Journal of Reliable Intelligent Environments
https://doi.org/10.1007/s40860-021-00155-0

ORIGINAL ARTICLE

Review and analysis of classical algorithms and hash-based post-quantum algorithm

Moses Dogonyaro Noel[1] · Victor Onomza Waziri[1] · Shafii Muhammad Abdulhamid[1] · Joseph Adebayo Ojeniyi[1]

Received: 13 February 2021 / Accepted: 3 September 2021

Abstract
Over the years, digital signature algorithms such as Rivest–Shamir–Adleman (RSA) and elliptic curve digital signature algorithm (ECDSA) are the commonly used algorithms to secure data in the public key infrastructure and other computing devices. The security notions of these algorithms relied on the difficulty of an attacker to solve the integer factorization problem used in RSA and the discrete logarithm problem in ECDSA. With the advent of quantum computers and the development of quantum algorithms, the security of data by cryptosystems are not secure. In this research, the authors carried out the review analysis of two classical algorithms (RSA, ECDSA) and hash-based signature schemes; Winternitz one time signature (W-OTS) and Merkle signature (MSS), their security strength, efficiency in terms of key generation time, signature generation and verification time. Two approaches were used: the algorithms prove of concepts which involved practical implementation of the selected hash-based signature schemes and the classical algorithms. From the results obtained and displayed in Table 8, the signature generation time of RSA and ECDSA were 0.08 ms and 0.02 ms as compared with MSS which has high values more than the RSA and ECDSA and it is 2.40 ms. The results showed that the two classical algorithms perform better in terms of the efficiency in key generation time, signature generation and verification time. However, the key generation time, signature generation and verification time increases when the key length increases. The security of the classical algorithms improved when the key length increase. Evidently an increase in signature verification time could lead to denial of service attack and quantum computer related attacks. The hash-based signature schemes in this research were considered to be the best alternative algorithms suitable for public key infrastructures considering the security properties exhibited by them. Their security depends on the hash function used and the collision resistant properties of the underlying hash function. Also the hash-based signature schemes are forward secure and uses collision resistant cryptographic hash function and a pseudorandom number generator as illustrated in Table 10.

Keywords Hash-based signature · Classical algorithms · Security · Encryption/decryption schemes

1 Introduction

This research work is an extension of the original work presented in the 2020 2nd International Conference on Computer and Information Sciences by [1]. Over the years, cryptography has been used to secure data on the internet and other information technology infrastructures. Digital encryption and public key cryptosystems uses the elliptic curve digital signature algorithms (ECDSA), Rivest–Shamir–Adleman (RSA) algorithm, and digital signature algorithms (DSA). These algorithms are referred to as classical algorithms. The security of classical algorithms is based on the difficulty in solving complex mathematical computations such as, integer factorization problem (IFP), and discrete logarithm problem (DLP). Research on post-quantum computing security has gained the attention of the research community since the development of quantum algorithm by Ref. [2] that is capable of breaking the security properties of RSA and ECDSA algorithms. It is assumed that quantum computers are designed

✉ Moses Dogonyaro Noel
moses.noel@futminna.edu.ng

Victor Onomza Waziri
victor.waziri@futminna.edu.ng

Shafii Muhammad Abdulhamid
shafii.abdulhamid@futminna.edu.ng

Joseph Adebayo Ojeniyi
ojeniyia@futminna.edu.ng

[1] Cyber Security Science Department, Federal University of Technology, Minna, Nigeria

with an increase in speed, memory and strong inbuilt cryptanalytic capabilities. This implies that the DLP and the IFP earlier considered to be hard can be reconstructed by an attacker with a sufficient large quantum computer in polynomial time. Also, the quantum search algorithm developed by Grover is capable of speeding up the IFP using a quantum computer [3].

It is anticipated that the next computing paradigm is post-quantum cryptography. This also implies that there is a need for the development of post-quantum algorithms that could replace the existing classical algorithms currently in used today. This development aroused the interest of researchers in the field of post-quantum cryptography with emphasis mostly focusing on Hash-based signature schemes. The reason might be that the security property of hash-based signatures schemes are well known and can withstand any attack originating from a quantum computer. Furthermore, hash-based signature schemes need no computationally expensive mathematical operations like large integer factorization or DLP computations. One of the most necessary requirement for hash-based signature schemes are the secure hash-function [4]. The aim of this research work is to carry out comparative analysis and review on some selected classical algorithms (Rivest–Shamir–Adleman and elliptic curve digital signature algorithm) with hash-based signature schemes (Winternitz one-time signature scheme and Merkle signature scheme) to ascertain their security properties, attack vectors, and efficiency to justify the need for the paradigm shift from classical algorithms to post-quantum hash-based signature schemes. This research work made the following contributions to knowledge:

1. A clear view and understanding on whether or not to accept the shift from classical algorithms to hash-based signature schemes
2. Broad understanding of the security parameters of the chosen classical algorithms and the selected hash-based signature schemes
3. The limitations of Classical algorithms in the post-quantum era and thereby established the need for the paradigm shift from classical algorithm to the hash-based signature scheme.

The rest of the research work is structured in this order: Sect. 2 is the review of related literatures which include the selected classical algorithms and the hash-based signature schemes. Section 3 is the methodology deployed during the research. The authors used two approaches in the methodology; algorithms prove of concepts and practical implementation of the algorithms. Discussion of results is in Sect. 4, while the conclusion and future research direction is stated in Sect. 5.

Table 1 Notations used

Notations	Description
RSA	Revest–Shamir–Adleman
ECDSA	Elliptic curve digital signature algorithm
OTS	One time signature
W-OTS	Wniternitz one time signature
LOTS	Lamport Diffie one time signature
MSS	Merkle signature scheme
IFP	Integer factorization problem
DLP	Discrete logarithm problem
FTS	Few time signature
PRNG	Pseudorandom number generator
Auth	Authentication path
I.F	Integer factorization
ECDL	Elliptic curve discrete logarithm
OWHF	One way hash function
H.F	Hash function
EU-CMA	Existentially unforgettable under message chosen attacks

Table 1 is the listed abbreviations that was used throughout in the work.

2 Review of related literatures

Several researches are ongoing in the field of classical algorithms and post-quantum cryptography. Reference [5] presents related work that focused on comparative performance analysis of RSA and ECC. The authors find time laps between encryption and decryption of the algorithms. The research was limited to classical algorithms. In the same vein [6] compared the performance of ECDSA and RSA on Name Data-link State Routing Protocol. Results from the experiment proved that RSA perform better than ECDSA in terms of speed. In 2018 [7], research work was on identifying the prominence algorithm between RSA and ECDSA. The authors reviewed the performance of RSA and ECDSA with emphasis on power consumption. On the security and efficiency of the MSS [8, 9], developed mathematical proofs of the MSS scheme. The weakness of MSS is that, there are limited number of signatures that the scheme can sign. In 2018, software implementation of MSS was carried out by Ref. [10] to investigate whether the scheme could be used to generate address signatures in mobile Internet Protocol version 6 (IPv6). The authors proved that MSS could be used to replace ECDSA because of its security strength. In the same vein, a secure compact signature scheme for the distributed ledger was proposed by [11]. The authors modeled a classical cryptocurrency using high level petri-nets to evaluate their findings. However, the proposed variance of W-OTS-S presented by the authors has large key sizes. The work of [12] focused on the comparative analysis of post-quantum hash-based signature schemes. The analysis was based on cost and performance analysis between Lamport Winternitz Merkle Scheme and SPHINCS + (a stateless hash-based signature scheme). The results showed that SPHNICS + could be a possible replacement of ECDSA in a public key infrastructure. The authors in [13] did similar work by comparing the performance of some hash functions in PHINCS signature scheme.

2.1 Significance of the research

The significance of the study is to come up with a clear distinction on the security of hash-based signature scheme over asymmetric cryptographic algorithms such as RSA and ECDSA. With the advent of quantum computers which it is estimated that by 2035, quantum computers would be in the market. The quantum supremacy over conventional algorithms would render (conventional computers) insecure [23]. The security of asymmetric algorithms can be broken by Peter Shor quantum algorithms (Shor, 1994). Also, the Grover's quantum search algorithm has a time complexity of O $(\sqrt{2^n})$ when applied on classical computers can compute the pre-image and second pre-image of an n-bit hash of a message faster. There are many cryptographic schemes that are developed to mitigate the attacks on conventional algorithms. These schemes include: hash-based signature scheme, lattice—based scheme, code-based schemes, multivariate quadratic equation cryptographic schemes, secret key isogeny schemes among others. The outcome of this study could be of benefit to the research community as the basis of developing quantum robust cryptosystems. The hash-based signature schemes where considered in this research as a good candidate in the post-quantum era because; their security properties are well understood, they do not rely on number-theoretic security or structured hardness assumptions, and are forward secure.

The outcome of this study could be of benefit to the research community as the basis of developing quantum robust systems.

2.2 Justification of the research

The research methods was in twofold: informal analysis of the algorithms and experimental analysis of the selected algorithms. The results obtained are illustrated in Table 10. The security strength of RSA and ECDSA is based on the assumption of some mathematical complexity in solving integer factorization problem and discrete logarithm problem. The security of Merkle signature scheme (a hash-based signature scheme) is proven to be existentially difficult to forge under selected message attacks. Given an output ϕ, it is nearly impossible to discover two separate inputs χ and $\chi^{|}$ such that:

$$H(\chi) = \phi \text{ and } H(\chi^{|}) = \phi.$$

This means that to forge an MSS signature the attacker has to calculate pre-image and second pre-image of the required hash data.

2.3 Classical algorithms

In this subsection, a brief explanation of the working principles of two classical algorithms (RSA and ECDSA) was done. The explanations would include algorithms formulation, encryption methodology, and security.

2.3.1 RSA algorithm description

The Rivest, Shamir, and Adleman popularly called RSA algorithm was published in 1978. It is believed to be among the first public key cryptosystem that is based on two large prime numbers factorization [14]. The algorithm encompasses the exchange of public and private keys among communicating entities to encode and decode a message send. This algorithm has three major components: key generation, encryption and decryption process. RSA working principles include:

i. *Key generation*
   In the work of [15], key generation algorithms use the following steps:
   Step 1 Chose two prime numbers α and ω.
   Step 2 Alice calculate the modulus $\lambda = \alpha \times \omega$
   (Note: λ is the modulus comprised of the public key and the private key)
   Step 3 Compute the totient:

   $$\beta(\lambda) = (\alpha - 1)(\omega - 1).$$

   Step 4 Select an integer value i which is between 1 and $\beta(\lambda)$ where : $(i, \beta(\lambda)$ are co-prime. i is publicly known.
   (The exponent of the private key P_k is calculated by Alice in a way that it fulfils the equivalence relation).
   Step 5 Bob receives Alice public key (λ, i).

ii. *Encryption part*
   To encrypt the message, Bob computes a cipher text C_t for the message δ thus:
   $C_t = \delta^i \bmod \lambda$ and send C_t to Alice.

iii. *Decryption part*
   Alice calculates δ from C_t using her private key P_k as:

   $$\delta = C_t^{Pk} \bmod \lambda \equiv (\delta^i)^{Pk} \bmod \lambda \equiv \delta^{i\,Pk} \bmod \lambda.$$

Here, it can be deduced that the security of the RSA depends on the hardness of factoring large integers. This is assumed difficult or hard using classical computers. Suppose a hacker could factor λ into α and ω, he could as well compute the private key P_k. The algorithm developed by Peter Shor in 1994 showed that factoring will not be regarded as an NP-hard problem especially when using a sufficiently large quantum computer.

2.3.2 ECDSA algorithm

The ECDSA is similar to the digital signature algorithm (DSA). It was recommended by Scott Vanstone in 1992 as a response to the request for comments from the National Institute of Standard and Technology (NIST). The algorithm was recognized in 1998 by International Standards Organization (ISO) with reference number (ISO 14888-3). In 1999, the American National Standards Institute (ANSI X9.62) also acknowledged the algorithm for use worldwide.

Like the RSA algorithm, ECDSA also consist of three process: key generation, signature generation and verification [16]. Basically the construction of a generic ECDSA as detailed in [16] is as follows: Given an integer $i \in Z$; $|r|$ represent the bit of length i. If F is a finite set, the $f \overset{R}{\leftarrow} F$ shows that an element f is chosen uniformly from F. Suppose the hash $\mathrm{HASH} : \{0, 1\}^* \to \{0, 1\}^*$ is a cryptographic hash function with a collision resistance properties. The algorithm setup takes δ as input in unary form and output an elliptic curve γ which is defined by the finite field F_T. The parameters λ_1 and λ_2 are also output of the curve γ. Hence T is considered a prime or a power of 2, and ω is a prime integer. That is; $\omega > 2^{160}$ and $\omega > 4\sqrt{T}$. $G \in \gamma$ is a point of order ω. G is the global output parameter of the elliptic curve such that: $G := (F_T, \lambda 1, \lambda 2, \omega, G)$.

i. *Key generation in ECDSA*
   In ECDSA, the key pair generation is linked with a set of elliptic curve parameters. The public key is a random manifold of the base point on the curve, and the private key is the integer that is applied to get the multiple keys. Key generation of ECDSA start by chosen a private key P_k as: $P_k \overset{R}{\leftarrow} Z^*_\omega$. The public key is a point $\beta := P_k G$. To *sign* (G, P_k, m_s); the signing algorithm would sign inputs from a global parameter which is G, and the private key P_k and the message m_s. The steps are:

Step 1 Calculate $\alpha := HASH(m_s)$.
Let i be the $|\omega|$ which is the most significant bit of α.
Step 2 Chose $\delta \overset{R}{\leftarrow} Z^*_\omega$.
Calculate $(x, y) := \delta \times G$.
Step 3 Compute $\gamma = 0$ proceed to step 2.
Step 4 Compute $f := \delta^{-1}(i + \gamma \times P_k) \bmod \omega$.
If $f = 0$, go to step 2.
Step 5 Output $\mu := (\gamma, f)$ as signature.

To verify (G, β, m_s, μ), the verification algorithm would have to input the global parameter G, then a public key β, a given message m_s and then the signature μ. This can be illustrated as follows:

Step 1 Parse μ as (γ, f).
Show that $\gamma, f \in Z_\omega/\{0\}$ if not, End.
Step 2 Compute $\alpha := \mathrm{HASH}(m_s)$.
Let i be the $|\omega|$ of the most significant bit of α.
Step 3 Calculate $j := f^{-1} \bmod \omega$,

$$b1 := f_j \bmod \omega$$
$$b2 := \gamma_j \bmod \omega$$

Step 4 Calculate $X(x1, y1) := b1\,G + b2\beta$.
If $X = 0$; reject the signature; Else, Accept, if $x1 \equiv \gamma (\bmod \omega)$.

2.4 Limitations of classical algorithms

The development of Shor's and Grover's algorithms are gradually marking an end to the traditional classical algorithms (e.g. RSA, AES, ECDSA). The Shor's algorithm solved the underlying mathematical problems of public key algorithms. In the same way, the Grover's algorithm is capable of reducing the security strength of the classical algorithms [17]. These two algorithms capabilities exposes infrastructure security architecture to quantum computer related attacks. The situation would be worst when quantum computers are fully in use and therefore compelled many researchers to ask the following questions: what are the reasons the traditional signature schemes are not capable of withstanding quantum computers?

The fact is that the speed of a quantum computer is high when compared to the classical computers. Quantum computers stem from the quantum mechanics superposition principle. Proper implementation of superposition state in a quantum computer can provide strong computing power that can break existing classical algorithms. The second question is that; what happens if all the present cryptographic secure algorithms suddenly becomes insecure? The point is that the failure of classical cryptosystems may have negative effect on all communicating devices that depends on the classical algorithms for their security. The third question is, when is such a predicament going to happen? Research conducted by the university of Waterloo showed that there is one in every seven chance of these cryptographic primitives been affected by quantum attacks in the year 2031 [18]. Lastly, with these questions enumerated, what is the way forward? To provide adequate security to all public key infrastructures, there is need to consider quantum-safe schemes that are believed to provide adequate security over quantum computer related attacks. To achieve this, the post-quantum signature schemes attract the interest of many researchers. The post-quantum signature schemes are classified into five categories: hash-based signature schemes, lattice-based, multivariate polynomial based, code-based, and the super-singular isogeny schemes. In this research, the authors considered the review of two generic hash-based signature schemes to be compared with two classical schemes (RSA and ECDSA).

2.4.1 Why hash-based signature schemes

As stated in Sect. 2.2, several methods have been proposed in the development of post-quantum signature schemes. The authors' choice of hash-based signature schemes was that the hash-based signature schemes relied on the security assumptions of the hash functions used, such as collision resistance, pre-image resistance, and second pre-image resistance properties, instead of a specific algebraic structure. In particular, hash-based signature schemes are forward secure. By forward security it means that even after a key earlier generated is compromised, all the signatures created before remain valid. In the same vein, if an attack is discovered in the hash function used, it is possible to replace it by another hash function without modifying the overall structure of the scheme. Also there is less computational expensive mathematical operations when compared with IFP and DLP computation used by classical algorithms.

2.5 Hash-based signature schemes

Hash-based Signature Schemes in recent time gained the attention of many researchers because they are considered the most suitable replacement for classical algorithms as discussed in [19]. The reason for this consideration is that, their securities are well understood, and that there is less computational expensive mathematical operations as compared with integer factorization and DLP used by classical algorithms.

Hash-based signature schemes are classified basically into two categories: stateful and stateless. A hash-based signature scheme is said to be stateful if at any point of signing a message the signature is created using the updated secret key, the signer must maintain a state that is modified at each time the signature is issued. In the case of stateless signature scheme, the scheme uses large tree-of-trees to thereby allowing it to sign many messages at a time without maintaining the state of the secret key used as it is explained in the work of [20]. At the lowermost part of the tree are a number of Few-Time-Signatures (FTS). The first step the signer needs to do is to pick a random FTS and validate it using the Merkle tree towards the root of the tree as shown in Fig. 1. This means that by using the FTS, no state is updated when generating the signature.

In this sub-section, the chosen stateful hash-based Signature Scheme would be discuss. This include: their security, key generation, signature generation and verification processes. The background formulation of stateful hash-based signature scheme is from the Lamport one-time signature scheme [20]. The signature scheme uses a one-way hash function given as: $\alpha : \{0,\ 1\}^n \rightarrow \{0,\ 1\}^n$. Where; n is a positive integer and a hash function with cryptographic properties represented as: $\omega :\ \{0,\ 1\}^* \rightarrow \{0,\ 1\}^n$.

To generate a pair of key in L-OTS, requires the following process:

Let P be the signature key that comprises of $2n$ bit strings of length n that is selected consistently at random, such that:

$$P = (p_{n-1}[0],\ p_{n-1}[1], \ldots, p_1[0],\ p_1[1],\ p_0[0],\ p_0[1]) \in R\{0,\ 1\}^{(n,\,2n)} \tag{1}$$

$$Q = \ (q_{n-1}[0],\ q_{n-1}[1],\ \ldots, q_1[0],\ q_1[1],\ q_0[0],\ q_0[1]) \in \{0,\ 1\}^{(t,\,2t)} \tag{2}$$

where:

$$q_i[j] = \alpha(p_i[j]),\ \ 0 \le i \le n-1, \quad j = \ 0,\ 1. \tag{3}$$

This means that key generation in L-OTS requires $2n$ evaluation of α. Also, the verification and signature keys are $2n$ bit strings of length n. Signature generation in L-OTS is illustrated thus:

Assuming a document to be sign is $K \in \{0,\ 1\}^*$ using the signature key P illustrated in Eq. (1).

Let $\omega(K) = \beta\ =\ (\beta_{n-1}, \ldots,\ \beta_n)$ represent the digest of the message K. L-OTS signature would be computed as:

$$\phi = (p_{n-1}[\beta_{n-1}],\ \ldots,\ p_1[\beta_1],\ p_0[\beta_0] \in \{0,\ 1\}^{(n,\,n)}. \tag{4}$$

ϕ is L-OTS signature with sequence n bit strings, each with length n. This is obtained as a function of the digest of β. To compute the ith bit string in the signature requires the signer to obtaining the $p_i[0]$. That is if the ith bit string in the digest is zero (0) and $p_i[1].[1]$. In this case, signing the message implies no evaluation of α. The signature length is therefore n^2. To verify this signature, $\phi\ =\ (\phi_{n-1},\ \ldots,\ \phi_0)$ of K as shown in Eq. (4), the verifier needs only to calculate the digest of the message $\beta = (\beta_{n-1},\ \ldots,\ \beta_0)$. After which he check whether:

$$(\alpha(\phi_{n-1}),\ \ldots,\ \alpha(\phi_0))\ = (q_{n-1}[\beta_{n-1}],\ \ldots,\ q_0[\beta_0]) \tag{5}$$

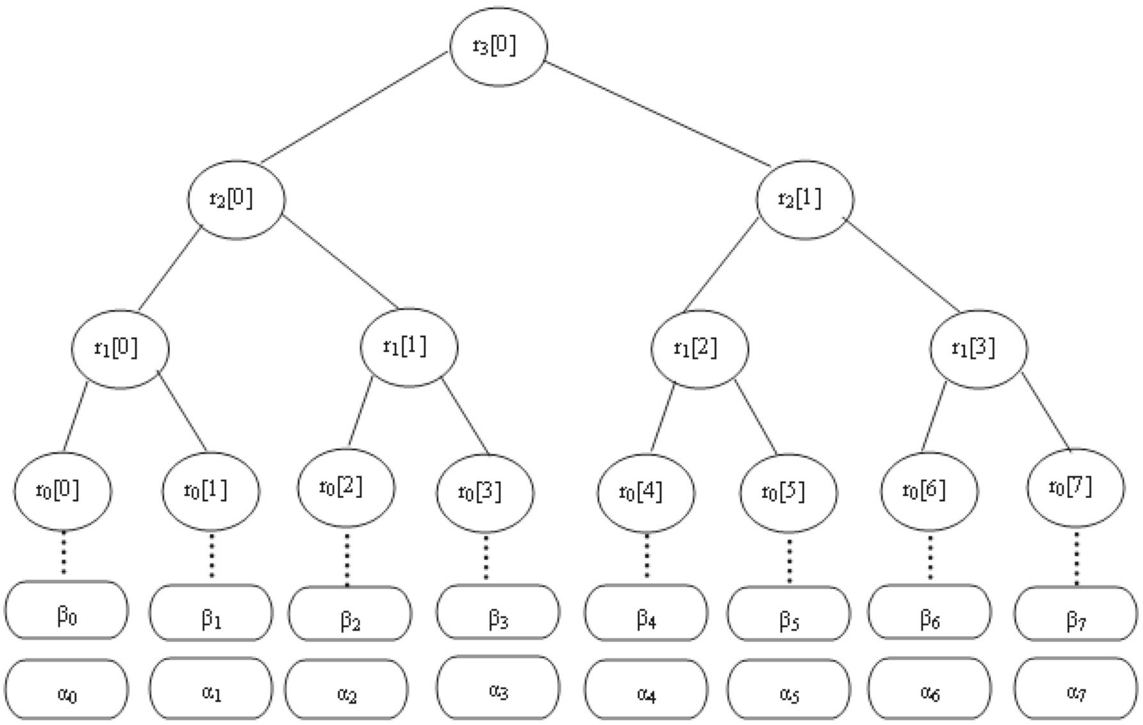


Fig. 1 Merkle tree (Height $H = 3$) [22, 23]

2.5.1 Winternitz one-time signature scheme

Raphael Merkle in 1979 improved the L-OST and named it after Robert Winternitz and this gave birth to the W-OTS [21]. As observed in L-OTS, the signature size is large but the signature scheme is secure when the message is sign once.

W-OTS would have shorter signatures if a single string of the OTS key is used to sign numerous bits in the message digest. W-OTS usage include a One-way function computed as: $\alpha : \{0,\ 1\}^n \rightarrow \{0,\ 1\}^n$ and a hash function that is cryptographically secure computed as:

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

i. *W-OTS key pair generation*
The W-OTS was first discussed in detail by [20]. To generate a Winternitz key pair, a parameter $\lambda \geq 2$ is selected. This parameter represents the number of bits to be signed simultaneously. The first bit is obtained as:

$$v_1 = \lceil n/\lambda \rceil, \quad v_2 = \left\lceil \lfloor \log_2 v_1 \rfloor + 1 + \lambda \right\rceil, \ v = v_1 + v_2. \tag{6}$$

To obtain the signature key P, then compute,

$$p = (p_{v-1},\ \ldots,\ v_1,\ v_0) \in R\{0,\ 1\}^{(n,v)}. \tag{7}$$

v_i are the bit strings that are selected at random. Thus, the confirmation key Q is obtained by applying a function α to each of the bit string in the signature key in $2^\lambda - 1$ number of times. This means;

$$Q = (q_{v-1},\ \ldots,\ q_1,\ q_0) \in \{0,\ 1\}^{(n,v)}, \tag{8}$$

where:

$$q_i = \alpha^{2\omega-1}(p_i),\ \ 0 \leq i \leq v-1. \tag{9}$$

Form Eq. (9), it means that key generation requires $v\ (2^\omega - 1)$ evaluation s of α.

Note that the length of the signature and the verification key are $v \times n$ bits.

ii. *Signature computation*
Suppose the message K with its digest $\omega(K) = \beta = (\beta_{n-1}, \ldots,\ \beta_0)$ is signed, the authors in [20] suggested that, the first step is to make sure the least number of zeros (0) are added to β such that β bits length can be divided with λ.

If there is any extension of the string β, it is separated into v_1 bit strings as $\ell_{v-1},\ \ldots,\ \ell_{v-v1}$ for a given length λ.

$$\beta = \ell_{v-1} \,||\ldots||\, \ell_{v-v_1}. \tag{10}$$

Thereafter, the bit strings ℓ_i are attached to integers in the form $(0,\ 1,\ \ldots,\ 2^{\lambda}-1)$; then a checksum θ is obtained as:

$$\theta = \sum_{i=v-v1}^{v-1} (2^{\lambda} - \ell_i). \tag{11}$$

Since $\theta \leq v_1 2^{\lambda}$. This means, in binary notation, θ should be less than,

$$\lfloor \log_2 v_1 2^{\lambda} \rfloor + 1 = \lfloor \log_2 v_1 \rfloor + \lambda + 1. \tag{12}$$

Note that it is advisable to add some zeros (0 s) bits to the binary representation such that the bit strings are divisible by λ. Thereafter the strings can be split into blocks of equal length as: $\ell_{v2-1,\ldots,}\ \ell_0$ of length λ. The checksum $\theta = \ell_{v2} - 1\,||\ldots||\,\ell_0$. The final signature K is obtained as:

$$\phi = (\alpha^{\ell v-1}(p_{v-1}), \ldots,\ \alpha^{\ell 1}(p_1),\ \alpha^{\ell 0}(p_0)). \tag{13}$$

In worst case scenario, W-OTS signature generation requires $v(2^{\lambda}-1)$ evaluations of α, hence the size of the W-OTS is $(v \times n)$. After generating the signature, the verification is done. Given the signature $\phi = (\phi_{v-1},\ \ldots,\ \phi_0$, the bit strings $\ell_{v-1},\ \ldots,\ \ell_0$ are calculated. Practical examples with proofs would be given in Sect. 3.

To confirm the signature, it is required to check if

$$(\alpha^{2\lambda-1-\ell_{v-1}}(\phi_{n-1}),\ \ldots,\ \alpha^{2^{\omega-1-\ell_0}}(\phi_0)) = (q_{n-1},\ \ldots,\ q_0). \tag{14}$$

If the signature is correct, then; $\phi_i = \alpha^{\ell i}(p_i)$. This implies that:

$$\alpha^{2^{\lambda}-1-\ell i}(\phi_i) = \alpha^{2^{\lambda}-1}(p_i) = q_i. \tag{15}$$

Equation (15) is valid for $i = (v-1,\ \ldots,\ 0)$.

At worst case, W-OTS signature verification requires $v(2^{\lambda}-1)$ valuations of α.

2.5.2 Merkle signature scheme

The MSS signature scheme as discussed in Ref. [22] is a digital signature scheme that comprised of three processes: key generation, signature generation and verification. In this scheme, a binary tree is constructed such that the OTS and the confirmation keys are referred to as the leaves of the tree, while the public key is the root. If a tree is constructed with height h and 2^h leaves, its corresponding one-time key pairs would be 2^h. Figure 1 is an example of a Merkle tree with a tree height $H = 3$. In Merkle tree in Fig. 1, α_i are the signature keys, while β_i are verification keys respectively. r_i are the nodes at different levels. The root hash is $r_3[0]$.

i. *MSS key pair generation*
To generate a key pair, the signer is required to choose the tree height $h \in N$, such that $h \geq 2$. MSS uses a hash function that is secure cryptographically and is given as: $\omega : \{0,\ 1\}^* \to \{0,\ 1\}^t$ to generate the one-time key pairs. Note that t is a positive integer. The 2^H leaves of the merkle tree is the results of the digests $\omega(\beta_i)$ of the one-time verification keys α_i. Beginning from the leaves, the verification key also referred to as the root node of the tree $rH[0]$ is generated following this process:

$$r_{h+1}[i] = \omega(r_h[2i]\ ||\ r_h[2i+1]);$$
$$0 \leq h < H,\ \ 0 \leq i < 2^{H-h-1}. \tag{16}$$

From Eq. (16) it means that the parent node generated simply by hashing the concatenation of its two child nodes. In principle, to create key pair it entails the calculation of 2^H one-time key pairs of $2^{H+1}-1$ evaluations of the hash function used.

ii. *Signature generation*
MSS uses the one-time signature keys to generate signatures [21]. Consider a Merkle signature $\phi_z(d)$ with a digest $d = \omega(K)$ of a message Kz is the signature index, ϕ_{OTS} is the one-time signature, β_z is the one-time confirmation key and an authentication path $(\mathrm{auth}_0,\ \ldots,\ \mathrm{auth}_{H-1})$. These parameters are contained in the Merkle signature $\phi_z(d)$ and its message digest K. These parameters allows the verification of the OTS with respect to the public verification key, such that;

$$\phi_z(d) = (z,\ \phi_{\mathrm{OTS}},\ \beta_z,\ (\mathrm{auth}_0,\ \ldots,\ \mathrm{auth}_{H-1})). \tag{17}$$

Note that the index z is used to decide the order of the authentication path nodes and the nodes on the path from leaf $\omega(\beta_z)$. The signature index z is such that $Z \in (0,\ \ldots,\ 2^H - 1)$. This value is increased at any time a signature is issued. Also, the OTS is applied using the α_z to generate the signature $\phi_{\mathrm{OTS}} = \mathrm{Sign}_{\mathrm{OTS}}(d,\ \alpha_z)$ of the message digest. The authentication path for the zth leaf also known as the sibling nodes is computed as $\mathrm{auth}_h,\ h \in (0,\ \ldots,\ H-1)$ on the path from the leaf $r_0[z]$ to the top root node $rH[0]$ as seen in Fig. 1. This will help the verifier to recalculate the root node of the Merkle tree so as to authenticate the current one-time signature. The authentication path at this time depends on the one-time signature verification key β_z and it is made public before the message is issued and therefore can be re-calculated by the receiver.

iii. *Signature verification*

Verification takes the following steps; Let $d = \omega(K)$ be the message digest, and $\phi_z(d)$ be the signature. The verifier compare the one-time signature ϕ_{OTS} with the OTS verification algorithm $VA_{\text{OTS}}(d, \phi_z(d))$. If the signature is true, then the receiver compute the authentication path to obtain the root node, else the signature is rejected.

3 Methodology

This section explains the steps carried out in the research work. The authors considered two approaches, the first is proof of concepts of the selected algorithms and the second approach is the practical implementation of selected algorithms. The first approach is the comparison that is based on the mathematical illustrations of the algorithms, attacks on these algorithms and their security properties. In the second approach, the efficiency (in terms of time taken to generate and verify signatures) of both the classical and hash-based signature algorithms would be implemented on a laptop computer.

3.1 Algorithms prove of concepts

3.1.1 RSA algorithm prove of concepts

The RSA cryptosystem relies on the belief that its security is the difficulty involved in factoring large integers. The algorithm involves the circulation of public and private keys between two communicating entities. The three steps in RSA algorithms include:

From Sect. 2.1 (A), to generate a key pair, Let

$$\alpha = 5 \text{ and } \beta = 7$$

$$\begin{aligned}\lambda &= \alpha \times \beta\\ \lambda &= 35\\ \therefore \quad \phi(\lambda) &= (\alpha - 1) \times (\beta - 1)\\ &= (5-1) \times (7-1)\\ &= 4 \times 6 = 24\end{aligned}$$

$$\text{GCD}(\delta, 24) = 1 \text{ and } 1 < \delta < 35$$

This implies that $\delta = 5$.

And $\delta \times v \bmod (\phi(\lambda)) = 1$

$$\begin{aligned}&= 5 \times v \bmod (24) = 1\\ &\therefore \quad v = 5\end{aligned}$$

The two key pairs are:

Let (5, 35) be the public key selected, and (5, 35) is the private keys selected. To encrypt a document, Let 3 be the document to be encrypted, Therefore, the cipher text $C_T = 3^5 \bmod (35) = 33$.

In the same way, to decrypt the message M,

$$\begin{aligned}M &= 33^5 \bmod (35)\\ &= 39135393 \bmod (35)\\ &= 33\end{aligned}$$

3.1.2 Security and attacks on RSA

The security of RSA is based on the hard computations involved in factoring very large primes, and increment in the key sizes (large key size) gives a strong security of data by the algorithm. Large key size could result in high latency and high memory consumption. This could lead to several attacks on RSA cryptosystems. These include: faulty key generation attacks, timing attacks, adaptive chosen cipher text attacks, and side-channel attacks. In the faulty key generation attacks, finding the two prime numbers p and q is achieved by testing random numbers in correct proportion and sizes. It is required that the numbers should not be too close else for better algorithm security. For example, if $(p - q) < 2n^{1/4}$ and $(n = p \times q)$ for small values of 1024 bits of n is 3×10^{77}, solving for p and q is insignificant. In the same way, if $(p - 1)$ or $(q - 1)$ has small prime factors, n can be factored out quickly by Pollard's algorithm.

3.1.3 ECDSA algorithm prove of concepts

ECDSA procedures have been explained in Sect. 2.1. In this Sect. 3.1, some mathematical prove of concepts is given which would be followed by the security of ECDSA and the common attacks associated with algorithm.

Given the following parameters:

Let the prime number $Z = 8209$

$E_z(e, f)$ is an elliptic curve with parameters $(e, f, \text{and } Z)$ where Z is an integer of the form $2^m, e = 2, f = 7, \phi = 4, 1313, h = 1\%$

ϕ is a point on the elliptic curve whose order is a large value n. Based on these parameters, the elliptic curve equation can be obtained as: $y^2 \bmod 8209 = (x^3 + 2x + 7) \bmod 8209$.

Suppose Alice and Bob, want to exchange keys, the following steps are involved:

Let $P_k A$ = Private key of Alice.

Let $P_{ub} A$ = Public key of Alice.

This implies that,

$$P_k A = 4706 \text{ (random value)}$$

$$P_{ub} A(x, y) = P_k A \times \phi(x, y)$$

$$= 4706 \times 41{,}313$$

$= 79{,}265{,}458$

Let $P_k B =$ private key of Bob.

This implies that, $P_k B = 4802$(a random number chosen by Bob).

Thai is, $P_{ub} B(x, y) = P_k B \times \phi(x, y)$.

$= 4802 \times 41{,}313$

$= 686{,}615$

To compute the secret key of Alice,

$$S_k A(x, y) = P_k A \times P_{ub} B$$

$= 47{,}706 \times 686{,}615$

$= 18{,}463{,}967$

In the same way, the secret key of Bob can be calculated as:

$$S_k B(x, y) = P_k B \times P_{ub} A$$

$= 4802 \times 79{,}265{,}458$

$= 18{,}463{,}967$

This means that both Bob and Alice have the same secret key.

i.e. $\begin{matrix} S_k(x, y) = 18463967 \\ \therefore\ 1\%\ of\ S_k(x) = 18 \end{matrix}$

Alice can encrypt her message using Bob's public key in this form:

Alice chose a message M and a random positive integer i. The cipher text of the message $M = i\phi,\ M + i P_{ub} B$.

Bob would decrypt this message using his own private key as:

Plain text $\begin{aligned} P_t &= P_t + i P_{ub} B - nB(i\phi) \\ &= P_t + i(nB\phi) - nB(i\phi) \end{aligned}$.

nB is the private key of Bob, and nA is the private key of Alice. Note that P_t is an (x, y) point encoded with the help of the message M. P_t is also the point used for the encryption and decryption.

3.1.4 ECDSA security

The main security assumption of the ECDSA is the difficulty in solving the DLP on the Elliptic curve over a finite field. This difficulty seems not to be reliable with the advent of quantum algorithms developed which could solve the DLP in polynomial time. For the ECDSA to have high level of security that is reliable in securing data, the following recommendations are necessary:

- Use of one-way hash function such that; given an output $\lambda(128 - 512\,\text{bits})$, it must be computationally infeasible to find an input α such that $\lambda = H(\alpha)$ and a collision resistant hash function must not have the probability of mapping two messages to be the same. That is $H(m1) \neq H(m2)$

Table 2 Security levels in RSA and ECDSA

Security level	RSA key size	ECDSA key size
80	1024 bits	Prime192v (192 bits)
112	2048 bits	Secp224r1 (224 bits)
128	3072 bits	Secp256r1 (256 bits)
192	7680 bits	Secp384r1 (384 bits)

Table 3 Different attacks on ECDSA

Attack type	Countermeasure
Pohlig–Hellman	Select P_k to be prime
Pollard–Rho	Select P_k such that $\sqrt{P_k}$ has some computational difficulties $P_k > 2^{160}$
Multiple logarithms	Chose P_k such that $\sqrt{P_k}$ has some computational difficulties. Minimum value of P_k should be $> 2^{160}$
Low randomness	Use of a random oracle in generating security parameters in the global space

- The key generator is unpredictable. This means that the secret key cannot be obtain by an attacker.

Table 2 is a comparison of RSA and ECDSA security levels as explained in Ref. [24]

3.1.5 Attacks on ECDSA

Common attacks on ECDSA are: quantum algorithms, Pohlig–Hellmen, exhaustive search, Pollard Rho algorithm, and baby-step giant step algorithms. Pollard Rho used to be one of the frequent attacks on the elliptic curve discrete logarithm problem (ECDLP). Pollard Rho algorithm has a running time of $\sqrt{n\pi/2}$; where n is the order of point G on the elliptic curve. Although it is possible to parallelize the running time and apply it on a different processor. This process may give rise to a new running time to be $(\sqrt{n\pi})/2r$. Where r is the number of processors used. In recent times, Peter Shor quantum algorithm is proficient to solve the ECDLP in polynomial time using a sufficient powerful quantum computer. Also, the Grover's search algorithm is a serious threat on ECDSA.

Table 3 list different attack types and their countermeasures associated with ECDSA.

3.1.6 L-OTS/W-OTS prove of concepts

In this sub-section, the L-OTS algorithm which is the basic foundation of hash-based signature scheme is discussed.

A generic L-OTS key generation, signature generation and verification process would be illustrate here with references to Eqs. (4) and (5).

Let $n = 3,\ \alpha : \{0,\ 1\}^3 \to \{0,\ 1\}^3;\ p \to p + 1 \mod 8$.

Let $\beta = (1,\ 0,\ 1)$ be the hash obtained from the message K. From Eq. (4), the signature keys are selected as:

$$P = (p_2[0],\ p_2[1],\ p_1[0],\ p_1[1],\ p_0[1]) \\ = \begin{pmatrix} 1\,0\,0\,1\,1\,0 \\ 1\,0\,0\,1\,0\,1 \\ 1\,0\,0\,0\,1\,0 \end{pmatrix} \in \{0,\ 1\}^{3,6}$$

To compute the verification key Q of the signature key of P, the steps are:

$$Q = (q_2[0],\ q_2[1],\ q_1[0],\ q_1[1],\ q_0[0],\ q_0[1] \\ = \begin{pmatrix} 0\,0\,1\,1\,1\,0 \\ 0\,0\,0\,1\,1\,1 \\ 0\,1\,0\,1\,0\,1 \end{pmatrix} \in \{0,\ 1\}^{3,6}.$$

Therefore the signature from the hash of K which is $\beta = (1,\ 0,\ 1)$ can be obtain as:

$$\phi = (\phi_2,\ \phi_1,\ \phi_0) = (p_2(1),\ p_1(0),\ p_0(1) \\ = \begin{pmatrix} 0\,0\,0 \\ 0\,1\,1 \\ 0\,1\,0 \end{pmatrix} \in \{0,\ 1\}^{3,3}.$$

The security of L-OTS is based on the fact that the signature keys must be used only once. This can be proven by considering this scenario. Using the same key twice releases some secret data for an attacker to manipulate.

Let $n = 4$(n is a security factor).

Assuming the signer wants to sign two messages applying the same signature key on these two digest $\beta_1 = (1,\ 0,\ 1,\ 1)$ and $\beta_2 = (1,\ 1,\ 1,\ 0)$. Then the signature of β_1 and β_2 are obtained as: $\phi_1 = (p_3(1),\ p_2(0),\ p_1(1),\ p_0(1)$ and $\phi_2 = (p_3(1),\ p_2(1),\ p_1(1),\ p_0(0)$. With these signatures, an hacker discerns $p_3(1),\ p_2(0),\ p_2(1),\ p_1(1),\ p_0(0),\ p_0(1)$ from the signature key generated. This information could be used by the attacker to create true signatures for the messages that has these digests $\beta_3 = (1, 0, 1, 0)$ and $\beta_4 = (1, 1, 1, 1)$ respectively.

If the hash function used is a cryptographic hash function, it would be difficult for the attacker to obtain the exact digest of the message. W-OTS improves on the L-OTS. To demonstrate the security and the improvement of W–OTS, refer to Eqs. (7) and (13) in Sect. 2.2.

Let $n = 3,\ \lambda = 2$, (λ is the Winternitz parameter). The one-way function given as:

$$\alpha : \{0,\ 1\}^3 \to \{0,\ 1\}^3,\ p \to p + 1 \mod 8$$

and $\beta = (1,\ 0,\ 0)$ is the message digest. The parameters $v_1,\ v_2$ are the total number of bits to be signed. That is:

$$v_1 = 2,\ \ v_2 = 2,\ \text{and}\ v = 4$$

The signature key can be obtain as:

$$P = (p_3,\ p_2,\ p_1,\ p_0) \\ = \begin{pmatrix} 1\,0\,0\,1 \\ 1\,0\,1\,1 \\ 1\,0\,1\,0 \end{pmatrix} \in \{0,\ 1\}^{(3,4)}.$$

To obtain the verification key, function α needs to be applied on the bit strings of P three times. This gives the verification key Q as:

$$Q = (q_3,\ q_2,\ q_1,\ q_0) \\ = \begin{pmatrix} 0\,0\,1\,0 \\ 1\,1\,1\,0 \\ 0\,1\,0\,1 \end{pmatrix} \in \{0,\ 1\}^{(3,4)}.$$

Adding one zero (0) to the let most bit of $\beta = (1,\ 0,\ 0)$, then concatenate by separating the bit string into two block length gives, $\beta = 01\,||\,00$. This helps in calculating the checksum as seen in Eq. (11).

From Eq. (11), the checksum θ is computed as: $\theta = (4 - 1) + (4 - 0) = 7$. The binary representation of $\theta = 111$.

Adding one zero (0) to the left and splitting into two block length gives $\theta = 01\ ||\ 11$. Therefore the signature as stated in Eq. (15) would be: $\phi = (\phi_3,\ \phi_2,\ \phi_1,\ \phi_0) = (\alpha(p_3),\ p_2, \alpha(p_1), \alpha^3(p_0))$. This would give a 3 by 4 matrix of the form:

$$\begin{pmatrix} 0\,0\,1\,1 \\ 0\,0\,0\,1 \\ 0\,0\,0\,1 \end{pmatrix} \in \{0,\ 1\}^{(3,4)}.$$

Applying Eq. (14) to verify the signature by computing $(\alpha^2(\phi_3), \alpha^3(\phi_2), \alpha^2(\phi_1), \phi_0)$

$$= \begin{pmatrix} 0\,0\,1\,0 \\ 1\,1\,1\,0 \\ 0\,1\,0\,1 \end{pmatrix} \in \{0,\ 1\}^{(3,4)}.$$

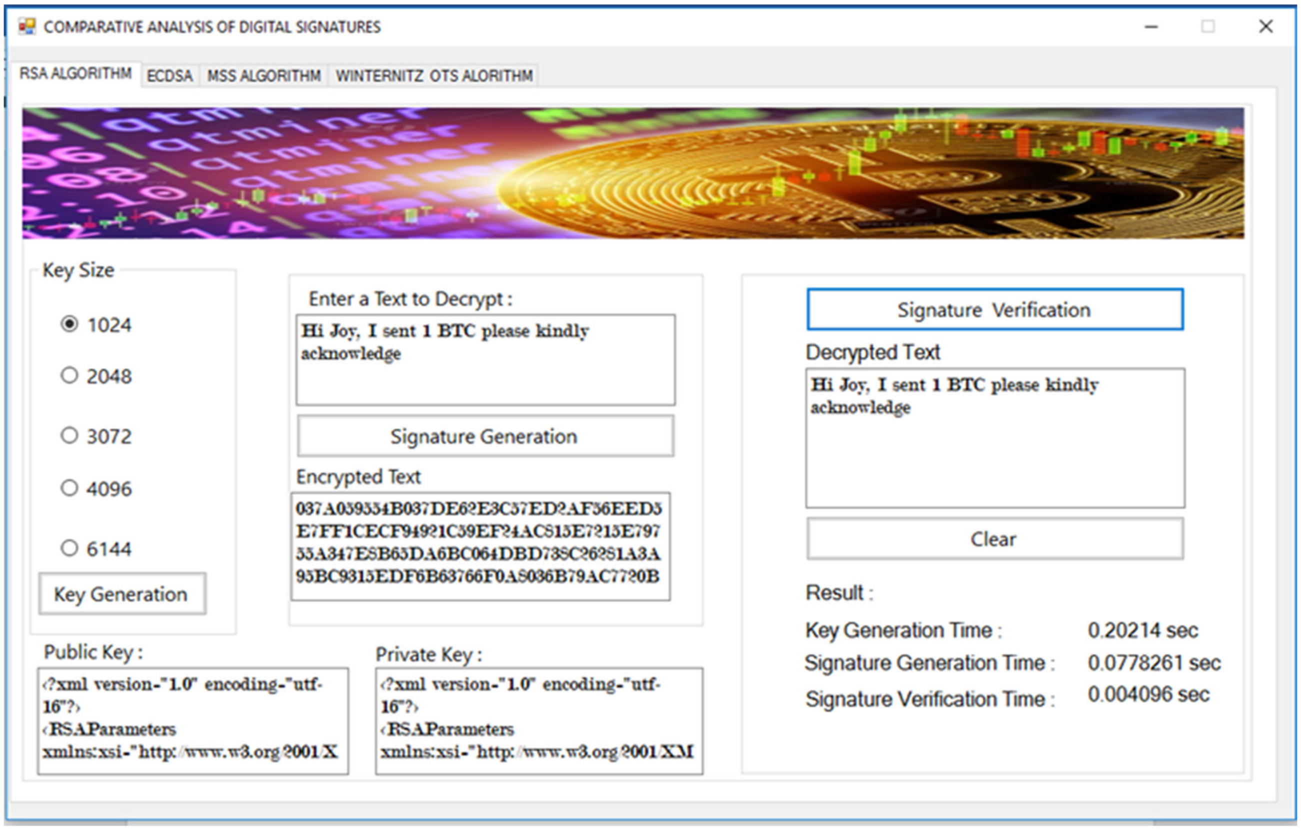
The verified signature is compared with Q which is the verification key as it is shown in Eq. (8).

3.1.7 W-OTS security

Key generation and verification procedures of the W-OTS are hard to forge with the addition of the checksum parameter. W-OTS is proven to be existentially difficult to forge under adaptive selected message attacks. This is more pronounced

Table 4 Summary of the algorithms

Criteria/scheme	RSA	ECDSA	WOTS	MSS
Math complexity	I.F	ECDL	OWHF	H.F
Algorithm complexity	(O (N^3)) Low	$Z/_nZ$	$M = \{0,1\}^k$	$N = 2^n$
Security		High	EU-CMA	EU-CMA
Decryption	High	Low	High	Low



Key Size: [✓ 1024] 2048 3072 4096 6144

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

Encrypted Text

```
037A039554B037DE62E3C57ED2AF56EED5
E7FF1CECF94921C59EF24AC815E7215E797
55A347E8B65DA6BC064DBD738C26281A3A
95BC9315EDF6B63766F0A8036B79AC7720B
```

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

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
```

Result:

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

Fig. 2 Screen shot of the G.U. I implementation

when it is used with a family of pseudorandom functions. W-OTS security also include key one-wayness and collision resistant. W-OTS one-wayness is explain as: given and output α (normally α ranges from 128 to 516) bits, it is computationally infeasible to discover another input β such that $\alpha = H(\beta)$. W-OTS collision resistant properties should be that: given an output λ, it should be hard to find two distinct inputs ω and $\omega^{|}$ such that,

$$H(\omega) = \lambda \text{ and } H(\omega^{|}) = \lambda.$$

The summary of the four (4) chosen algorithms are illustrated in Table 4 based on their mathematical complexity, algorithm complexity and security rating.

3.2 Implementation

This subsection explains the second approach to this research, which involves practical implementation of these algorithms on a laptop computer system. The findings of the comparative analysis of the four algorithms were discussed. The implementation of the four algorithms on a laptop computer is based on the mathematical formulations of these algorithms as discussed in Sects. 2.1 and 2.2. A system was developed which accept inputs from the user then the system processed the inputs to give the corresponding output results. The variable inputs used are key sizes for the RSA and ECDSA and then the output results which comprise of the key generation time, signature generation and verification time are recorded in seconds. The performance of the four algorithms (RSA, ECDSA, W-OTS, MSS) were imple-

Table 5 Key generation processing time

Key length		Time (s)	
ECDSA	RSA	ECDSA	RSA
163	1024	0.04	6.20
233	2240	0.18	7.40
283	3072	0.21	9.80
409	7680	0.57	133.90
571	15,360	1.40	678.06

mented using Java programming language. The reason for the choice Java programming language is due to its flexibility and platform independent.

3.2.1 Hardware/software requirements

The configuration of the computer system used is as follows: Hp Laptop 15 with intel® core i5, the processor speed is 2.5 GHz, and 4.0 gigabyte of random access memory. The requirements of the software include: Windows 10.0 professional, Ms Visual, .Net framework. Java + Eclipse IDE. The screen shot of the Graphical User Interface (G.U.I) is shown in Fig. 1.

4 Discussion of results

Figure 2 is the simulation environment that allows the user to input several variables of each algorithm. Different key sizes were used and the time was recorded in seconds to generate and verify the signature. The results were recorded and tabulated. The aim is to obtain key generation time, signature generation and confirmation time measured in seconds for all the algorithms. The authors seek to find the time required for each chosen parameter so as to compare the efficiency of each algorithm. Efficiency is a function of processing time among other parameters used in determining the effectiveness of a given algorithm. The lesser the time, the more efficient is an algorithm.

4.1 Performance of RSA and ECDSA

From Table 5, as ECDSA and RSA key sizes increase, there is an increase in key generation time. When RSA key size was 15,360 KB, the key generation time was highest (approx. 678 s) as compared to when ECDSA key size was 571 KB and the key generation time is approximately 1.4 s.

The graphical representation of the key generation of the algorithms as illustrated in Table 5 is shown in the bar chart in Fig. 3.

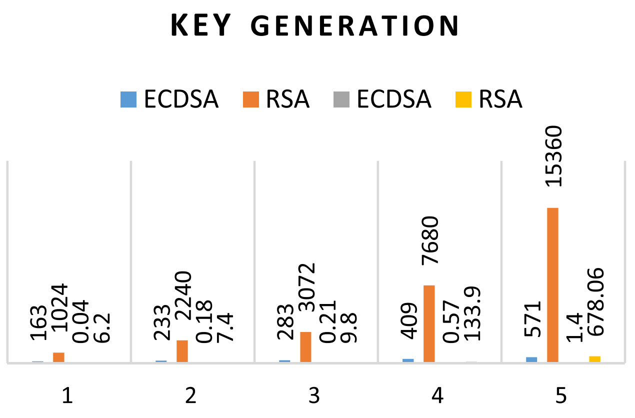


Fig. 3 Key generation processing time (s)

Table 6 Signature generation processing time

Key length		Time (s)	
ECDSA	RSA	ECDSA	RSA
163	1024	0.02	0.08
233	2240	0.33	0.17
283	3072	0.59	0.47
409	7680	1.18	1.67
571	15,360	4.53	9.08

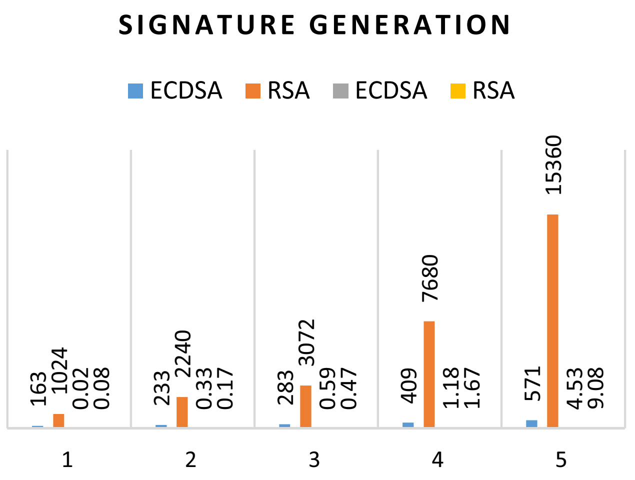


Fig. 4 Signature generation processing time

The signature generation time is shown in Table 6. In Table 6, it can be seen that ECDSA has the least time for signature generation time which is approximately 0.02 s when the first key length was selected. Although when RSA has its highest key size, the time taken to generate the signature was higher than ECDSA.

The graphical representation of the signature generation of the algorithms as illustrated in Table 6 is shown in the bar chart in Fig. 4.

Table 7 Signature verification time

Key length		Time (s)	
ECDSA	RSA	ECDSA	RSA
163	1024	0.22	0.02
233	2240	0.51	0.02
283	3072	0.76	0.02
409	7680	1.70	0.02
571	15,360	3.75	0.02

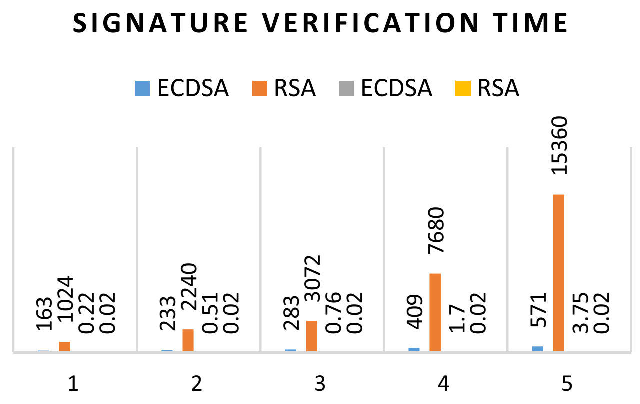


Fig. 5 Signature verification times (s)

The signature verification time is shown in Table 7. From the results obtained, the verification time of RSA was 0.02 s which was constant at all the key length selected. The time varies when using ECDSA. This implies that RSA signature verification performs better than ECDSA.

The graph representing the signature verification time is illustrated in Fig. 5. The results on the chart is based on Table 7.

Considering the results of the two classical algorithms explained in Sect. 4 (A), it can be deduced that ECDSA performs better than RSA in terms of key generation and signature generation time. However, in terms of signature verification, the results showed that RSA outperforms ECDSA. The second part of our experiment would be to write program codes for the two hash-based signature schemes (W-OTS & MSS) in which the algorithms procedures are detailed in Sect. 4 (B).

4.2 Performance evaluation of W-OTS and MSS

The algorithms for key generation, signature generation and confirmation time would be calculated. The algorithms used are illustrated here. First is the W-OTS:

Algorithm (a). W-OTS Key pair creation

```
Required variables parameters: hash function
$\omega : \{0,1\}^* \to \{0,1\}^n$, parameter $\lambda \in \aleph$ and $v_1 = \lceil n/\lambda \rceil, v_2 = \lfloor \log_2 \rfloor + \lfloor 1+\lambda \rfloor, v = v_1 + v_2$
Begin

Output: Sign key P, confirmation key Q
Step1. Select $p_1,\ldots,p_v \in R\{0,1\}$ at random
Step2. Set $P = (p_1,\ldots,p_v)$
Step3. Compute $q_1 = \omega^{2\lambda-1}(p_i)$
               for $i = 1,\ldots,v$
Step4. Q=$\omega(q_1 \| \ldots \| q_v)$ where || denotes
            concatenation
Step5. Return $(P,Q)$
      End
```

Algorithm b is the signature generation in W-OTS.

```
Algorithm b) Signature generation
System parameters:
$v = \lceil n/\lambda \rceil + \lceil (\lfloor \log_2 \rfloor \lceil n/\lambda \rceil + 1 + \lambda)/\lambda \rceil$
Begin

Input: document $\beta$, sign key $P$
Output: OTS $\phi$ of $\beta$

Step1. Calculate the $n$ bit hash value $H(\beta)$
        of the document $\beta$
Step2. Split the binary form of $H(\beta)$ into
$\lceil n/\lambda \rceil$ blocks $\ell_1,\ldots,\ell_{\lceil n/\lambda \rceil}$ of length $\lambda$,
padding $H(\beta)$ with zeros from the left if
necessary
Step3. Handle $\ell_1$ as an integer programmed
by each block and calculate the checksum
$\theta = \sum_{i=1}^{\lceil n/\lambda \rceil} 2^{\lambda} - \ell_i$
Step4. Convert $\theta$ into binary and spilt into
$\lceil (\lfloor \log_2 \lceil n/\lambda \rceil \rfloor + 1 + \lambda)/\lambda \rceil$ blocks of
$\ell_{\lceil n/\lambda \rceil} + 1,\ldots,\ell_v$ of length $\lambda$, padding $\theta$ with
zeros from the left side if needed.
Step5. Treat $\ell_i$ as an integer coded by the
blocks then compute
$\phi_i = H^{\ell i}(p_i),$
$i = 1,\ldots,v; where: H^0(p) := p$
Step6. Return $\phi = (\phi_1,\ldots,\phi_v)$
    End
```

Algorithm c is the signature verification in W-OTS.

```
Algorithm (c). Signature verification
Required parameters: hash function:
$\omega : \{0,1\}^* \to \{0,1\}^n$, parameter $\lambda \in \mathrm{N}$ and
$v = \lceil n/\lambda \rceil + \lceil (\lfloor \log_2 \rfloor \lceil n/\lambda \rceil + 1 + \lambda)/\lambda \rceil$
Begin

Input: document $\beta$, signature
$\phi = (\phi_1,\ldots,\phi_v)$, verification key $Q$
Output: VALID if signature is TRUE,
        ELSE,
        FALSE if signature is INVALID
Step1. Calculate $\ell_1,\ldots,\ell_v$ as it is stated in
        algorithm b.
Step2. Compute $\phi_i = H^{2\lambda-1-\ell i}(\phi_i)$
        for $i = 1,\ldots,v$
Step3. Calculate $\phi = H(\phi_1 \| \ldots \| \phi_v)$
Step.4 If $\phi = Q$
        Then return VALID SIGNATURE
        Else
            Return INVALID
        End
```

Furthermore the algorithms used in the Merkle Signature Scheme for key generation, signature generation and verification are illustrated here.

Algorithm d). MSS key generation

```
Parameters: tree height $h$, number of messages to be signed N, signature key $\alpha_i$, verification key $\beta_i$, cryptographic hash function $\omega$.

Begin
Step1. Select $h$
        $h \geq 2 \quad and \quad N = 2^h$
Step2. Output $2^h$ one-time key pairs
        $(\alpha_i, \beta_i)$
Step3. Calculate the leaves of merkle tree
        such that $n(0,1) = \omega(\beta_i)$ where:
        $0 \leq i < 2^h$
Step4. Compute MSS public key (root of the MSS tree)
        parent node
        $n(j,i) = \omega(n(j-1, 2i) \,||\, n(j-1, 2i+1))$
        $where: \quad 1 \leq j \leq h, \quad 0 \leq i < 2^{h-j}$
End
```

Algorithm e is MSS signature generation.

Algorithm e). MSS signature generation

```
Parameters: signature index τ, message digest $d_g$, message $K$, verification path $p$
Begin
Step1. Compute $n-bit$ digest $d_g = \omega(K)$
Step2. Generate OTS signature $\phi_{OTS}$ $of$ $d_g$
        using $\tau^{th}$ OTS signature key $\alpha_\tau$
Step3. Compute signature
        $\phi_\tau = (\tau, \phi_{OTS}, \beta_\tau, (r_0, ..., r_{n-1}))$
        Where: $auth_0, ..., auth_{n-1}$, is the authentication path for $\beta_\tau$
End
```

The algorithm f is the verification algorithm in MSS.

Algorithm f). MSS signature verification

```
Step1. Apply the verification key $\beta_\tau$ to verify $\phi_{OTS}$ of digest $d_g = \omega(K)$
Step2. Validate $\beta_\tau$ by building the path $(p_0, ..., p_h)$ from $\tau^{th}$ leaf of $\omega(\beta_\tau)$ to the origin of the merkle binary tree. Use $\tau$ and authenticate path $(auth_0, ..., auth_{h-1})$ and apply the construction
        $p_\tau = \begin{cases} \omega(auth_{i-1} \,||\, p_i), \; if \; \lfloor \tau / 2^{i-1} \rfloor \equiv 1 \bmod 2 \\ \omega(p_{i-1} \,||\, auth_{i-1}), if \; \lfloor \tau / 2^{i-1} \rfloor \equiv 0 \bmod 2 \end{cases}$
        $for \; i = 1, ..., h, p_0 = \omega(\beta_\tau)$
Step3. Output $TRUE$ if $\beta_\tau$ is success
        If $p_h = Publkey \;\; ELSE$
        $"INVALID"$
End
```

The results obtained from the implementation of the MSS algorithm (i.e. executing the key generation algorithm, signature generation and confirmation algorithms) are stated in Table 8.

To improve the security of MSS, the Merkle tree authentication scheme was combined with W-OTS scheme and the collision resistance hash function. This gives the provable security of MSS.

The number of possible signatures in MSS is 2^N. In Table 8, it is seen that the key generation time of MSS when $N = 8$ is 0.22 s. From the results in Table 5, the authors observed that as the number of signatures to be generated increases, the key generation, signature generation and verification time also increased. Table 9 is the comparison of performance of RSA, ECDSA, and MSS combined with Winternitz OTS. The reason is to summarize the performance of the selected classical algorithms with the hash-based signature scheme.

The graphical representation of the results in Table 9 is shown in the bar chart in Fig. 6. The graph is plotted time versus RSA, ECDSA and MSS.

Table 9 shows that the key generation in RSA and ECDSA are faster than the generic MSS. The same is with the sig-

Table 8 Performance of MSS algorithm

N	Key gen time (ms)	Signature gen time (ms)	Signature verify time (ms)
8	0.22	1.50	2.40
10	0.43	135.00	2.57
12	0.87	159.00	2.67
14	1.80	1.00	2.85

Table 9 Comparison of performance between RSA, ECDSA, and MSS

	RSA (1024) ms	ECDSA (163) ms	MSS + W-OTS ms
Key generation	0.20	0.04	0.22
Signature generation	0.08	0.02	1.50
Signature verification	0.02	0.22	2.40

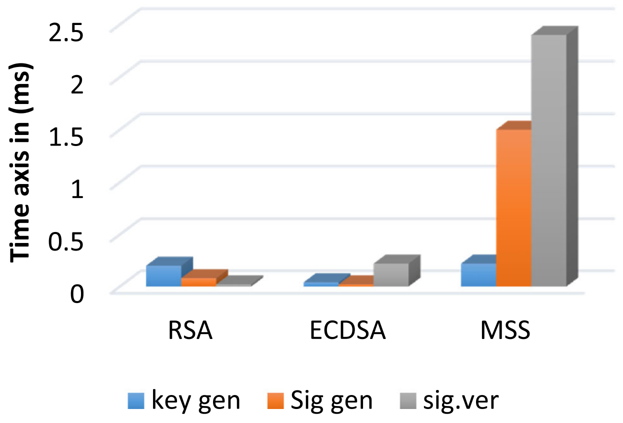


Fig. 6 Comparison performance of RSA, ECDSA, and MSS

Table 10 Comparison of the security strength of RSA, ECDSA & MSS.

Security strength	RSA key size	ECDSA key size	MSS security parameters
80	1024	160–223	Hash
112	2048	224–255	Function, $N = 2^n$, EU-CMA
128	3072	256–383	
192	7680	384–511	
256	15,360	512 above	

nature generation time of ECDSA and RSA which is faster than MSS. From Tables 5, 6, and 7, it is seen that as the key sizes of RSA and ECDSA increases, the key generation time, signature generation and verification time increases. While in Table 8, as the number of possible signatures in MSS increases, the key generation, signature generation and verification time also increases. The results in Table 9 show that RSA and ECDSA perform better than generic MSS algorithm in terms of signature generation and verification.

Table 10 represents the security strength of RSA ECDSA and MSS as obtained from 2018 National Institute of Standard and Technology.

The strength of RSA and ECDSA depends on the key size used. The security of RSA algorithm based on the assumption of the hardness involved in factoring large prime numbers. While the ECDSA is built on the principle of the difficulty of solving DLP on the elliptic curve. The development of a quantum algorithm by Peter Shor to solve the DLP and IFP of both RSA and ECDSA in polynomial time makes the two classical algorithms insecure.

The column for security of Merkle Signature Scheme which was left blank in Table 8 is hereby explained thus: MSS is proven to be existentially difficult to forge under selected message attacks, such that, given an output ϕ it is nearly impossible to discover two separate inputs χ and $\chi^{|}$ such that:

$$H(\chi) = \phi \text{ and } H(\chi^{|}) = \phi.$$

This means that to forge an MSS signature it requires that the attacker has to calculate pre-image and second pre-image of the required hashing data. In MSS algorithm, using a Pseudorandom Number generator (PRNG) for the OTS key generation makes it forward secure as long as the PRNG is forward secure. The forward security property of MSS means that all signatures issued out before a withdrawal remains valid since the actual private key can only be used to generate the one-time signature keys for the next signatures as seen in Fig. 1.

5 Conclusion

The rationale for conducting this research is to gain an understanding of the basic security strength of hash-based signature schemes over conventional algorithms. The study also revealed the efficiency of hash-based signature algorithms in terms of key generation time, signature generation and verification time. The study further gives an in-depth understanding of the limitations of conventional algorithms over the hash-based signatures schemes. The conventional algorithms such as the ECDSA and RSA from the literatures reviewed pointed out the weaknesses associated with them. One of the weakness is that there exists Grover's quantum search algorithm that can be used to find pre-images of hash functions faster than the conventional algorithms. It is estimated that the time complexity of Grover's quantum search algorithm to compute the pre-image of an n-bit hash of a message is $O(\sqrt{2^{n}})$ as compared to quadratic speedup of $O(2^n)$ obtained in conventional computers [2]. Also, Peter Shor in 1994 (Shor, 1994) showed that by using a quantum computer, there is high probability to construct an algorithm that could solve the discrete logarithm problem and integer factorization problem in polynomial time. The implication is that all cryptosystems using the conventional algorithms are at risk of attacks. Buchmann et al. [9,10] discuss the security of hash-based signature schemes with emphasis on the Merkle signature scheme. These weaknesses in the conventional algorithms prompted the need to conduct more research in the area of post-quantum cryptography. The results of this research as stated in Tables 8 and 9 showed that the Merkle signature scheme is the best candidate signature scheme that could withstand quantum computer related attacks. Considering the classical algorithms, it was found that increasing the key length would increase the security strength of the algorithm and thereby increase the time to sign and verify the signature as illustrated in Tables 3, 4, and 5. This increase in time for the signature generation and confirmation time could lead to denial of service attack. With the limitation of classical algorithms, from the results as shown in Table 8 and 9, hash-based signature schemes are considered as an alternative signature algorithms that could replace the classical algorithm based on their security properties.

5.1 Future research directions

Future research work is to consider other post-quantum signature schemes (such as lattice-based, code-based, and multivariate polynomial-based schemes), compare and evaluate these schemes with Hash-based signature schemes in terms of efficiency and security in blockchain technology and other constrain devices. The authors are currently working in reducing high latency in hash-based signature schemes for their implementation in light weight devices. Latency is as a result of large key size, signature size, and high computation time.

Funding Not applicable.

Declarations

Conflict of interest There is no conflict of interest.

References

1. Noel MD, Waziri OV, Abdulhamid MS, Ojeniyi AJ, Okoro MU (2020) Comparative analysis of classical and post-quantum digital signature algorithms used in Bitcoin transactions. In: 2020 2nd international conference on computer and information sciences (ICCIS), Al Jouf University, Saudi Arabia. IEEE, pp 1–6. https://doi.org/10.1109/iccis49240.2020.9257656
2. Pan M, Qiu D, Mateus P, Gruska J (2019) Entangling and disentangling in Grover's search algorithm. Theor Comput Sci 773:138–152. https://doi.org/10.1016/j.tcs.2018.10.001
3. Ugwuishiwu CH, Orji UE, Ugwu CI, Asogwa CN (2020) An overview of quantum cryptography and Shor's algorithm. Int J. https://doi.org/10.30534/ijatcse/2020/214952020
4. Buchmann J, Dahmen E, Szydlo M (2009) Hash-based digital signature schemes. Post-quantum cryptography. Springer, Berlin, pp 35–93
5. Mahto D, Yadav DK (2017) RSA and ECC: a comparative analysis. Int J Appl Eng Res 12(19):9053–9061
6. Perbawa MR, Afryansyah DI, Sari RF (2017) Comparison of ECDSA and RSA signature scheme on NLSR performance. In: 2017 IEEE Asia Pacific conference on wireless and mobile (APWiMob), Bandung, Indonesia, IEEE, pp 7–11. https://doi.org/10.1109/APWiMob.2017.8284007
7. Toradmalle D, Singh R, Shastri H, Naik N, Panchidi V (2018) Prominence of ECDSA over RSA digital signature algorithm. In: 2018 2nd international conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, IEEE, pp 253–257.https://doi.org/10.1109/I-SMAC.2018.8653689
8. Iavich M, Gagnidze A, Iashvili G, Okhrimenko T, Arakelian A, Fesenko A (2020) Improvement of Merkle signature scheme by means of optical quantum random number generators. International conference on computer science, engineering and education applications, vol 1247. Springer, Cham, pp 440–453
9. Buchmann J, Dahmen E, Ereth S, Hülsing A, Rückert M (2013) On the security of the Winternitz one-time signature scheme. Int J Appl Cryptogr 3(1):84–96
10. Buchmann J, Dahmen E, Hülsing A (2011) XMSS-a practical forward secure signature scheme based on minimal security assumptions. In: International workshop on post-quantum cryptography, Taipei, Taiwan, Springer, Berlin, pp 117–129
11. Shahid F, Khan A, Malik SUR, Choo KKR (2020) WOTS-S: a quantum secure compact signature scheme for distributed ledger. Inf Sci 539:229–249. https://doi.org/10.1016/j.ins.2020.05.024
12. Katz J (2016) Analysis of a proposed hash-based signature standard. International conference on research in security standardisation, vol 10074. Springer, Cham, pp 261–273
13. Karatay M, Alkım E, Gürsoy NK, Kurt M (2020) A performance comparison of some hash functions in hash-based signature. J Mod Technol Eng 5(3):234–241
14. Panda PK, Chattopadhyay S (2017) A hybrid security algorithm for RSA cryptosystem. In: 2017 4th international conference on advanced computing and communication systems (ICACCS), Coimbatore, India, IEEE, pp 1–6. https://doi.org/10.1109/ICACCS.2017.8014644
15. Abdeldaym RS, Abd Elkader HM, Hussein R (2019) Modified RSA algorithm using two public key and Chinese remainder theorem. Int J Electron Inf Eng 10(1):51–64
16. Mehibel N, Hamadouche MH (2020) A new enhancement of elliptic curve digital signature algorithm. J Discrete Math Sci Cryptogr 23(3):743–757. https://doi.org/10.1080/09720529.2019.1615673
17. Mushtaq MF, Jamel S, Disina AH, Pindar ZA, Shakir NSA, Deris MM (2017) A survey on the cryptographic encryption algorithms. Int J Adv Comput Sci Appl 8(11):333–344
18. Gyongyosi L, Imre S (2019) A survey on quantum computing technology. Comput Sci Rev 31:51–71. https://doi.org/10.1016/j.cosrev.2018.11.002
19. Suhail S, Hussain R, Khan A, Hong CS (2020) On the role of hash-based signatures in quantum-safe internet of things: current solutions and future directions. IEEE Internet Things J 8(1):1–7. https://doi.org/10.1109/JIOT.2020.3013019
20. Holmgren J, Lombardi A (2018) Cryptographic hashing from strong one-way functions (or: One-way product functions and their applications). In: 2018 IEEE 59th annual symposium on Foundations of Computer Science (FOCS), Paris, France, IEEE, pp 850–858. https://doi.org/10.1109/FOCS.2018.00085

21. Hülsing A, Rausch L, Buchmann J (2013) Optimal parameters for XMSS-MT. International conference on availability, reliability, and security. Springer, Berlin, pp 194–208
22. de Oliveira AKD, Lopez J, Cabral R (2017) High performance of hash-based signature schemes. Int J Adv Comput Sci Appl 8(3):421–432. https://doi.org/10.14569/IJACSA.2017.080358
23. Bernstein DJ, Lange T (2017) Post-quantum cryptography. Nature 549(7671):188–194
24. Fernández-Caramés TM, Fraga-Lamas P (2020) Towards post-quantum blockchain: a review on blockchain cryptography resistant to quantum computing attacks. IEEE Access 8:21091–21116. https://doi.org/10.1109/ACCESS.2020.2968985