Comparative Analysis of Classical and Post-quantum Digital Signature Algorithms used in Bitcoin Transactions

[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 Securiyt 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

[5]Okoro, Malvis Ugonna
Cyber Security Science Department
Federal University of Technology,
Minna, Nigeria
malvisokoro@gmail.com

***Abstract* - The use of public key cryptosystems ranges from securely encrypting bitcoin transactions and creating digital signatures for non-repudiation. The cryptographic systems security of public key depends on the complexity in solving mathematical problems. Quantum computers pose a threat to the current day algorithms used. This research presents analysis of two Hash-based Signature Schemes (MSS and W-OTS) and provides a comparative analysis of them. The comparisons are based on their efficiency as regards to their key generation, signature generation and verification time. These algorithms are compared with two classical algorithms (RSA and ECDSA) used in bitcoin transaction security. The results as shown in table II indicates that RSA key generation takes 0.2012s, signature generation takes 0.0778s and signature verification is 0.0040s. ECDSA key generation is 0.1378s, signature generation takes 0.0187s, and verification time for the signature is 0.0164s. The W-OTS key generation is 0.002s. To generate a signature in W-OTS, it takes 0.001s and verification time for the signature is 0.0002s. Lastly MSS Key generation, signature generation and verification has high values which are 16.290s, 17.474s, and 13.494s respectively. Based on the results, W-OTS is recommended for bitcoin transaction security because of its efficiency and ability to resist quantum computer attacks on the bitcoin network.**

***Index Terms* – Post-Quantum Cryptography, Security, Hash-based Signatures, Cryptocurrency**

I. INTRODUCTION

Information and Communication Technology has transformed the way businesses are been carried out. Computers and computing devices are been developed to be smarter and intelligent nowadays. In the same vein, networks are growing rapidly such that people are connected globally which makes communication easier and convenient for doing business. These developments led to the emergence of many online shopping websites such as e-bay. Other digital money transfer intermediaries such as Paypal have also emerged.

Bitcoin is an electronic currency that is been used today in online business. Bitcoin is a peer-to-peer (p2p) network that is manage by all the peers in the network and controlled by nobody [9]. Bitcoin cryptocurreny does not require a third party agent (such as a central authority) for its regulation. Due to its assumed privacy and anonymity, bitcoin is widely accepted digital currency. Its security relies on the application of cryptography [3]. The algorithm used by bitcoin is the Rivest Shamir Adleman (RSA) and the Elliptic Curve Digital Signature Algorithms (ECDSA).

In a p2p network, digital signatures play a very important role in making sure the transactions from the sender and the receiver are well secure. This is achieved by numerous methods such as time-stamping of trusted transactions in the bitcoin ecosystem. This protocol is efficient using classical algorithms installed on them. Research work done by [8] proved that Shor quantum algorithm is capable of breaking the security elements of ECDSA and RSA in polynomial time using large quantum computer machines. The insecurity in these algorithms prompt for urgent research on the best post quantum algorithm that could resist quantum attacks on the bitcoin network and other devices that uses ECDSA and RSA algorithms [5]. The idea of hash-based digital signature algorithms came onboard. This research considered two hash-based signature schemes- Winternitz One-Time Signature (W-OTS) and Merkle Signature Scheme (MSS), then compared them with ECDSA and RSA in terms of key generation time, signature generation time and signature verification time respectively [7].

The research is organized in this order: part II summarizes related literatures; part III, IV, V, and VI analyzed the working mechanisms of the selected algorithms. Part VII explained the system implementation; part VIII discussed the results, while part IX is the conclusion. The references cited are also listed.

II. REVIEW OF RELATED LITERATURES

Several research works have been done recently on the need to identify an alternative algorithm suitable for use in the post quantum algorithm. "Ref [3]" compared and analyzed three classical encryption algorithms (RSA, DSA and ECDSA). The comparison was in terms of key generation, signature generation and verification. The results showed that RSA algorithm has some weakness in terms of processing speed; DSA takes large amounts of CPU time, battery power, and memory computing resources. Similar research by [8] compared RSA, ECDSA, and BLISS-B used in public key infrastructure. The results showed that BLISS-B performs better and more efficient than the others. "Ref [2]" research work was on anonymous authentication based on RSA encryption. The outcome of the research showed that signature verification time was minimal, while breach of confidentiality was a major challenge. "Ref [6]" designed and implemented a topological Quantum Error correction method. The authors introduced a new class of cryptographic algorithm to be used by classical computers that can mitigate the computational power exhibited by quantum computers. However, the security analysis of the proposed algorithm was its resistance against physical attacks. In the same vein [1] introduced a Named Data Network (NDN) with the use of RSA and ECDSA digital signature schemes and compare their performance in the NDN networks. ECDSA was considered to be the best algorithm and it provided an optimal time for signing. Authors showed that key length and efficiency plays a very important role in signature generation.

III. RSA CRYPTO SYSTEM

The RSA uses arithmetic modular principle to digitally carry out the signature of a message. RSA algorithms [3] have the following four stages: key generation, key distribution, signature generation, and signature checks. The three fundamental principles behind RSA can be found as entries for *e*, *d* and *n* so it could be really hard to find *d*. Although when *e* and *n* or even *m* are identified to have an integrated exponentiation for all entries *m* (with $0 \leq m < n$): $(m^e)^d \equiv m \pmod{n}$. This modular congruity is defined by the triple bar (≡). Consequently, in some processes, the instructions of both exponentiations can be altered and the expression can be written as:

$$(m^d)^e \equiv m \pmod{n}$$

RSA key generation process

The following are the steps in creating RSA algorithm:

Stage 1. Select two (2) distinctive prime numbers *p* and *q* that are divisible by itself and only one

State 1. Due to safety resolutions, the two prime numbers *p* and *q* would be selected indiscriminately, but it should be comparable in size, with a few digits in length.

Stage 2. Determine *n* equal to $p * q$

Stage 3. For general and secret keys, *n* will be used for eigenvalues. The size is the key dimension, usually expressed in bits. The *n* component of the encryption key is made known to the public.

Stage 4. Determine $\lambda(n)$, and *'λ'* which means Carmichael's totient
function.

i. Subsequently *n* equal to p * q, $\lambda(n)$ equal to $lcm(\lambda(p),\lambda(q))$, hence *p* and *q* are indivisible by 2, $\lambda(p)$ equal to $\varphi(p)$ equal to $(p - 1)$ and similarly $\lambda(q)$ equal to $(q - 1)$. Hence $\lambda(n)$ equal to $lcm(p - 1, q-1)$.

ii. $\lambda(n)$ remains undisclosed.

Stage 5. Select a figure in such way that $1 < e < \lambda(n)$ and greatest common divisor $(e, \lambda(n))$ equal to 1; which means, *e* and $\lambda(n)$ are mutually prime.

i. *e* with a brief bit size and a miniscule large volume, the first and most common value selected for *e* is $2^{16} + 1$ equal to 65,537. The lowest (and firmest) conceivable significance for *e* is 3. Nevertheless, in some cases; this low quality of *e* has proved to be less safe.

ii. *e* is set out as a fragment of the general key.

Stage 6. Define *d* as $d \equiv e-1 \ (mod\ \lambda(n))$; *d* is the modular reverse multiplication of *e* mod $\lambda(n)$.

i. That implies, overcome *d* in the calculation $d * e \equiv 1 \ (mod\ \lambda(n))$. *d* could be calculated proficiently by means of the extended euclidean algorithm.

ii. *d* as the secret key exhibitor is made confidential.

The general key contains *mod n* and the general (encoding) advocate *e*. Hence, secret key contains the secret (decoding) advocate *d*, and must be made confidential. *p, q*, and $\lambda(n)$ necessity must be made private since *d* can be computed with them. In addition, after *d* is determined they can all be discarded.

a). RSA key distribution process

Let's assume that Charlie wishes to send a message to Eve. The dual agreed on RSA algorithm application. Eve's general key must be made known to Charlie in order to provide the cipher text to encode the information; also Eve can use her secret key to decode the information. To assist Charlie transmit his encoded information, Eve transmits her public-key (p_k) to Charlie through a dependable, but not always hidden path. Eve's secret key (*n*) is certainly not disseminated.

b). RSA encoding scheme process

Afterward Charlie acquires Eve's public-key, he could then transmit his information to Eve. In order to achieve that, Charlie changes the text (plaintext) into a digit text in such a way that $0 \leq m < n$ *b*y using a padding mechanism regarded as a redundant standard negotiated upon. He then uses Eve's general key *e* to measure the ciphertext.

$c \equiv m^e \pmod{n}$

Even if the number is large, this can be achieved with modular exponentiation reasonable speed. Charlie now sends ciphertext to Eve.

c). RSA decryption process

Eve uses her private key expected value *d* by calculation to retrieve message from the ciphertext

$c^d \equiv (m^e)d \equiv m \pmod{n}$ assumed message, Eve could retrieve the original information by applying the reverse padding mechanism.

IV. ECDSA CRYPTOSYSTEM

A. ECDSA key generation

Assuming Eve wishes to transmit an authorized text to Charlie. Firstly, Charlie and Eve have to reach an agreement on some specified factors (such as *CURVE, H, M)*. Apart from the curve and also the field equation, there is a need for a base point of prime number that is required for *H*.

TABLE I. ELLIPTIC CURVE PARAMETERS

Parameter	Meaning
CURVE	Area and calculation of the elliptic curve
H	Elliptic curve unit point. A premise on the curve which produces a large prime order group subset *m*
M	Integral order of H, which implies the component status

The order *M* of the unit point *H*, shall be prime. Certainly, it is assumed that each number that is not zero in the circle $\mathbf{Z}/n\mathbf{Z}$ are continuous, such that $\mathbf{Z}/n\mathbf{Z}$ needs to be a sector. It denotes that *M* is a necessity to be a prime.
Eve generates a pair of key, that contains a secret key number d_A arbitrarily, and carefully chosen at interval $[1, n-1]$; and a general key curve point Q_A equal to d_{AxG} .The variable x is used to indicate the point of elliptic curve multiplication by a scalar [5].

B. *ECDSA signature generation*

To encrypt a message, Eve follow the following procedures.

i. Compute e=HASH(m) (Here, HASH is a cryptographic hash function, like SHA2, that transforms the output to a whole).
ii. Let's have **z** as L_n leftmost bits of *e*, where; L_n is the bit size of the circle order *n*. (Note: **z** may be bigger than *n*, but not smaller).
iii. Choose a random integral k from cryptographic hash $[1, n-1]$.
iv. Determine the circle point $(x1, y1 \text{equal to} K \times G)$
v. Determine $r\ \textit{equal to}\ x1 \bmod n$.; *if* r *equal to* 0 repeat step iii
vi. Determine $s\ \textit{equal to}\ K^{-1}(\mathbf{Z}+r\,d_{A)} \bmod n.\ \textit{if } s \textit{ equal to } 0$ repeat step iii
i. The pair is signed `(r,s)`. Also `(r, -s mod n)` it is authorized for use.

C. *ECDSA Signature Verification*

To authenticate the signature of Eve, Charlie must have a duplicate of its public key Q_A. Charlie can validate that Q_A is an effective point on the curve point as:

- Confirm that Q_A is not equal to the unit element 0, and its synchronizes, else valid.
- Confirm that Q_A falls within the curve.
- Confirm that $n \times Q_A$ *equal to* 0

  Afterwards, Charlie takes these steps:
- Attest that r and s are digits in `[1, n-1]`. Otherwise, the signature is not accepted.
- Compute, `e equal to HASH(m)`, Where HASH is used in signature generation for the same purpose.
- Let's have **Z** to be the L_n leftmost bits of e.
- Compute $u1\ \textit{equal to}\ \mathbf{Z}^{8-1} \bmod n$ and $u2\ \textit{equal to}\ r^{8-1} \bmod ulos\, n$.
- Compute the circle points $(x1, y1)\ \textit{equal to}\ u1 \times G + u2 \times Q_A$ .If $(x1, y1)\ \textit{equal to}\ 0$, the signature is therefore invalid.
- The signature is true only when $r \equiv x1 (\bmod n)$, else invalid.

V. W-OTS KEY GENERATION PROCESS

Assuming G: {0, 1}* → G: {0, 1}^{s} represent a function that has cryptographic properties. At foremost, a factor *w* is chosen, such that $w \in N$, is selected with *t* equal to ⌈s/w⌉+⌈(⌊log$_2$ ⌈s/w⌉⌋ + 1 + w)/w⌉ is computed. The factor *w* if chosen large value will decreases the size of the signature but would rise processing period. Presently, *t* random numbers $X_1, ..., X_t \in \{0, 1\}^s$ are chosen. These arbitrary numbers are the secret keys *X* equal to $(X_1||...||X_t)$. Subsequent, in the next phase the general key '*K*' is created by calculating K_i equal to $H^{2w-1}(X_i)$ for *i* equal to 1, ..., t.

A. *W-OTS encryption process*

Let's have *m* equal to $m_1, ..., m_s \in \{0, 1\}$ remain the text to be authorized as, $X_1, ..., X_t$ the secret *id*, *w* and *t* the factors as defined in the key generation process. The text *m* is divided into ⌈*s/w*⌉ chunks $b_1, ..., b_{\lceil s/w \rceil}$ of the size ***w.*** If required, zeros from left are applied to the message. Presently b_i will be treated as the number encrypted by the corresponding chunk and calculate the ciphertext $D\ \textit{equal to} \sum_i^{[s/w]} = 2^w - bi$. The dual illustration of *D* is then divided into ⌈(⌊log$_2$ ⌈s/w⌉⌋ + 1 + w)/w⌉ blocks $b_{[s/w]+1, ...,}\ b_t$ of size *w*. *D* is amplified with the nulls from the leftward if possible. b_i is treated as the number encrypted by the chunk b_i and calculates sig_1 equal to $H^{bi}(X_i)$ for *i* equal to *1, ..., t* with $H^0(X_i)$ equal to X_i. The signature $(sig_1 \|...\| sig_t)$ of the text is the combination of all sig_1 for *i* equal to *1, ..., t.*

B. *W-OTS decryption*

To validate a signature, $'sig'$ is equal to $(sig_1 \|...\| sig_t)$ for a given text *m* is equal to {0, 1}^{s} the factors $b_1 ..., b_t$ are calculated initially. This is achieved during the generation of signatures. For *i* equal to ***1**, ..., t,*

$sig_i'\ \textit{equal to}\ H^{2w-1-bi}(sig_1)$ is designed.

$sig_i'\ H^{2w-1-bi}(sig_i)(H^{bi}(X_i))\ \textit{equal to}\ H^{2w-1}\ \textit{equal to} Y_i$

Therefore, if Y' is equal to $H(sig_i' \|...\| sig_t^i))$ equals *Y* and equal to $H(Y_1 \|...\| Y_t)$ the signature is recognized else invalid.

W-OTS Size: The signature *sig* = (sig_i ||...|| sig_t) comprises *t* chunks of sig_i. Every chunk has the size of an outcome of hash function. The signature bit length $|sig|$ is $|sig|$ = t∗s = ⌈s/w⌉+⌈(⌊log2 ⌈s/w⌉⌋ + 1 + w)/w⌉∗ s ≈ s/w. Thus, the signature length is almost contrariwise relative to the factor *w*. In every step of a Winternitz signature algorithm, the impact of a parameter *w* is now observed.

Key generation time (gen_{time}): Throughout the key generating period (*t*) arbitrary integers X_i has to be selected and $H^{2w-1}(X_i)$ need to be calculated thus, $t \approx s/w$ values X_i .Therefore;

$gen_{time} \approx s/w*(2^{w-1})*hash_{time} + s/w*rand_{time}$

$= o(2^w)*hash_{time} + o(1/w)*rand_{time}$ is the time for one hash processing and $rand_{time}$ is the time required to generate one arbitrary integer number. Hence, the key generation time is maximized in respect to the length of *w*.

Signature time (Sig_{time}) **:** Creating a signature is such that: $Sig = (sig_1 \| ... \| sig_t)$, the worth of sig_i has to be figured out $t \approx$ s/w times. To create one $sig_i = H^{bi}(X_i)$ with $bi \le 2^w - 1$ as normal $\sum_{j=1}^{w-1} 2^j / w = 2^w - 2/w$ hash processes has to be carried out. This outputs the cost of signature signing. $Sig'_{time} \approx s/w*(2^w - 2)/w^* hash_{time} = s*(2^w - 2)/w^2 * hash_{time} = o(2^w)$

Verification time (ver_{time})**:** In order to authenticate a message, Sig'_i has to be figured out t ≈ s/w times. To compute one $Sig_i = H^{2w-1-bi}$ with bi <= 2^w − 1 as usual $(\sum_{j=1}^{(w-1)} 2^j / w = 2^w - 2/w)$ hash processes has to be carried out. Therefore, verification period is the same as the signature period: $Ver_{time} = Sig_{time} \approx s*2^w - 2/w^2 * hash_{time} = o(2^w)$
From here, the optimum rate for factor *w* depends on the accessible assets. If the signature system is quick enough, we can reduce the signature duration to a minimum. However, the time of signature rises exponentially while the length of the signature drops linearly, and it is not a reason to start again to pick a too high number value of *w* [4].

VI. MERKLE SIGNATURE SCHEME (MSS)

A. *MSS key generation*

Only a relatively small amount of text with one public key can be encrypted with the Merkle Signature Scheme [4]. In order to show the number of texts to be received, there is a need to have a power of two. Such that, $N = 2^n$. The first phase of public key generation is the production of public keys *Xi* and private keys *Yi* of 2^n One-Time Signatures. For every general key *Yi*, with $1 \le i \le 2^n$, a hash value *hi* = *H(Yi)* is calculated. With these hash values h_i Merkle binary tree is built. The node of the tree is $a_{i,j}$, whereby *i* represent the level of the node. The node position is determined by the space between the node and the leaf. Thus, a leaf of the tree has level *i = 0* and the root has level *i = n*. The nodes of one level are numbered from the left to right, such that $a_{i,0}$ is to the left of the node at level *i*. In the Merkle tree, the hash values h_i equals the leafs of a binary tree, such that h_i = $a_{0,i}$. The hashing value of an inner node of the tree is given as:

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

Therefore, a tree with 2^n leafs and $2^{n+1} - 1$ nodes is constructed. The base of the tree $a_{n,0}$ is the *pubkey* of the MSS.

B. *MSS signature generation*

In order to authenticate a text t using the MSS, the sender choses a key sets $(X_i Y_i)$, places the digital signature by means of one-time signature scheme.

Foremost, the sender selects $(X_i Y_i)$ key sets that has not been used to other messages, and makes use of the one-time signature scheme to authenticate the message, creating a signature sig' and equivalent public key Y_i .To verify to that message, the recipient $(X_i Y_i)$ has in detail one of the unique key pairs, the sender merely embraces intermediary nodes of the Merkle tree so that the recipient can confirm that $(h_i = a_{0,i})$ was used to determine the public key $a_{n,0}$ at the root of the tree. The pathway in the hash tree and $a_{n,0}$ to the root known as $n+1$ nodes, call them $Ao, ..., An$ with $A0 = a0, i = H(Y_i)$ being a leaf and $A_n = a_{n,0} = public key$ being the root.
It is known that A_i is an offspring of A_{i+1} **.** To allow the recipient, compute the next node A_{i+1} given the prior, other offspring needs to be known as A_{i+1} **,** the sibling node of A_i **.** This node is called $auth_i$ **,** such that $A_{i+1} = H(A_i \| auth_i)$. Thus, n nodes $auth_0, ..., auth_{n-1}$ are desired. $A_n = a_{n,0} = p_k$ from $A_0 = a_{0,i}$ The nodes $auth_0, ..., auth_{n-1}, Y_i$ and the one-time signature sig' generally comprises a signature of N using the MSS.

$sig = (sig' \| Y_i \| auth_0 \| auth_1 \|, ..., \| auth_{n-1})$.

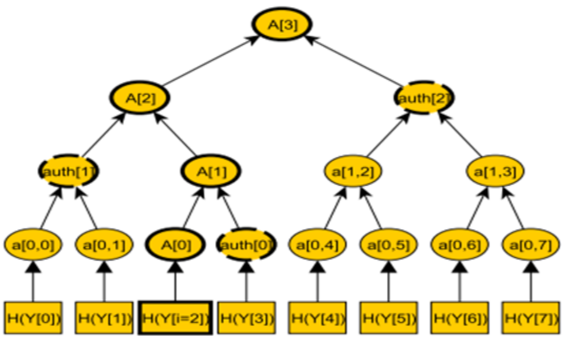


Figure 1. MSS tree with height H=3 [4]

C. To verify MSS signature

The recipient recognizes the public key, the text, and the signature as:

$Sig \text{ is equal to } (sig' \| Y_i \| auth_0 \| auth_1 \|, \ldots, \| auth_{n-1})$. In the beginning, the recipient confirms the one-time signature (OTS) ***sig'*** of text *m*. Assuming *sig'* is the true signing of *m*, the recipient calculates A_0 equal to $H(Y_i)$ by hashing the pub-key of the OTS. In figure 1, *j* equal to *1,..., n − 1,* the nodes of A_j of the path 'A' are calculated as: $A_j \text{ equal to } H(a_{j-1} \| b_{j-1})$.

VII. SYSTEM IMPLEMENTATION AND TESTING

This section presents the findings of a comparative study of four algorithms. The key sizes of ECDSA and RSA were 160, 224, 256, 1024, 2048, 302 and 4096 bits to determine the timing for key generation, signature generation and verification respectively. While the W-OTS and MSS program codes where written in C# to record the key generation time, signature generation and verification time as well. All four algorithms have been implemented using C-Sharp (C #).

Hardware/Software requirements

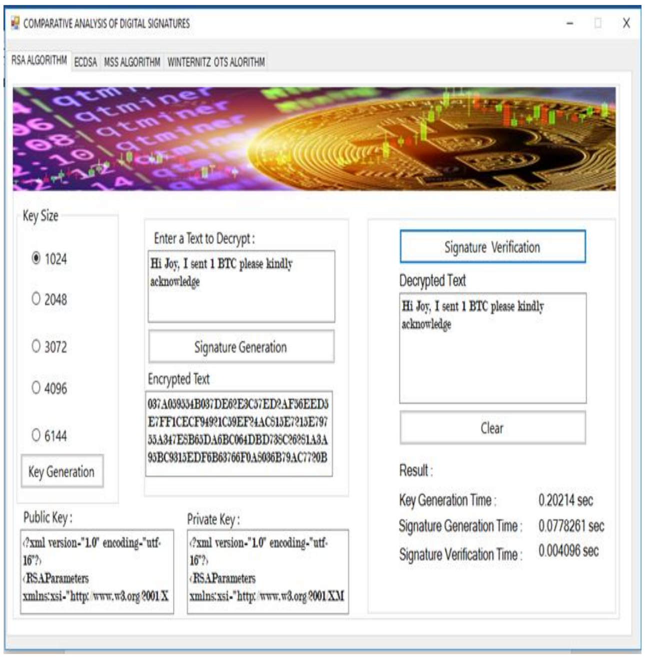
Hp Labtop 15-Bs0xx with Intel (R) core i5-6006U CPU with 2.50GHz processor; 4.0 GB of RAM

The software requirements include:

Windows 10.0 pro; Ms Visual studio .Net framework;

Programming language: C-sharp (C#). The choice of the programming language is because of its flexibility.

Figure 2 is a sample screenshot of the comparative analysis of the chosen algorithms (RSA, ECDSA, W-OTS and MSS).



Key Size: [✓ 1024] 2048 3072 4096 6144

Enter a Text to Decrypt :
Hi Joy, I sent 1 BTC please kindly acknowledge

Encrypted Text
037A039334B037DE62E3C37ED2AF36EED3E7FF1CECF94921C39EF24AC813E7213E797 33A347E8B63DA6BC064DBD738C26281A3A 93BC9313EDF6B63766F0A3036B79AC7720B

Decrypted Text
Hi Joy, I sent 1 BTC please kindly acknowledge

Result :

Key Generation Time :	0.20214 sec
Signature Generation Time :	0.0778261 sec
Signature Verification Time :	0.004096 sec

Public Key :
<?xml version-"1.0" encoding-"utf-16"?>
<RSAParameters xmlns:xsi-"http://www.w3.org/2001/X

Private Key :
<?xml version-"1.0" encoding-"utf-16"?>
<RSAParameters xmlns:xsi-"http://www.w3.org/2001/XM

Figure 2. Graphical User Interface implementation of the four algorithms

VIII. DISCUSSION OF RESULTS

The results are shown in table II for the four algorithms.

TABLE II. TIME RECORDED

Algorithm	**Key generation time (seconds)**	**Sign generation time (seconds)**	**Sign verification time (seconds)**
RSA	0.2021	0.0778	0.0040
ECDSA	0.1378	0.0187	0.0164
W-OTS	0.002	0.001	0.0002
MSS	16.290	17.474	13.494

Fig. 3 is the bar chart of the results obtained during the experiment for the four algorithms.

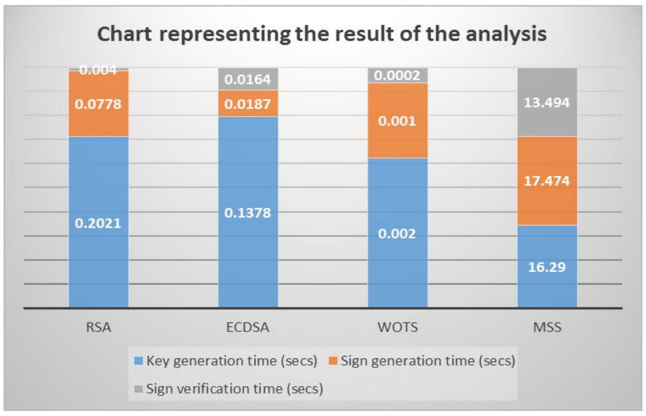


Figure 3. Bar Chart representing Time efficiency of all the algorithms

The results of the experiment are calculated and measured in seconds (s). The results as shown in table II indicates that RSA key generation takes 0.2012s, signature generation takes 0.0778s and signature verification is 0.0040s. ECDSA key generation is 0.1378s, signature generation takes 0.0187s, and verification time for the signature is 0.0164s. The W-OTS key generation is 0.002s. To generate a signature, it takes 0.001s and verification time for the signature is 0.0002s. Lastly MSS key generation, signature generation and verification has high values as compared with RSA, ECDSA and W-OTS which are 16.290s, 17.474s, and 13.494s respectively.

IX. CONCLUSION

From the research, security of ECDSA and RSA depends on the complexity of solving mathematical problems such as Discrete Logarithm Problem and Integer Factorization Problem. While the Hash-based Signature Schemes (HBSS) security depends on the hash function used. HBSS are assumed to withstand computer quantum attacks and unforgeable under chosen message attacks. The results obtained are based on the time efficiency of the algorithms selected. In all the parameters under consideration, W-OTS performs better as shown in table II. The security of W-OTS is from the property of the hash function used, and can be recommended to secure bitcoin transactions. Further research is recommended for other HBSS and their variance. Comprehensive study of different attacks on HBSS is also recommended.

REFERENCES

[1] A.A Imem, "Comparison and evaluation of digital signature schemes employed in NDN network" *arXiv preprint arXiv:1508.00184*, 2015

[2] A.D. Alrehily, A.F. Alotaibi, S.B. Almutairy, M.S. Alqhtani, and J. Kar, "Conventional and improved digital signature scheme: A comparative study," Journal of Information Security, vol. 6 no.1 p.59 August, 2015

[3] A.H. Mansour, "Analysis of RSA Digital Signature Key Generation using Strong Prime," *International Journal of Computer*, vol. *24, no.*1, (pp. 28-36), 2017.

[4] D. Butin, "Hash-based signatures: State of play" *IEEE security & privacy*, vol.*15 no.*4, (pp. 37-43) 2017.

[5] J. Breitner, & N. Heninger, "Biased nonce sense: Lattice attacks against weak ECDSA signatures in cryptocurrencies", *In International Conference on Financial Cryptography and Data Security* (pp. 3-20). Springer, Cham, (February, 2019).

[6] F. Regazzoni, A. Fowler, and I. Polian, "Quantum era challenges for classical computers," *In Proceedings of the 18th International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation* (pp. 173-178). ACM, (July, 2018).

[7] F. Pauls, R. Wittig, & G. Fettweis, "A Latency-Optimized Hash-Based Digital Signature Accelerator for the Tactile Internet", *In International Conference on Embedded Computer Systems* (pp. 93-106). Springer, Cham. (July, 2019)

[8] 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.

[9] S. Nakamoto, " Bitcoin: A peer-to-peer electronic cash system", Manubot 2019 . Download from: https://git.dhimmel.com/bitcoin-whitepaper/