sufficient to sign every message with the same key. We generated the keys once for each hash functions, therefore the key generation time is not included in the diagrams.

We summarize the average simulation results in the following table.

Table no. 1

Hash function	*Key generation*	*Signature generation*	*Verification*
MD5	0.000447	0.004755	0.000243
SHA1	0.000712	0.004871	0.000306
SHA256	0.002063	0.004886	0.000536
SHA512	0.003345	0.004934	0.000981

As we can see in the table, the signature generation does not slow down considerably, because the signature generation does not involve any hash calculation, just some value selection from the pre-calculated hash values. In contrast, the key generation and the verification time slows down, as we choose larger hash functions, because these processes involve hash calculations.

4. Conclusion

In this work we focused on the efficiency of the Lamport signature scheme. Based on the simulation results, we can conclude, that there are no significant difference between the running time of the signature generation if we use different hash functions. In contrast, the key generation and the verification time is considerably increasing if we use hash functions with larger output size. As a summary, it is recommended to use SHA256 or SHA512 hash functions for digital signatures as these are more secure hashes, despite the fact, that MD5 and SHA1 are computable faster.

REFERENCES

Bernstein, D.J. (2009). Introduction to post-quantum cryptography. In Bernstein, D. J., Buchmann, J., & Dahmen, E. (Eds), *Post-Quantum Cryptography*. Berlin, Heidelberg: Springer.

Katz, J. & Lindell, Y. (2007). *Introduction to Modern Cryptography*. Chapman & Hall/Crc Cryptography and Network Security Series.

Lamport, L. (1979). *Constructing digital signatures from a one-way function*. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory.

Merkle, R.C. (1988). A Digital Signature Based on a Conventional Encryption Function. Advances in Cryptology – CRYPTO ’87. *Lecture Notes in Computer Science. Vol. 293*, 369-378.

National Institute of Standards and Technology. (1994). Digital Signature Standard (DSS). *Federal Information Processing Standards Publication.*

Rivest, R., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM, Vol. 21, Issue 2*, 120–126.