

## A Novel Approach for Signing Multiple Messages: Hash-Based Signature

Fajge Akshay M.<sup>1</sup> and Hatkar S. S.<sup>2</sup>

<sup>1</sup>*M.Tech. Student, CSE Dept., SGGS IE&T, Nanded, Maharashtra-431606*

<sup>2</sup>*CSE Dept., SGGS IE&T, Nanded, Maharashtra-431606*

### Abstract

In this paper, we study digital signature algorithm (DSA) which are widely used in practice, and their security related issues which might occur in the presence of quantum computer. Security of traditional DSA relies on the factorization problem which is found to be susceptible in quantum ERA. We propose here a new approach for general purpose signature systems for signing multiple messages that is somewhat similar to the Merkle's tree signature (MTS). The advantages of this new approach- it is simple to implement, and can resist the attacks that are even made with the help of quantum computer. We compared overall signature size of the message which is generated using the proposed approach with Merkle's approach. Results show that our algorithm is notable when it comes to space requirement and run-time of algorithm, due to reduced size of the path information and the number of hash operations that are required for verification of the MTS public key respectively, making our approach more computationally efficient. Experimental results are presented to verify proposed approach.

**Keywords:** asymmetric key; digital signature algorithm; hash-based cryptography; merkle tree signature; post-quantum cryptography; quantum computing;

### Introduction

Digital signatures are one of the major inventions of modern cryptography. Cryptographic schemas such as Digital Signature Algorithm, Diffie-Hellman on Elliptic Curves, Elliptic Curve Digital Signature Algorithm (ECDSA), and ElGamal Cryptosystem [1, 2, 3] are vital components of IT-security solutions. The problem is how can Alice signs a message such that Bob can verify the digital signature which is good enough also for legal purposes. Digital signature schemas with this feature have

many advantages, including smaller bandwidth for signatures on small message and direct integration into other schemas such as ElGamal encryption, identity-based public key systems or key agreement protocols. Handwritten signature appeared to be same for different documents, while digital signatures of the messages varies from message to message. At the time of sending necessary message, Signer generates signature using signing algorithm and in response to that Verifier uses verification algorithm for verification of the received message by validating received digital signature.

Signing a message  $m$ , by a Signer  $X$  so that any Verifier can verify the signature as  $S_X(m)$  by using a public-key cryptosystem. Signing a message  $m$  by a user  $X$  so that only Verifier  $Y$  can verify the signature;  $E_Y(S_X(m))$ . Sending a message  $m$  and a signed hash value of  $m$  with the help of hash algorithm as:  $(m, S_X(H(m)))$ . Digital signature should not expose any information about how to generate copies of the signature on the behalf of the Signer. Modified version of the ElGamal digital signature schema was proposed in August 1991 as DSA and adopted in December 1994. Security of the ElGamal signatures resides on the hardness of solving the discrete logarithm problem. Since ElGamal signature is unsecure than discrete logarithm, it is essential to increase the parameter size while signing the message using ElGamal signature algorithm which results in large signature size.

In 1994, Shor [4] developed quantum algorithms which can solve integer factorization problem and discrete logarithmic problem [5] in bounded error quantum polynomial time (BQP) on quantum computer. A quantum computer is the device, which exploits the laws of quantum physics, such as superposition and entanglement, to process information. The idea of a quantum computer was first proposed in 1981 by American theoretical physicist Richard Feynman and Paul Benioff independently. Nobel laureate Richard Feynman pointed out that “Accurately and efficiently simulating quantum mechanical systems would be impossible on a classical computer, but that a new kind of machine, a computer itself built of quantum mechanical elements which obey quantum mechanical laws, might one day perform efficient simulations of quantum systems [6]”.

Post-quantum cryptography has become a topic of research since the several key enhancements have been made in quantum computing, such as the invention of quantum algorithms, and quantum computers in the last few years. Cryptosystem protects data from stealing or modification, and can also be used for verification of the users. There are some specific security requirements such as authentication, integrity, privacy, and non-repudiation in any application-to-application communication. The most complex cryptosystems which are in existence today, are relying on the hardness of certain mathematical problems such as integer factorization problem and discrete logarithm problem. These cryptosystems are not provably secure since the mathematical structure of the problem is not provably hard. However, the security schemas, which are most widely used today, have implementation based on these problems and remains secure over the past years due to the fact that there is no known classical algorithm which can solve these problems in polynomial time. Invention of Shor’s algorithm boosts the development of quantum computer and post-quantum cryptosystems. Quantum computer, which deals with thousands of qubit, can

make current cryptosystems which rely on integer factorization problem and discrete logarithmic problem vulnerable. However, there are few more public-key cryptographic techniques for which no known efficient quantum algorithm exists, and which remain intact against the attacks performed with quantum computers, and systems which implement these cryptographic techniques are nothing but post-quantum cryptosystems.

In recent years, One-Time signature (OTS) systems gaining more attention because of their post-quantum security and their appropriateness for compact implementations. OTS systems which is used to sign single message and general purpose signature systems for signing multiple messages, have been well-known since 1979 [7], and have advanced from renewed development in the last decade. A signature system provides asymmetric message authentication. In the setup stage of the algorithm, it produces a public/private key-pair. The generated message signature can be verified using OTS public key.

The remaining paper is organized into five sections. Second section deliberates work related to OTS systems, Third Section discusses the Merkle's Tree Signature (MTS) to solve problem of large public size while signing multiple messages. Fourth section deals with the novel approach for minimizing the signature size, which helps to improve the space/time trade-offs of general purpose signature systems. In the next section we discussed the results compared with MTS. The paper is concluded in the last section with some observations.

### **Related Work**

Most of the One-Time Signature systems were well considered from 1990 and have advanced from new development over the last decade. Post-quantum signature schemas have not depend on number theoretic problems to ensure security and come with the modularity in selecting hash function, and it is not tied to any specific hash function as that of traditional signature. In 1979, Lamport proposed Lamport-Diffie One-Time Signature schema (LD-OTS) detailed in [8]. In this approach, Signer selects the two random values say, 'X' and 'Y' which serve as the key pair, and publishes  $H(X)$  and  $H(Y)$  as public keys. In the signature generation phase signer has to compute the hash of the message  $m$ , i.e.  $H(m)$  and for each bits of  $H(m)$ , the signer then exposes  $i$ 'th bit  $H(X)$  if  $i$ 'th bit of  $H(m)$  is 0, and  $i$ 'th bit  $H(Y)$  otherwise. It is impossible for adversary to forge such signature without inverting selected one-way function. The large memory requirement of the original one-time signature schema makes schema impractical for general use.

It will work for the one bit message, however, it is insecure to use same 'X' and 'Y' key pair values for different messages, since it was found that, the one-time use of the signature exposes the half of the signing key once it used. The security of this schema depends on function 'H', i.e. selected one way Hash function which states that, it is impossible to generate two different valid messages  $m_1$  and  $m_2$  for a given collision resistant hash function, say  $H$ , such that  $H(m_1) = H(m_2)$ . Some of the hash functions designed to be remains secure even in presence of quantum computer, for example, SWIFFT in [9]. The main disadvantages of the single bit version of LD-OTS

schema are- the size of the signature which found to be relatively large; it is not an efficient to generate the message signature of very large message, since it processes bitwise; and it does not allow to sign multiple messages.

Lamport One-Time Signatures multi-bit version is used to overcome the problem of signing large messages. In this schema, Alice selects random values  $X [X_0, X_1, X_2, X_3, \dots, X_{255}]$  and  $Y [Y_0, Y_1, Y_2, Y_3, \dots, Y_{255}]$  and publishes the set of the public key, such as,  $[H(X_0), H(X_1), H(X_2), H(X_3), \dots, H(X_{255})]$  for  $X$  and  $[H(Y_0), H(Y_1), H(Y_2), H(Y_3), \dots, H(Y_{255})]$  for  $Y$ . Alice can use these public keys to sign arbitrarily long message,  $m$ , efficiently by running many instances in parallel. This multi-bit version just help Alice to sign large messages efficiently, however, it does not overcome the other problems of the single bit version of the LD-OTS.

To overcome the problem while signing multiple messages Alice have to use multiple One-Time signatures. This can be achieved at the cost of some space. In case of the multi-bit version of the LD-OTS for single signature, Alice has to publish public keys of the size  $(256 + 256) * \text{output size of Hash function}$ . If Alice is using a SHA-2 algorithm with its 512 bit variant, then required size of the public and private keys become  $512 * 64$  Bytes, which is nearly equal to 32KB and if Alice required 10,000 signatures to sign multiple messages, then required size become 320MB which is much larger for devices having small memory. The main problem of this approach is the large memory requirement. The problem of the private key size can be solved by using pseudo random number generator by using a single input stream for generating private keys, however, the problem with the size of the public key remains the same. Merkle Tree provides the solution to the problem of large public key size by arranging the OTS at the leaf level of the tree.

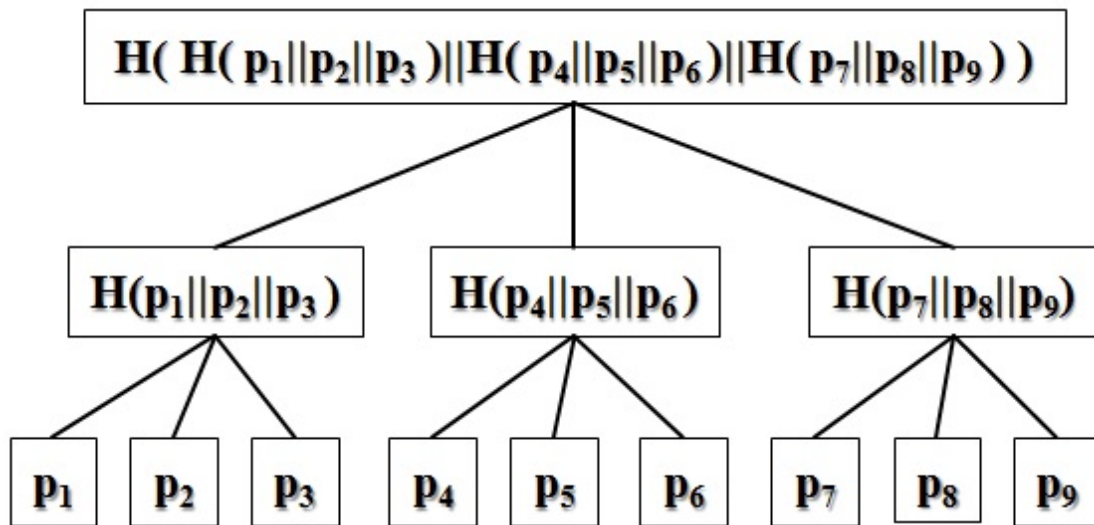
Witernitz OTS (W-OTS) [10] proposed the idea to use iterating hashes and to sign several bits at a time. Instead of selecting two different random values, it selects only one value, say 'X' and instead of second value, it selects  $H(X)$ . Alice needs a checksum to ensure multi-message signing capability, as an attacker can generate signature of remaining bit by using the signature of the given bit. Witernitz's trick found to be more useful while signing larger messages. It is appropriate to combine W-OTS with Merkle's tree authentication schema to generate efficient signature.

### **Merkle Tree Signature**

Merkle in his Ph.D. dissertation [7], proposed a method to sign multiple messages, without the gigantic cost of storing two secret values per bit to be signed. The Merkle Tree signature (MTS) schema contains  $2^d$  possible number of signatures, bound together in a Merkle tree structure of depth  $d$ , which helps to solve key distribution problem of LD-OTS. It solves the problem of signing multiple messages. An MTS system can be used with any OTS system to avail the system with the capability of signing multiple messages, without initializing new keys each time, with the help of collision-resistant hash function. Merkle hash tree provides the solution to the problem of the large public keys. In this schema, OTS are arranged at the leaf level of the tree, while all the intermediate nodes come with the same degree (having the same number of the children) and holds the hashed value which is calculated from all of its

child nodes as shown in Fig 1. The root node of the Merkle Tree holds the MTS public key which serves as the public key for all its OTS, henceforth reducing the size of the public key from 320MB to just 64 bytes. Some one-time key generation cost is involved in this process which makes generation phase some of time consuming.

The problems with this approach is larger signature size as the signature consist of the all intermediate hashes in case of non-binary hash tree; and repeated hashing required to verify MTS public key.



**Fig. 1. Merkle Hash Tree**

In Fig. 1, leaf level of the tree contains 9 OTS public keys from  $p_1$  to  $p_9$ . Each non-leaf node has three children and contains the hash of the concatenation of the values of its children. The root node of the Merkle's tree contains a public key and its size has been reduced to a single hash. Though, each leaf value is considered to be the public one, Verifier only requires the root value. Signer has to supply additional public information to the Verifier such as, signature number, values of all of the siblings from left to right of the nodes, and hashed values of nodes which are not on the path to the root.

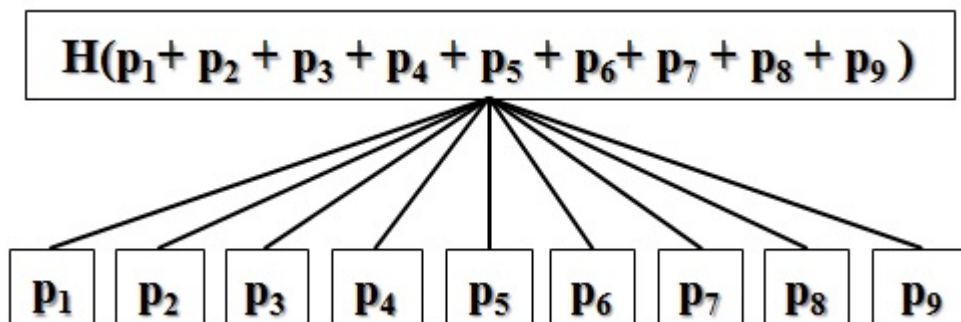
For Example, if Signer is using  $p_1$  OTS as shown in Fig 1, then Verifier can regenerate the OTS public key and using the signature and path information, which contains the path up to the root of the tree - i.e. the values  $p_2$ ,  $p_3$ ,  $H(p_4||p_5||p_6)$  and  $H(p_7||p_8||p_9)$  as in Fig. 1, one can assure Verifier that the OTS public key is part of Signer's overall MTS public key. The process to calculate path information for each OTS is time consuming, however Signer can minimize the time at the cost of huge space, if the tree contains 1 million OTS.

Merkle Tree parameter plays vital role in creating an optimal tree such that cost will be minimized. The capacity of the non-leaf nodes and height of the tree gives the total number of the OTS that can be held in Merkle Tree. For Example, if tree height is 'h', and each non-leaf node has capacity 'c' then given Merkle tree can hold  $c^h$

OTS. Height and number of children of the each non-leaf nodes should be selected such that path information of the OTS for verification of MTS public key, will remain minimized. The number of the siblings of the nodes in a path from OTS to the root of the Merkle tree having height of the tree 'h', and capacity of each non-leaf node is 'c', is found to be  $h(c-1)$ . The Main problem with Merkle's idea is to traverse the tree from leaf node to root node. While signing the message, Signer has to provide path verification for each OTS and for that, Signer has to remind all intermediate hash values up to the root node. In Merkle's approach, algorithm required  $O(h)$  hash evaluations for each OTS verification with  $O(h^2)$  space requirement for storing intermediate hash values. In [11] Markus Jakobsson et al, presented a modified version of Merkle's scheduling algorithm to accomplish a space-time trade-off by increasing speed for signing operation by random factor of 'a' as compared to original Merkle's algorithm, at the cost of  $2^a$  more space. Michael Szydlo exhibits new alternative approach which is found to be space efficient [12], but provides no trade off. This paper proposes a novel schema for generating MTS public key and path information for each OTS so that space and time required for signing multiple messages will dramatically get reduced.

### Proposed Schema

In this section we describe the new approach compared to original Merkle's algorithm and the main difference will be in how the MTS public key is generated and how the verification path of each OTS is calculated. Similar to Merkle's Tree our approach is also capable of signing multiple messages, however it will require less space and time for signing messages, owing to reduced number of sibling information included in the verification path of OTS. With the help of cryptographic components: a one-way function, and one-time signature system, we have successfully verified our proposed idea. Our tree has height,  $h=1$ , having all OTS residing at level 1 of the tree as shown in Fig. 2.



**Fig. 2. Proposed Approach**

In Fig. 2, level first is itself a leaf level of the tree containing 9 OTS public keys from  $p_1$  to  $p_9$ . Root node contains the MTS public key for all one-time signatures public keys. MTS public key is calculated by taking hash of the addition of all the

OTS public keys. Verifier requires the root value and Signer has to supply additional public information to the Verifier which is nothing but the sum of all the OTS public key minus the public key of the current one-time signature. For Example, suppose summation of all the OTS public key is  $Z$ , i.e.  $Z = (p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 + p_9)$ ; then,  $H(Z)$  serves as the MTS public key. We presented below the more efficient and simple algorithm to generate MTS public key.

**Algorithm 1: Generation of  $Z$  and MTS public key**

Input: Required number of OTS\_public\_keys, name\_of\_hash\_algorithm

Output:  $Z$ , mts\_pub\_key

Step I: OTS\_public\_keys=one-time signature generation system('PASSWORD', 'no. of OTS keys required')

Step II:  $Z$ =sum(OTS\_public\_keys)

Step III: mts\_pub\_key=HASH( $Z$ , name\_of\_hash\_algorithm)

In Algorithm I, Step I describe the setup stage of OTS system for the generation of required number of OTS public keys and return the set of all OTS public keys generated depends on the one-time signature system used such as LD-OTS, W-OTS, and LDWM one-time signature system. In Step II, sum of all the OTS public keys is calculated and stored in  $Z$  which is of type big integer. MTS public key is calculated as a hash of  $Z$  using the hash algorithm specified at the time of input. Algorithm 2 describes the procedure while signing the message and how the message signature is generated. While signing the first message using a  $p_1$ -OTS public key as shown in Fig. 2, similar to Merkle's algorithm, Signer has to provide some extra information, here, we are providing value  $Z-p_1$  as the path verification data. Verifier, on the other hand, calculate the public key  $p_1$  for the received message and perform verification of the MTS public key by adding it to the provided path information data ( $Z-p_1$ ) followed by hash operation.

**Algorithm 2: Generation of message signature**

Input: Message  $m$

Output: message\_signature

Step I: signature=message\_sign\_generator( $m$ ,OTS\_pubkey)

Step II: pathinfo= $Z$ -OTS\_pubkey,

Step III: message\_signature={algo\_info,pathinfo,signature}

In Algorithm II, Message signature is calculated by using one-time signature and OTS path information is calculated for verification of MTS public key respectively in Step I and Step II. Original message signature with some extra information which Signer has to publish is combined with message signature and provided to verify MTS public key with the help of recalculated OTS public key and path information which is provided at the time of signing.

### Experimental Results

In order to validate the proposal, we have used the SHA-256 algorithm for hashing operation and LDWM one-time signature, and tested the proposed system in comparison to the Merkle's idea. We have implemented the LDWM schema using Merkle's idea and using proposed schema, considering security analysis made in [13], in JAVA with multiple message signing capability and comparison of these systems is carried out on the basis of the message signature size. Table 1 provides results of comparison between Merkle's approach and proposed approach on the basis of signature size generated by a general purpose signature system which implements LDWM signature schema LDWM\_SHA256\_M20\_W4 system for signing the message of size 1KB.

### Results

**Table 1**

No. of OTS	Size of Signature <sup>1</sup>		For	
	[bytes]		Merkle's Approach	
	Merkle's	Proposed	C <sup>2</sup>	N <sup>3</sup>
	Approach	Approach		
16	3074	2746	4	6
36	3334	2747	6	10
64	3594	2747	8	14
100	3855	2747	10	18
225	4505	2747	15	28
529	5545	2747	23	44
1000	6715	2748	32	62
5041	11785	2748	71	140
10000	15,556	2749	100	198
100489	43,766	2749	317	632

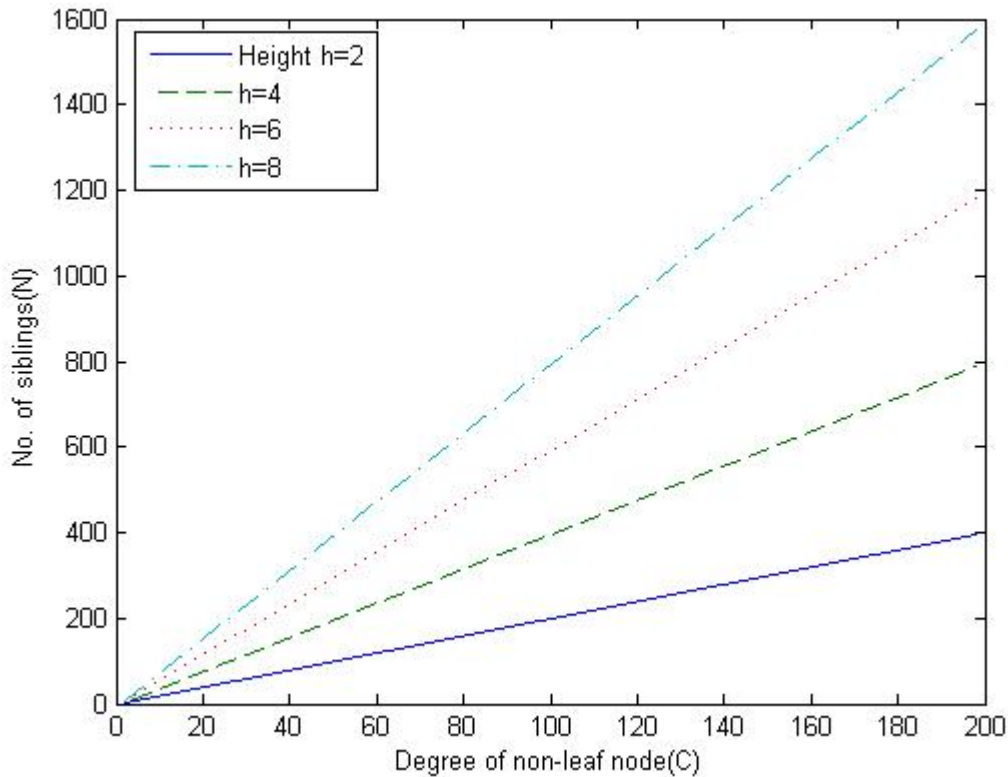
<sup>1</sup>It includes algorithm info, path information for OTS, and signature of the message.

<sup>2</sup>Degree of non-leaf node.

<sup>3</sup>Number of siblings involved in path information.

In Table 1, for example purpose, we have provided the results of Merkle's approach with the help of the hash tree of height 2, however it is not standard height. Fig. 3 shows the how the selection of height and capacity of non-leaf node affect the involvement of siblings in the formation of path information of OTS which directly affects the overall signature size of the message. Fig. 3 shows the involvement of N number of siblings in path information of OTS for different hash tree height h with varying degree of non-leaf node from 2 to 200. General purpose signature systems implemented using Merkle's approach, for greater multi-message signing capability, generally end up with the large message signature, due to more number of N involvement in path information of each OTS.





**Fig. 3. N vs. C for different height**

However, in proposed approach, instead of path information of each intermediate nodes, we have provided single information generated from leaves which result much better performance as compared to Merkle's approach.

### Conclusion

Researchers are looking for secure, efficient, and quantum resistant schemas, which fulfil the need of security even in quantum era. In hash-based signature system, the size of cryptographic parameter reducing the practicability of the system, nevertheless, continuous development in the field of hash-based signature promises for secure quantum resistant general purpose signature system. Recent development in quantum computing can threaten the conventional cryptosystems, due to the ability of quantum algorithms to use quantum parallelism. It is better to go for mathematically proved assumption rather using unproven mathematical problems such as integer factorization and discrete logarithm problem.

**References**

- [1] R.L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM, vol. 21, pp. 120-126, Feb.1978.
- [2] Whitfield Diffie and Martin E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory IT-22, 1976, no. 6, pp. 644–654.
- [3] Taher ElGamal, *A public key cryptosystem and a signature schema based on discrete logarithms*, IEEE Transactions on Information Theory IT-31 1985, no. 4, pp. 469–472.
- [4] Peter W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, in Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1994, pp. 124–134.
- [5] Victor Shoup, *Lower bounds for discrete logarithms and related problems*, in Proc. Eurocrypt'97, Lecture Notes in Computer Science, Springer-Verlag, 1997, pp. 256–266, vol. 1233.
- [6] R. P. Feynman, *Simulating Physics with Computers*, International Journal of Theoretical Physics, vol. 21, pp. 467–488, 1982.
- [7] R. Merkle, *Secrecy, authentication, and public key systems*, Stanford Univ., 1979.
- [8] L. Lamport, *Constructing digital signatures from a one way function*, SRI International Computer Science Laboratory, Technical report, SRI-CSL-98, 1979.
- [9] V. Lyubashevsky, D. Micciancio, C. Peikert, and A. Rosen, *Swift: A modest proposal for the hashing*, in FSE, 2008, pages 54-72.
- [10] R. C. Merkle, *A certified digital signature*, in Proceedings of Advances in Cryptology – CRYPTO '89, vol. 435 of LNCS, pages 218-238. Springer, 1989.
- [11] Markus Jakobsson and Michael Szydlo et al, *Fractal Merkle Tree Representation and Traversal*, RSA-CT '03.
- [12] Michael Szydlo, *Merkle Tree Traversal in Log Space and Time*, Eurocrypt 2004.
- [13] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas Hulsing, and Markus Ruckert. *On the security of the Winternitz one-time signature schema*, In A. Nitaj and D. Pointcheval, editors, Africacrypt 2011.