Stateful Hash-based Digital Signature Schemes for Bitcoin Cryptocurrency

15th International Conference on Electronics Computer and Computation (ICECCO 2019)

[1]Noel, Moses Dogonyaro
Cyber Security Science Department
Federal University of Technology,
Minna, Nigeria
moses.noel@futminna.edu.ng

[2]Waziri, Onomza Victor
Cyber Security Science Department
Federal University of Technology,
Minna, Nigeria
victor.waziri@futminna.edu.ng

[3]Abdulhamid, Muhammad Shafii
Cyber Secuiryt Science Department
Federal University of Technology,
Minna, Nigeria
shafii.abdulhamid@futminna.edu.ng

[4]Ojeniyi, Adebayo Joseph
Cyber Security Science Department
Federal University of Technology,
Minna, Nigeria
ojeniyia@futminna.edu.ng

***Abstract* - Modern computing devices use classical algorithms such as Rivest Shamir Adleman (RSA) and Elliptic Curve Digital Signature Algorithm (ECDSA) for their security. The securities of these algorithms relied on the problem and difficulty of integer factorization and also calculating the Discrete Logarithm Problems. With the introduction of quantum computers, recent research is focusing on developing alternative algorithms which are supposed to withstand attacks from quantum computers. One of such alternatives is the Hash-based Digital Signature Schemes. Chosen hash-based signature schemes over classical algorithms is because their security is on the hash function used and that they are meta-heuristic in nature. This research work presents basic analysis and the background understanding of Stateful Hash-based Signature Schemes, particularly the Lamport One-Time Signature Scheme, Winternitz One-Time Signature Scheme, and the Merkle Signature Scheme. The three schemes selected are stateful, hence has common features and are few-time hash-based signature schemes. The selected Stateful Hash-based Digital Signature Schemes were analyzed based on their respective key generation, signature generation, signature verification, and their security levels. Practical working examples were given for better understanding. With the analyses, Merkle Signature Scheme proves to be the best candidate to be used in the Bitcoin Proof of Work protocol because of its security and its advantage of signing many messages.**

Index Terms - Post-Quantum Cryptography, Hash-based Digital Signature, Cryptocurrency, Bitcoim

I. INTRODUCTION

Among the widely used cryptographic primitives today is digital signature [3]. Digital signature algorithms are important today in modern communication because they provide and guarantee authenticity, integrity, and non-repudiation. In secure communication protocol, digital signatures are used to protect software updates, online banking, e-commerce and other areas of applications such as electronic cash [12].

Digital signatures used hash functions for effective and secure communication. Among the digital signature schemes used today are the Elliptic Curve Digital Signature Algorithm (ECDSA), Rivest Shamir Adleman (RSA), Digital Signature Algorithm (DSA) [5]. The security provided by these algorithms is as a result of the difficulty that exists in factoring large complex integers or computing Discrete Logarithm Problem. Shor, in 1994 developed a quantum algorithm that could factor large complex integers and as well solve the Discrete Logarithm Problem in polynomial-time [4]. It means that classical computers that use digital signatures are insecure. The insecurity of classical algorithms prompted the research of post-quantum digital signature algorithms that could withstand quantum attacks especially in electronic cash system (such as bitcoin) in the nearest future.

Bitcoin as defined by [12] is a network of systems that does not have a central connection and by implication does not depend on a trusted third party for the processing of its transactions. Bitcoin transactions are recorded and maintained by a public ledger called the blockchain. Since its development by Satoshi Nakamato in 2008 [12], bitcoin has proven to be acceptable by many users as compared to other digital currencies. Bitcoin transactions are supported by the use of digital signatures. To transfer some coins to someone else, the current owner adds a link to a chain of blocks, thereby creating a new transaction.

In bitcoin, the security of transactions relies on the Proof-of-Work (PoW) protocol which aimed at preventing double spending of the coin [1]. The PoW is a measure to find a pre-image of an output of a cryptographic hash function. The PoW technique will be insecure with the use of quantum computers. A quantum computer can apply Grover's search algorithm to execute the PoW faster than classical computer that uses ECDSA or RSA [11]. With these limitations, hash-based digital signature schemes are therefore good alternative algorithms in securing bitcoin transactions.

The research work is arranged this way: part II briefly summarizes relevant literatures of stateful hash-based digital signature Schemes; part III gives details explanation of some selected families of hash-based signature schemes; the authors in part IV discussed application of hash-based digital signature schemes as an alternative to ECDSA that is used in bitcoin security. Section V presents the research conclusion and recommendations for future work.

II. REVIEW OF RELATED LITERATURES

The advent of quantum computing and its application capabilities called for an open research gap in post-quantum cryptography. It is in line with this motivation that [13] builds on the multi-time signature schemes recommended by Raph Merkle in 1979 and showed that the initial MSS is difficult to forge. The contribution to his work was the development of an improved version of the Merkle Signature Scheme with cost reduction in terms of key

978-1-7281-5160-1/19/$31.00 ©2019 IEEE

generation. However, [7] carried out analysis of Winternitz One-Time Signature Scheme (W-OTS) and its security levels. The research work proof that W-OTS is also difficult to forge when applied with pseudo-random functions. Comparative study of post-quantum hash-based digital signature was done by [9] using hierarchical method on different Hash-based Digital Signature Schemes. The research work suggests future implementation in Public Key Infrastructure (PKI). "Reference [10] considered using two hash-based digital signatures (L-OTS and W-OTS) to analyze their strength and security levels. The results showed that W-OTS signature length is shorter than the LD-OTS".

III. HASH BASED DIGITAL SIGNATURE

Hash-based digital signatures are either stateful or stateless. In stateful signature scheme, signing a message reads a secret key and the message then a signature is generated which include the updated secret key. This means that a signer must maintain a state that is modified every time a signature is issued. In stateless signature scheme (such as SPHINCS) has a large tree-of-trees, but at the bottom of the tree, are a number of Few-Time-Signature (FTS). A message is signed by a signer by picking a random FTS, and then authenticates that through the Merkle tree up to the root. Using FTS, you do not need to update any state when generating a signature [8]. The hash-based digital schemes discussed in this research work are mainly stateful hash-based signature schemes. The basic fundamental of hash-based signature schemes is the One-Time Signature scheme (OTS). The first and the most intuitive of OTS was developed by Lamport and Diffie in 1979 also known as Lamport Diffie One-Time Signature Scheme (LD-OTS). The OTS allows using a pair of key to sign one message at a time [6]. The use of one way function is the characteristic of LD-OTS and is given as: $f: \{0,1\}^n \to \{0,1\}^n$ where n is a positive integer, and a hash function that is cryptographically secure given as: $H_c: \{0,1\}^* \to \{0,1\}^n$

A. Generating key pairs in LD-OTS

Let's assume LD-OTS signature key to be K with $2n$ bit strings which has length n signature key selected randomly. Therefore:

$$K=(k_{n-1}[0],k_{n-1}[1],...k_1[0],k_1[1],k_0[0],k_0[1]\in_R \{0,1\}^{(n,2n)} \quad (1)$$

i. To verify LD-OTS:

Let's assume L to be the verification key to be computed given as:

$$L=(l_{n-1}[0],l_{n-1}[1],...,l_1[0],l_1[1],l_0[0],l_0[1]\in\{0,1\}^{(n,2n)} \quad (2)$$

Such that $l_i[j] = f(k_i[j]),\ 0\le i\le n-1,\ j=0,1 \quad (3)$

From (3), it is shown that key generation in LD-OTS needs $2n$ evaluation of function f. This means that $2n$ bit strings with length n is the requirement for both the signing key as well as the verification key.

ii. Generating Signature using LD-OTS

Suppose $P\in\{0,1\}^*$ is a document to be signed by a signature key K illustrated in (1).

Suppose $q(P)=dm=(dm_{n-1},...,dm_0)$ denotes the digest of the message P. The signature of LD-OTS will be written as:

$$\delta=(k_{n-1}[dm_{n-1}],...,k_1[dm_1],k_0[dm_0]\in\{0,1\}^{(n,n)} \quad (4)$$

Note that the signatures are chosen as a function of the digest of the message dm. In this signature, i^{th} is the bit string and is given as $k_i[0]$ when i^{th} bit string in message digest dm equals zero (0) and $k_i[1],$ otherwise. In this case, no evaluation of f function is required in the signing. Therefore the signature length is n^2.

iii. Signature verification with LD-OTS

Given the signature $\delta=(\delta_{n-1},...,\delta_0)$ of the document P shown in (4), the message digest which is $dm=(dm_{n-1},...,\delta_0)$ is calculated by the verifier. The verifier needs to check whether

$$(f(\delta_{n-1}),...,f(\delta_0))=(l_{n-1}[d_{n-1}],...l_0[d_0]) \quad (5)$$

In this equation, the verification of the signature requires n evaluations of the function f.

iv. Application scenario

Let's the positive integer n be 3, and the hash function $f: \{0,3\}^3 \to \{0,1\};\ k \mapsto k+1 \bmod 8.$

Let's assume dm to be the hash value in the message P given as (1,0,1). The signature key is chosen as:

$K=(k_2[0],k_2[1],k_1[0],k_1[1],k_0[0],k_0[1]$ .Yields a 3 by 6 matrix as:

$$\begin{bmatrix}1&0&0&1&1&0\\1&0&1&1&0&1\\1&0&1&0&1&0\end{bmatrix}\in\{0,1\}^{(3,6)}$$

The corresponding verification key can be computed as:

$$L=(l_2[0],l_2[1],l_1[0],l_1[1],l_0[0],l_0[1]=$$

$$\begin{bmatrix}0&0&1&1&1&0\\0&0&0&1&1&1\\0&1&0&1&0&1\end{bmatrix}\in\{0,1\}^{(3,6)}$$

Given that the message digest $dm=(1,0,1)$; this implies that the signature of the digest will be:

$$\delta=(\delta_2,\delta_1,\delta_0)=(k_2[1],k_1[0],k_0[1]=\begin{bmatrix}0&0&0\\0&1&1\\0&1&0\end{bmatrix}\in\{0,1\}^{(3,3)}$$

Note that the LD-OTS signature keys must be used only once to avoid an attacker been able to know the signature of

the digest. For example, if the security parameter n is chosen as 4; assuming the signer signs two messages whose digest are:

$dm_1 = (1011) and\ dm_2 = (1110)$ maintaining the same signature key. These digests would give the following signatures as:

$\delta_1 = (k_3[1], k_2[0], k_1[1], k_0[1])$

$\delta_2 = (k_3[1], k_2[1], k_1[1], k_0[0]$

From these signatures $\delta_1\ and\ \delta_2$ the hacker understands $k_3[1], k_2[0], k_2[1], k_1[1], k_0[0], k_0[1]$ because the hacker can understand the signature key. The hacker may apply this information to determine the actual signatures of the two messages whose digests are given as:

$dm_3 = (1,0,1,0)\ and\ dm_4 = (1,1,1,1).$ It would be difficult for the hacker to find the actual message provided the used hash function is secure.

v. The Security of LD-OTS

LD-OTS security depends on the cryptographic hash functions [6]. For the scheme to be flexible, any hash function could be used. The security notion of LD-OTS could be affected if the hash function f is inverted by an attacker who intends to forge the signature of the sender. For example; Let's $K = 01001101 and\ f(K)$ be a given function that can converts 0's with 1's and vice versa. Then $L = 10110010$. Assuming Alice wants to sign two messages given as: $M1 = 1010$ and $M2 = 1101$ with the same private key K. By LD-OTS, Alice computes the corresponding signatures as: $\delta_1 = 1010$ and $\delta_2 = 1011$ then send to Bob. By this, Bob knows $\delta 1, \delta 2,$ and M1,M2. It will be easy for Bob to forge a new message $M3 = 1111$ with its corresponding signature $\delta_3 = 1011$ by just combining the previous two messages. A signer has to save $2 * n$ hash values in order to sign a message given as $M = \{0,1\}^n$. To attain $O(2^{80})$ security level in LD-OTS, the hash function used must be at least 160 bits. This means that both private and public keys are expected to be at least $160 * 2 * n = 320 * n$ bits.

B. (W-OTS)

i. Generating Keys-W-OTS approach

In LD-OTS, the signature key and signature generation is efficient, but the magnitude of the signature is large [9]. The aim of the W-OTS is to produce few signatures. The notion is one string in the OTS key should be able to sign many bits of the digest message. This can be illustrated as:

Suppose $H:\{0,1\}^* \rightarrow H:\{0,1\}^n$ represent a function that is cryptographically secure with a parameter given as w such that $w \in N$ is selected, then L can be calculated. The variable L represents the number of elements in the private key and has the following formula:

Let $L_1 = \lceil |H(m)/w| \rceil$ and $L_2 = \lceil \lfloor \log 2L1 \rfloor + 1 + w/w \rceil$

$$L = the\ sum\ of\ L_1\ \&\ L_2 \quad (6)$$

L_1 represents the number of all the private key elements required for signing the message digest, when signing $w - bits$ simultaneously. Whereas L_2 represents the number of private key elements to sign the checksum of the message digest. All elements of the private key are generated from the Pseudorandom Number Generator (PRNG) and consist of $n - bits$ each. That is;

$$Sk = (Sk_0, ..., Sk_{l-1}) \quad (7)$$

The public key is generated when F is applied to every element from the private key $2^w - 1$ times. The public key is supposed to be available to the public along with parameter w ,functions $F\ and\ H$.

$$Pk_i = F2^w - 1(Sk_i);\ for\ i = 0, ..., L-1 \quad (8)$$

$$Pk = (Pk_0, ..., Pk_{l-1}) \quad (9)$$

The table I represent correlation in numbers of how key elements decrease and evaluations of F increase while w also increases.

TABLE I

Differences between W, private key elements, and evaluations of *F* when n=256

Key length (L)	133	40	67	55	45	39	34
Evaluation of F	399	630	1005	1705	2055	473	8672
Signature size in kb	4.2	2.9	2.1	1.8	1.4	1.2	1.1

ii. W-OTS Signing process

To sign a message, the sender has to select the parameter w and L private and public keys. Thereafter, hash the message m with function H to produce a message digest $H(m)$. Next, the sender needs to slice the message in pieces so that each piece consist of $w - bits$. If the message digest is not divisible by *w*, the sender should append additional zeros in the most left position resulting in

$$dm = (dm_0, ..., dm_{l-1}) \quad (10)$$

This is in base 2^w notation. The checksum Cs and dm can be calculated later. To do so, the sender has to calculate the sum of all differences between $2^w = 8_{10}$ and each of the dm_i from the sliced message digest. This is done by using the equation,

$$Cs = \sum_{i=0}^{l1-1} 2^w - dm_i \quad (11)$$

$$Cs = (Cs_0, Cs_1, ..., Cs_{l2-1}) \quad (12)$$

Then, the sender needs to concatenate both the message digest, and the checksum to variable B. The variable B consists of L elements in 2^w notation given as:

$$B=(b_0,b_1,...,b_{l-1})$$

The new variable B is the actual message to sign. In other words, W-OTS signs both message digest and the checksum of the message digest. To generate the signature, the sender needs to apply b_i many times the function F on input S_k from index i.

$$\delta_i = F^{bi}(Sk_i): \ for \ i=0,...,L-1 \quad (13)$$

$$\delta=(\delta_0,\delta_1,...\delta_{L-1} \quad (14)$$

Each of n-bit length which would yield the signature size of $L-bits$

iii. Signing a message

Let's assume that the sender has chosen Winternitz parameter to be w = 3 and a small message digest $H(m)=16\,bits$. Then $H(m)$ would be expressed as:

$$H(m)=1001111010100011$$

Appending two zeros to the left, the message becomes:

$$H(m)=0010011110100011 \quad (15)$$

The parameter L is calculated to be 8 elements each. Next, the sender slices the message digest 3 by 3 and append additional two zeros to the left as shown in equation (15)

$$dm=(dm_0,...,dm_5)=001001\ 111\ 010\ 100\ 011 \quad (16)$$

Thereafter, the checksum Cs of the message digest dm can be calculated. The sender has to calculate the sum of all differences between $2^w=8_{10}$ and each of the dm_i from the message digest. This can be done with the following formula:

$$\sum_{i=0}^{l-1} 2^3 - dm_i .$$

Table II shows the calculation of the checksum for this particular instance, when $H(m)=16\,bits \ \ and \ \ n=3$

This implies that: $dm_1=\lceil H(m)/w \rceil = 6$

The checksum Cs needs to be converted to binary sliced 3 by 3 bits with the extended zeros at the left position.

$$Cs=(Cs_0,Cs_1,Cs_2,Cs_3,Cs_4,Cs_5) = (1,\ 1,\ 7,\ 2,\ 4,\ 3)$$

Thus, $\sum_{i=0}^{l-1} 2^w - dm_i \ = \ \sum_{i=0}^{6-1} 2^3 - dm_i$

= (8-1) + (8-1) + (8-7) + (8-2) + (8-4) + (8-3)

= 7 + 7 + 1 + 6 + 4 + 5

= 30_{10}

Thus, B = 30_{10} = 11110_2

Padding B, gives, 011 110

Concatenating both message digest and checksum to create B consisting of L = 8 elements

B = d||C (17)

Using "13", gives the following signatures;

$$\delta=Sig(B)=F^1(Sk_0),F^1(Sk_1),F^7(Sk_2),F^2(Sk_3),F^4(Sk_4),F^3(Sk_5)\,||\,F^3(Sk_6),F^6(Sk_7) \quad (18)$$

In this instance, the signature size;

$\delta=8x16=128\,bits,$ since F is a length preserving function.

TABLE II

Calculation of checksum in W-OTS; when $w=3$

$2^w=2^3=8$	8	8	8	8	8	8
b_i (binary)	001	001	111	010	100	011
b_i in decimal	1	1	7	2	4	3
Checksum	7	7	1	6	4	5

iv. W-OTS Signature verification process

To verify the signature (δ, m, Pk), the receiver has to perform similar calculations as the sender while signing. Knowing both function F and H along with parameter w, the receiver obtain all necessary information for signature verification. First, the message m is hashed to obtain message digest $H(m)$, then appending zeros in the most left position. Next, calculate the checksum Cs and append zeros to the most left if needed.

$$Cs=\sum_{i=0}^{L1-1} 2^w - d_i \quad (19)$$

The receiver concatenates the message digest $H(m)$ with checksum Cs to create B containing L elements.

$$B=(b_0,...,b_6) \ \ =d\,||\,Cs$$

The integer values from $b_i \ in \ 2^w$ notation contain information about how many times a specific part of the signature has the function F been applied on. During the key generation phase, the sender has applied function F on private key elements 2^w-1 times to obtain the public key.

$$Pk_i = F^{2w-1}(Sk_i): \ for \ i=0,1,2,...l-1$$

$$Pk=(Pk_0,...Pk_{l-1})$$

Combination of this information implies that the receiver has appliedy the function F on the parts of the signature 2^{w-1-bi} times to reconstruct the sender's public key. Thereby verify the signature.

$$\delta=(\delta_i,...,\delta_{l-1}) \quad (20)$$

$$\forall : F^{2w-1-bi}(\delta_i) = Pk_i \quad (21)$$

If the calculated values match the sender's public key, then, the signature is valid, otherwise it would be rejected.

v. Signature Verification scenario

Considering the signing process in subsection (iii), where it is necessary to calculate the message digest $H(m)$, slicing it up and interpret it as integer values in 2^w notation is required.

Given that Cs = 30_{10} = 11110_2

Appending one zero to the most left position to Cs and dividing by w implies; Cs = 011 110

Concatenate the message digest $H(m)$ with the checksum Cs to create B containing L elements gives:

$$B=H(m)||C=001001111010100011011110_2 \quad (22)$$

Interpreting these values as integers, gives;

$$B = (1, 1, 7, 2, 4, 3, 4, 2)_{10} \quad (23)$$

Since parameter $w=3$, then the sender needed to apply function F to all private key elements 2^w-1 times (which is 7 times) to generate the public key. Thereafter, applying the formula $\forall: F^{2^w-1}-bi(\delta_i)=?\ Pk_i$ to reconstruct the sender's public key to give:

$$Pk_i = F^1(\delta_0),\ F^1(\delta_1),\ F^7(\delta_2),\ F^2(\delta_3),\ F^4(\delta_4),\ F^3(\delta_5),\ F^4(\delta_6),\ F^2(\delta_7) \quad (24)$$

vi. Security of W-OTS

"Reference [17] was the first to proof that a generic W-OTS is difficult to forge under adaptive chosen message attacks". There are two security properties that are applicable to W-OTS; pseudorandom property and key one-wayness. These security properties make W-OTS quantum resistant.

C. Merkle Signature Scheme (MSS)

The MSS is based on hash trees (known as Merkle trees) and it is a one-time signature like the LD-OTS. It was introduced by R. Merkle in 1979 as an alternative to the traditional digital signature (RSA, DSA). The advantage of MSS is that it is tested to be resilient to quantum computer attacks. The MSS security is the present of the hash function used [14].

i. Key generation in MSS

MSS can only be used to sign a few number of messages using one public key [3]. The total number of messages is $N=2^n$. In MSS key generation, the public keys (*pubkey)* χ_i are generated first then the private keys γ_i with 2^n OTS scheme. Given private key γ_i, with $1\le i\le 2^n$, a hash value $h_i=H(\gamma_i)$ is calculated and with these hash values h_i, a hash tree is built. Fig 1 illustrates a Merkle tree of height H = 3.

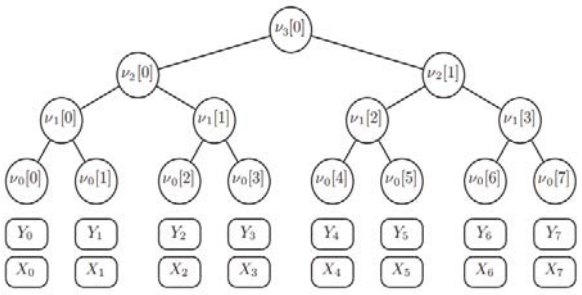

Fig 1. Merkle hash tree with height H = 3 [7]

As seen in fig. 1, the verification keys are $\gamma_0,...,\gamma_7$, while $\chi_0,...\chi_7$ are the signature keys.

$a_{i,j}$ is referred to as the nodes of the tree, were i represents the level at which the nodes are located (for example $a_{0,1}$ is level zero, node 1). The node level is the gab from the node to a particular leaf. For example, a leaf has level $i=0$ and the root has level $i=n$. The hash values h_i are the leafs of the tree, such that $h_i=a_{0,i}$. The inside nodes are the hash value of the concatenation of its two children. For example;

$$a_{1,0}=H(a_{0,0}\,||\,a_{0,1})\ and\ \ a_{2,0}=H(a_{1,0}\,||\,a_{1,1}).$$

This is how the tree is made up. This include $2n$ leafs and $(2^{n+1}-1)$ nodes. The tree has a root and it is given as $a_{n,0}$ and it is referred to as the public key.

i. Generating Signatures in MSS

Signing a message M can be achieved by signing first with OTS to obtain the signature "*sig"* and by utilizing the public key and private key pairs (χ_i,γ_i) respectively. The route in the hash tree from $a_{0,1}$ to the root could be represented as A. The path A consists of $n+1$ nodes, that is, $A_0,...,A_n$ with $A_0=a_{0,1}$ being the leaf and $A_n=a_{n,0}=pubkey.$ In calculating the path A, all entities of the nodes $A_0,...,A_n$ are needed. Note, A_i is an entity of A_{i+1}. This node is called the authentication path $auth_i$ such that;

$A_{i+1}\ is\ equal\ to\ H(A_i\,||\,auth_i)$. To get the number of nodes n; $auth_0,...,auth_{n-1}$ are required to calculate all the nodes in the path 'A'. These nodes are calculated and saved as $auth_0,...,auth_{n-1}$ respectively. The final signature of message M is: $'sig'=(sig'||\,auth_0\,||\,auth_1\,||\,auth_{n-1})$ in MSS.

iii. How to Verify a Signature

To verify any signature, the receiver must know the *pubkey,* the message M and the signature; $Sig=(Sig'||\,auth_0\,||\,auth_1\,||,...,||\,auth_{n-1})$. Firstly, the receiver checks the OTS Sig' of a given message M. If the Sig' is true, then the receiver can calculate $A_0=H(\gamma_i)$ by performing hashing operation on the public key of the OTS.

iv. The security of MSS

The MSS security as explained by [7] is forward secure, one-time signature, and collision resistant. A cryptographic hash function G is collision resistant if it is hard to find two inputs that hash to the same output. Given p and q, such that $G(p)=G(q)$ and ($p\ne q$). However, to forging an MSS, an attacker is required to compute the pre-images and second pre-images hash function.

D. Comparison of L-OTS, W-OTS, and MSS

In table III, it is shown that the L-OTS key size is $2n^2$ and that of W-OTS is np, while in MSS the private key size are in the range of $1 \le i \le 2^n$. To generate a key, L-OTS requires $2n$ evaluation of function f, W-OTS requires $t(2^{\omega}-1)$ evaluation of function f, and in the MSS, the key generation is 2^H (H is the tree height). L-OTS and MSS does not use function f for signature generation, while W-OTS requires $t(2^w-1)$ parameters. On the use of function f for signature verification, L-OTS needs to use the value of parameter n for signature verification, W-OTS needs to apply the parameters $t(2^w-1)$ to verify signatures; while MSS requires the parameters $t(2^w-1)/2$ for the same process. The summary of these analyses is illustrated in table III.

TABLE III
Comparison between L-OTS, W-OTS, and MSS Schemes

Parameters	L-OTS	W-OST	MSS
Key size	$2n^2$	$w \ge 2$	$1 \le i \le 2^n$
Key generation	2n evaluation of f	$t(2^w-1)$ evaluation of f	2^H leaves
Signature length	n^2	$n(t)$	$Sig'\|\|auth_0, \ldots, \|\|auth_{n-1}$
f for Signature generation	Not used	$t(2^w-1)$	Not used
Use of f for signature verification	n	$t(2^w-1)$	$t(2^w-1)/2$
Security levels	Hash function used	EU-CMA	Pre-image of the hash function

Note: n is a positive integer, w is Winternitz parameter, t is the bits string, and f is the hash function.

IV APPLICATION OF HASH-BASED DIGITAL SINGNATURES ON BITCOIN CRYPTOCURRENCY

Bitcoin uses two cryptographic primitives to secure its transactions [2]. The first primitive is the Proof of Work (PoW) protocol, and the second is the Digital Signature Algorithm (DSA). This work focused on the PoW protocol. In bitcoin, the function used is;

$H(r) = SHA256(SHA256(R))$ where; $n = 256 bits.$

The difficulty in PoW is explained thus: Suppose a message M has a target T, to locate r so that $H(m,r) \le T.$ H is model as a random function such that the algorithm to find r is to apply a brute force attack. PoW enables a miner to find a valid block. The miner increases the nonce r until a block is valid and a reward is granted. For example, using classical algorithms, the probability of finding an item of data in a bitcoin ledger is 2/3 queries; a quantum miner can apply Grover's search methods to perform such search operations in a polynomial time with less queries.

V. CONCLUSION

Hash-based signature schemes are an alternative for quantum attacks. This research critically explains the basic foundation of the three (3) hash-based digital signatures (L-OTS, W-OTS and MSS). The research proceed further to show the basic working principles of these schemes and comparing them in terms of key generation, signature generation, verification and their security levels. The comparative analysis is in table III. Based on the study, MSS is considered most suitable for bitcoin security. MSS permits a user to generate many signatures with one public key as compared to L-OTS and W-OTS. There are many variants of MSS (stateful signature scheme) and other stateless signature schemes that where not discussed in this work. Further research work can be done in this area with the aim of identifying the most suitable hash-based digital signature scheme for bitcoin cryptocurrency.

REFERENCES

[1] O. Sattath, "On the insecurity of quantum Bitcoin mining," *arXiv preprint arXiv:1804.08118*, March, 2018. Unpublished.

[2] D. Aggarwal, G.K, Brennen, T, Lee, M, Santha, and M. Tomamichel, "Quantum attacks on Bitcoin, and how to protect against them" *arXiv preprint arXiv:1710.10377*, 2017. Unpublished.

[3] J. Buchmann, E. Dahmen, and M. Szydlo, "Hash-based digital signature schemes. In *Post-Quantum Cryptography* pp. 35-93, Springer, Berlin, Heidelberg, 2009.

[4] P.W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," *In Proceedings of IEEE 35th annual symposium on foundations of computer science* pp. 124-134, November, 1994.

[5] D. Johnson, A. Menezes, and S. Vanstone, " The elliptic curve digital signature algorithm (ECDSA)," *International journal of information security*, vol. 1 no. 1, pp. 36-63, 2001.

[6] L. Lamport, "Constructing digital signatures from a one-way function" Vol. 238. [Palo Alto: Technical Report CSL-98, SRI International, 1979].

[7] A. Hülsing,, "Practical forward secure signatures using minimal security assumptions," Doctoral dissertation, Technische Universität, 2013.

[8] R. El Bansarkhani, and R. Misoczki, "G-merkle: a hash-based group signature scheme from standard assumptions," In *International Conference on Post-Quantum Cryptography,* pp. 441-463, Springer, Cham, April, 2018.

[9] A. Zeier, A. Wiesmaier, and A. Heinemann, "API Usability of Stateful Signature Schemes," *In International Workshop on Security,* pp. 221-240, Springer, Cham, August, 2019.

[10] O. Potii, Y. Gorbenko, and K. Isirova, "Post quantum hash based digital signatures comparative analysis: Features of their implementation and using in public key infrastructure," *In IEEE 4th International Scientific-Practical Conference Problems of Inforcommunications, Science and Technology (PIC S&T),* pp. 105-109, October, 2017.

[11] K. L., Grover, "A fast quantum mechanical algorithm for database search". *arXiv preprint quant-ph/9605043*, 1996. Unpublished.

[12] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system", 2008. Downloaded from: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=bitcoin+an+electronic+cash+system&btnG=

[13] L.C. Garcıa, "On the security and the efficiency of the Merkle signature scheme," *Technical Report 2005/192, Cryptology ePrint Archive,* 2005. Available at: http://eprint. iacr. org/2005/192. Unpublished.

[14] J. Buchmann, L.C. García, E. Dahmen, M. Döring, and E. Klintsevich, "CMSS–an improved Merkle signature scheme".In *International Conference on Cryptology in India* (pp. 349-363). Springer, Berlin, Heidelberg, (December, 2006).