International Journal of Information & Computation Technology.
ISSN 0974-2239 Volume 5, Number 1 (2015), pp. 1-10
© International Research Publications House
http://www. irphouse.com

Signing Multiple Messages using MD6 Hash Algorithm

S. S. Hatkar[1], T.R.Sontakke [2], A. M. Fajge [3]

[1] Associate Prof., Dept. of CSE,SGGSIE&T, Vishnupuri,Nanded-431606, India
sshatkar@sggs.ac.in
[2] Ex. Director, SGGSIE&T, Vishnupuri, Nanded-431606, India
trsontakke@sggs.ac.in
[3] M.Tech Student, Dept. of CSE, SGGSIE&T, Vishnupuri, Nanded-431606, India
fajgeakshaym@gmail.com

Abstract

In this paper we introduced an approach for signing multiple messages using MD6 hashing. Merkle's signature scheme is studied and implemented for a binary tree. This approach is simple to implement, and can resist the attacks that are even made with the help of quantum computer. Both the sizes of the signature are compared generated by proposed approach and Merkle's approach. Overall the space required for signature size of proposed algorithm is reduced as compared to Merkle's signature size. The size of the path information and number of hash operations that are required for verification of Merkle's public key are also reduced. Experimental results are compared and show that our approach is more computational efficient.

Keywords: quantum computing, post-quantum cryptography, asymmetric key, digital signature algorithm, hash-based cryptography, merkle tree signature, elliptic curve cryptography.

1. Introduction

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 sender signs a message such that receiver 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 vary 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.

When message m is signed, by X so that any Verifier can verify the signature as Sx(m) by using a public-key cryptosystem. When a message m is signed 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 1991as 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.

Factorization problem may occur during signing a message by using public key cryptosystem. 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]".

Since the several key enhancements have been made in quantum computing, post-quantum cryptography has become a topic of research such as the invention of quantum algorithms, and quantum computers in the last few years. Many Cryptosystem protect 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. 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. So, these cryptosystems are not provably secure since the mathematical structure of the problem is not provably hard. Therefore the security schemas, which are most widely used today, have implementation based on these problems and remain secure over the past years due to the fact that there is no known classical algorithm which can solve these problems in polynomial time. Shor's algorithm boosts the development of quantum computer and post-quantum cryptosystems. Quantum computer deals with thousands of qubit, can make current cryptosystems which rely on integer factorization problem and discrete logarithmic problem vulnerable. 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 post-quantum cryptosystems.

One-Time signature (OTS) system is gaining more attention because of their post-quantum security and their appropriateness for compact implementations. This 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. This signature system provides asymmetric message authentication. It produces a public/private key-pair and the generated message signature can be verified using OTS public key.

The rest of paper is organized into five sections. Second section describes the work related to OTS systems, Third Section elaborates the Merkle's Tree Signature (MTS) to solve problem of large public key size to sign 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 fifth section, we discussed the results compared with MTS and in the last section the paper is concluded with some observations.

2. Signature Schemas

One-Time Signature systems were well considered from 1990 and have advanced from new development over the last decade. The security of One-Time Signatures is based on cryptographic secure hash functions. A hash function H is cryptographic secure, if it is preimage resistant, second preimage resistant and collision resistant. The schemas called post-quantum signature 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.

Lamport proposed Lamport-Diffie One-Time Signature schema (LD-OTS) detailed in [8]. it is a signature scheme in which the public key can only be used to sign a single message. The security of the LDOTS is based on cryptographic hash functions. Any secure hash function can be used, which makes this signature scheme very adjustable. If a hash function becomes insecure it can easily be exchanged by another secure hash function.

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. This is 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 signing multiple messages.

To overcome the problem of signing large messages, Lamport One-Time Signatures multi-bit version is used. here, sender selects random values X [X_0, X_1, X_2, X_3…, X_{255}] and Y [Y_0, Y_1, Y_2, Y_3…, Y_{255}] and publishes the set of the public key, such as, [H (X_0), H (X_1), H (X_2), H (X_3)…, H (X_{255})] for X and [H (Y_0), H (Y_1), H (Y_2), H (Y_3)…, H (Y_{255})] for Y. Sender can use these public keys to sign arbitrarily long message, m, efficiently by running many instances in parallel. This multi-bit version just help sender 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 sender has 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, Sender has to publish public keys of the size (256 + 256) * output size of Hash function. If a sender is using a hash 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 Sender 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.

One major disadvantage of the Lamport One-Time Signature Scheme is the big size signature. Winternitz One-time Signature Scheme reduces the signature size at the cost of hash operations. 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). Sender needs a checksum to ensure multi-message signing capability, as an attacker can generate signature ofremaining 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.

3. Merkle’s Tree Signature (MTS)

Merkle’s tree signature [7], provides a method to sign multiple messages, without the gigantic cost of storing two secret values per bit to be signed. MTS contains 2^d possible number of signatures, bound together in a Merkle’s tree structure of depth d, which helps to solve key distribution problem of LD-OTS. MTS solves the problem of signing multiple messages. 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. The main idea of the MTS algorithm is to calculate the needed subtree from left to right saving the nodes using a stack. 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 time consuming.

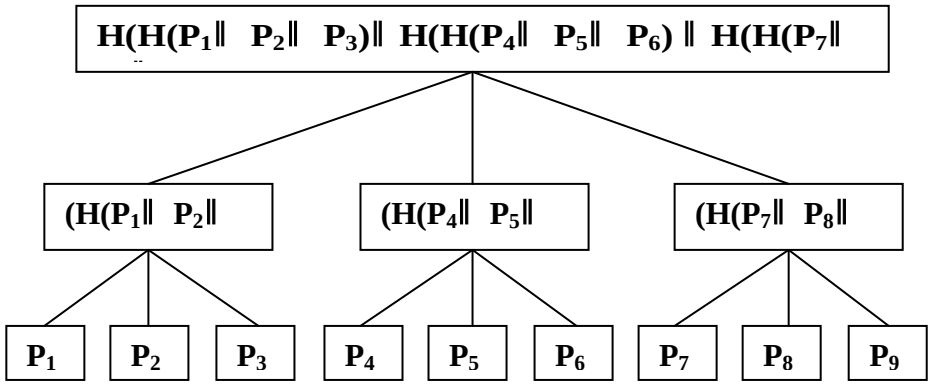

Fig.1 Merkle Hash Tree

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.

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.

If Signer is using p_2 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_1, 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.

MTS parameter plays vital role in creating an optimal tree such that cost will be minimized [11]. 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. If a tree has height '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 problem with MTS 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 signer has to remind all intermediate hash values up to the root node. MTS algorithm required O(h) hash evaluations for each OTS verification with $O(h^2)$ space requirement for storing intermediate hash values. In [12] 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 [13], 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.

4. Proposed Approach

Here we introduce an approach to sign multiple messages using MD6 hash algorithm. The main difference between proposed approach and MTS 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. In proposed approach, tree height is 1 and all OTS reside at level 1 of the tree as shown in Fig. 2.

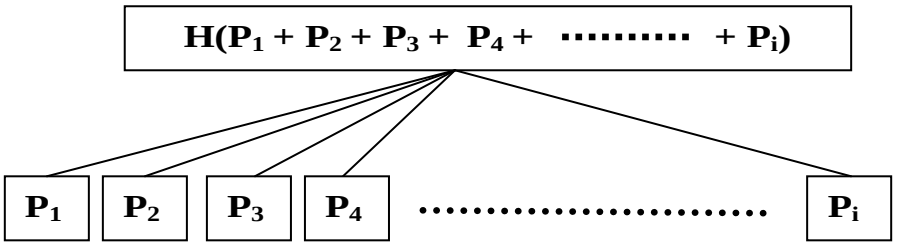

Fig. 2 Proposed Approach

In first level of a tree contains i number of OTS public keys from p_1 to p_i. 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 the entire 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. $Z= (p_1+ p_2 + p_3 + p_4 +$ -------- $+ p_i)$; 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)

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. 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: 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}

Procedure of signature generation in described in algorithm 2 while signing the message. (p_1-OTS) public key is used to sign the first message as shown in Fig.2. similar to Merkle's algorithm, Signer has to provide some extra information; 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.

Step I is for calculating message signature by using one-time signature. Step II is for calculating OTS path information for verification of MTS public key. 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.

5. Experimental Results

We have used LDWM one-time signature and proposed system using MD6 for hashing operation in comparison to the Merkle's idea. We have implemented the LDWM schema using Merkle's idea, considering security analysis made in [14], in JAVA with multiple message signing capability. Comparisons of these systems are 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. General purpose signature system which implements LDWM signature with MD6 hashing, 20 bytes of length of each elements of LDWM signature, Winternitz's paramenter w=4 and message size of 1KB.

Table 1: Using MD6

Number of One Time Signature	Size of Signature in bytes		For Merkle's Approach	
	Merkle's Approach	Proposed Approach	Degree of non leaf node	Number of nodes involved in a path
4	2958	2823	2	2
9	2993	2889	3	4
16	3172	2898	4	6
36	3231	2923	6	10
64	3476	2956	8	14
100	3957	3026	10	18
255	4628	3254	15	28
529	5748	3354	23	44
1024	6995	3526	32	62
5041	12343	3612	71	140
10000	17231	4123	100	198
100489	46653	5294	317	632
116281	55765	6623	341	680

We have provided the results of Merkle's approach with the help of the hash tree of height 2 and proposed approach; 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. 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 is shown in the figure. 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 involvements in path information of each OTS.

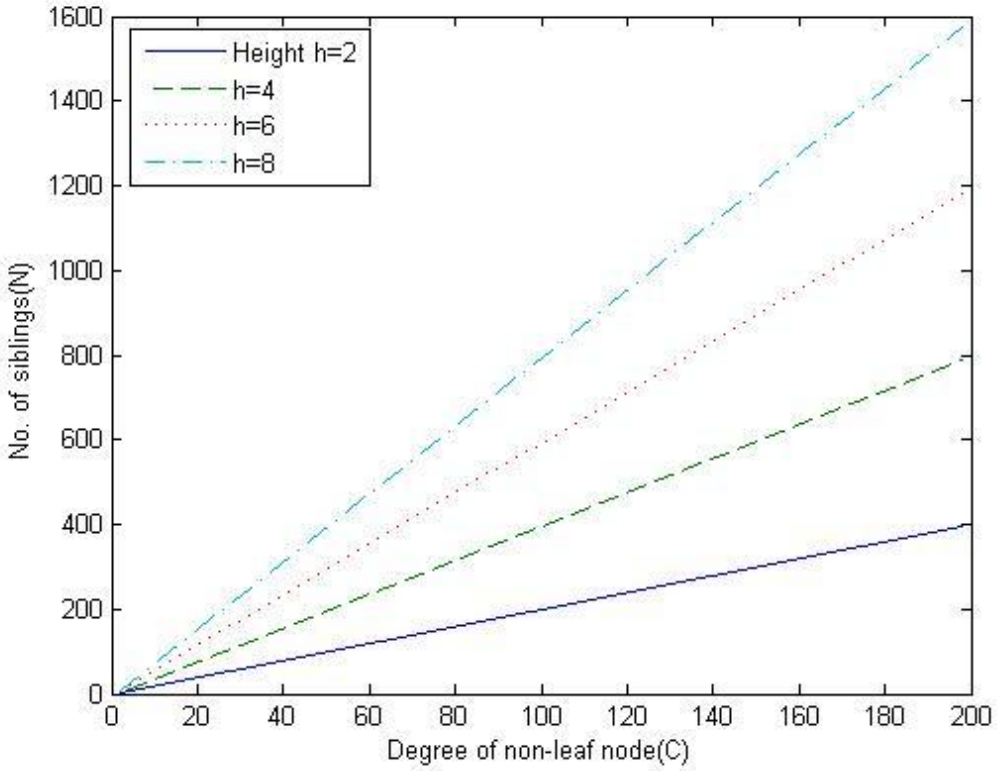

Fig. 3 N vs. C for different height

Therefore, we have provided single information generated from leaves instead of path information of each intermediate node. This approach gives us better performance.

6. Conclusion

Efficient and secure with quantum resistant schema can be achieved to fulfil the need of security in quantum era. We receive the signature of smaller size which reduces the burden of the bandwidth of communication channel. This will helpful for secure quantum resistant general purpose signature system. Since the ability of quantum algorithms to use quantum parallelism, the developments in quantum computing can threaten the conventional cryptosystems. For efficient implementation of hash based signatures in real world, there is a need to reduce the signature size. Therefore, we should go for mathematically proved assumption instead of unproven mathematical problems.

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, *Swifft: 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] Georg Becker, Merkle Signature Scheme, Merkle Tree and Their Cryptanalysis.

[12] Markus Jakobsson and Michael Szydlo et al, *Fractal Merkle Tree Representation and Traversal*, RSA-CT '03.

[13] Michael Szydlo, *Merkle Tree Traversal in Log Space and Time*, Eurocrypt 2004.

[14] 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.