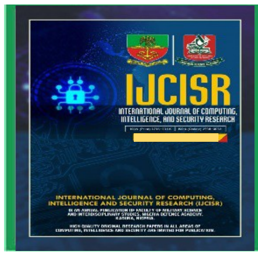


INTERNATIONAL JOURNAL OF COMPUTING, INTELLIGENCE, AND SECURITY RESEARCH (IJCISR)

Available Online @
ijcsir.fmsisndajournal.org.ng
IJCISR 3(1) 2024
Original Research

DEVELOPMENT OF AN IMPROVED WINTERNITZ HASH-BASED SIGNATURE ALGORITHM FOR BLOCKCHAIN CRYPTOCURRENCY TRANSACTION SECURITY

*M.D Noel, S. Ahmad, A.O. Isah and S.O. Subairu
Cyber Security Science Department, Federal University of Technology, PMB 65 Minna Niger State, Nigeria
*Corresponding author email: moses.noel@futminna.edu.ng

Abstract

The Winternitz One Time Signature Scheme is a Hash-based Signature Schemes that is used to secure online communicating devices. The low randomness weakness that exists in the ECDSA makes it easier for an attacker using a quantum algorithm to discover the secret key of a legitimate user. However, the major challenge of the hash-based signature algorithm is the large key and signature sizes as well as high computation time. This research work developed an improved Winternitz One Time Signature (i-WOTS) algorithm that could be used for securing cryptocurrency transactions before the transactions are added to a block in the blockchain. The i-WOTS algorithm uses the randomization technique and exclusive OR (XOR) instead of bit masking operations. The i-WOTS computation time as compared with the default algorithm was 0.73 seconds as against 5.89 seconds. This gives 87.61% improvement over the default algorithm. Comparing i-WOTS with WOTS+ and WOTS-S algorithms, the results indicate that the signature generation was 0.011 seconds which was better than the WOTS+ and WOTS-S with 0.045seconds and 0.025 seconds respectively. The same improvement was noticed in terms of signature verification time of i-WOTS which was 0.11seconds as against the verification time for WOTS+ and WOTS-S which was 0.034 seconds and 0.024 seconds. The time complexity for the i-WOTS algorithm was *O*(nd). The i-WOTS algorithm could be used on light weight devices such as IoT devices because the computation time for the enhanced algorithm will be fast with less memory consumption.

Keywords: Hash-based Signature Scheme, One Time Signature, Quantum algorithm, Quantum computer, Digital Signature Scheme

1. Introduction

Cryptography as a field of science is applied in different application areas to provide secure communication and protection of data among communicating devices. To ensure confidentiality, integrity, and authenticity of sent messages in communicating devices, digital signature algorithms and some relevant encryption schemes need to be deployed (Menezes *et al.,* 2018). The commonly used public key algorithms are the Rivest, Shamir, and Adleman (RSA) algorithm, the Digital Signature Algorithm (DSA), and the Elliptic Curve Digital Signature Algorithm (ECDSA). These algorithms are based on the hardness of prime factorization of relatively large integers. The security provided by RSA, ECDSA, and other public key digital signature algorithms that are in use by all cryptosystems are at risk of threats that are related to quantum computer attacks (Wang *et al.,* 2020).

factorization as well as discrete logarithms (Shor, 1999).

This implies that a quantum computer is capable of reconstructing the secret key provided the ECDSA public key is known. The second security threat to classical algorithms is Grover's search algorithm (Sattath, 2020). Grover's algorithm offers a great advantage to illegal cryptocurrency miners. This is because the likelihood of finding a block increases as the number of algorithm repetitions also increases when the algorithm is applied on a target system (Grover, 1996).

Several cryptographic schemes have been proven to resist quantum computer-related attacks; these include Hash-based
Cryptography, Code-based Cryptography, Lattice-based Cryptography, Multivariate-quadratic Equations, and Secret Key Cryptography (Joseph *et al*., 2022). These cryptographic schemes are alleged to repel modern computer and quantum computer-aided attacks given sufficient large key sizes (Fernández-Caramès and Fraga-Lamas, 2020). The Hash-based Cryptography is the generic term used to develop cryptographic schemes in which their security is based on the hash function that is used.

Most cryptocurrency applications
implemented on the blockchain platforms certainly relied on the ECDSA and the RSA algorithms for securing transactions on the blockchain. Nofer *et al*. (2017) explained that the security of these two algorithms is centered on the large Integer Factorization Problem (IFP) and the Discrete Logarithm Problem (DLP). The research work of (Fedorov *et al.,* 2018) revealed that the advancement in the study of quantum computing proves that quantum computers are capable of breaking many public key cryptosystems. Given these limitations, the hash-based signature algorithms were considered to be alternatives to the classical algorithms. The reason is that the hash-based signature algorithm security does not rely on the IFP or the DLP (Perin *et al*., 2021). However, the major drawback in these algorithms particularly the Winternitz One Time Signature (W-OTS) algorithm is the problem of relatively large signature sizes and high processing time which could lead to an increase in computation time (Fernández-Caramés and Fraga-Lamas, 2020). In this research, an improved Winternitz One Time Signature algorithm (i-WOTS) was developed with less computation time and reduced cost.

2. Related Literature Review

The research breaking ground in the development of quantum computers in today's Information Technology industry poses a threat to most standardized encryption schemes (Bernstein and Lange, 2017). Research works in post-quantum cryptography are still in progress to develop alternative algorithms that could resist quantum computer-related attacks such as differential side-channel attacks on the PKI. A promising alternative is the hash-based digital signature algorithms which are believed to resist quantum computer-related threats (de Oliveira *et al.*, 2017). It is in line with this that in 1989 Raph Merkle published a research article titled: A Certified Digital Signature based on encryption function (Merkle, 1989). Merkle uses the concept of a binary tree to replace many verification keys with one Public Key. The method used was known as 'tree signature'. In this method, the divide and conquer rule to authenticate signature verification was adopted. The limitation of this method was the high cost of the authentication path and high memory consumption.

Buchmann *et al.* (2007) proposed the Chained Merkle Signature Scheme (CMSS), an enhanced version of the MSS that provides significant improvement in the signature and key generation algorithms. The signing key size of the CMSS algorithm is reduced with a corresponding large signature size. The CMSS algorithm stored only the seed input for the PRNG instead of storing the whole signing key. This action caused a reduction in the size of the signing keys. The CMSS has a limitation because the signature size is large when compared to the MSS. The reason for the limitation is that the verification (*auth*) path on the sub-tree genesis node and the signature are included in the message signature.

In 2013, the Extended MSS also abbreviated as XMSS was developed. The algorithm was described as a practical forward secure signature scheme based on minimum security expectations (Buchmann, *et al.*, 2013). The authentication tree construction methods adopted by XMSS is similar to the one used by SPR-MSS. This was done for the purpose of achieving lower second pre-image resistant security needs for the hash function. When comparing the XMSS signatures size with the SPR-MSS. Buchmann *et al.* (2013) proved that XMSS is four times smaller but at a slightly better in terms of the security level. In 2015, the authors improve on the XMSS algorithm and rename it as XMSS+. The improvement was in the time taken to generate the keys which was from $O(n)\ to\ O(\sqrt{n})$ using big O notation representation. However, testing the algorithm smart card, security level of XMSS+ was 71 bits. With the XMSS+ it is feasible to apply hash function that is more than 128 bits digest length to improve its (Hülsing, *et al.,* 2018).

The research of (De Oliveira and Lopez, 2015) and (de Oliveira *et al.,* 2017) focuses on the improvement of HBSS by using parallel optimization techniques on XMSS key generation, signature verification and execution time. The scheme has limitation on the number of signatures it can generate. Pauls *et al.* (2019) research work investigated the compatibility of XMSS with low-latency needs of the tactile internet. The actors designed an XMSS-specific accelerator by combining hardware-software co-design method. The experimental results for the signing and verification was 176µs and 42µs which is 63% better as compared with the work of Wan *et al.* (2020).

To reduce the hash tree construction time (Shahid *et al.*, 2020) developed a Smart Digital signature scheme (SDS) for distributed ledgers. The SDS incorporated a unique OTS scheme in XMSS which showed a significant decrease in hash-tree creation time in comparison with the default XMSS. The authors used a key compression technique to achieve better results in computation time for the XMSS tree hashing algorithm. Kearney *et al. (2021)* carried out a comparative analysis of the vulnerability of some selected blockchain-based cryptocurrency technologies susceptible to quantum attacks. The results of the analysis showed that 95% of blockchain-based cryptocurrencies today are vulnerable to quantum computer attacks. The exposure of quantum computer-related attacks calls for urgent research in the field of blockchain application technology.

Bos *et al.* (2021) presented an improvement in the of XMSS when applied on constrained devices. The authors applied the W-OTS tuning technique proposed by (Perin *et al.,* 2018) which involved a slight change in the checksum computation and the application of a cryptographic nonce to a message and then hashed. The idea is to append a cryptographic nonce to the already signed messages before the signature is generated. To reduce the cost of verifying signatures, the procedure is repeated severally. Although significant results were obtained, the limitation of this approach is the additional cost of searching the integer used in finding the cryptographic nonce that speeds up the W-OTS algorithm. Perin *et al.* (2021) did similar work and the authors proposed the WOTS-CS. In this research work, a new set of parameters that caused a decrease in the number of hash processes specifically in the key generation was introduced. The WOTS-CS key generations were reduced by 11.37% less than the WOTS+ scheme. The parameter selection problem and the complexity that exist in the constant-sum encoding algorithms is a major limitation of this algorithm. Based on the related literature reviewed, it can be seen that the majority of the research work aimed at improving the efficiency of the algorithm. There are several research done aim at improving the Winternitz One-Time Signature scheme. Some of these researches focuses on enhancing the security, efficiency and practical deployment of the WOTS. Hulsing et al. (2018) improved the security of WOTS. The improved version was called WOTS+. The authors achieved this by using different chaining techniques for the hash function making it resistant to multi-target attacks. In the same vein, the work of (Bernstein *et al.*, 2017) provides a comprehensive security analysis of the WOTS to know how its strength and weaknesses. However, the work of (Perin *et al.*, 2018) concentrate on the redunction of the cost in signature verification as a result of the number of chained iterations of the cryptographic hash function. Oraei and Dehkordi (2022) proposed an improved WOTS based on graded encoding techniques. Similarly (Shahid *et al.*, 2020) developed an improved version of WOTS called WOTS-S. The improved version of the WOTS was applied on Distributed Ledger Technology systems. After reviewing some of the research works aimed at improving, there are still need for further improving on the WOTS to make the scheme more efficient and with less time taken during execution particularly on IoT devices. In this research, an improved Winternitz One Time Signature algorithm was developed with reduced execution time of the key generation, signature generation, and verification.

3. METHODOLOGY

The development of an improved WOTS starts with mathematical formulation of the algorithms. The algorithm parameters are the Key generation, Signature generation and Signature verification. This will be followed by the implementation process. In the implementation, the secure hash algorithm (SHA 256) was used to attain a post-quantum security level. In as much as the generic Winternitz OTS uses a collision-resistant hash function, the i-WOTS adopts the Pre-image Resistant Hash Function (PRHF) in the implementation. The reason for adopting the PRHF is that the pre-image-resistant hash function is assumed to be more efficient in withstanding multi-target attacks as compared to the collision-resistant hash function used in the generic W-OTS.

The i-WOTS uses a randomization technique with exclusive OR (XOR) operations during the signature generation processes. The exclusive OR (XOR) operations have a negligible cost on the signature size and signature verification while maintaining the same security standard. The use of randomization and XOR is to enhance un-detectability in case of any brute force attacks by an adversary. Unlike the default W-OTS algorithm and other variance such as the WOTS+ that adopts the use of bit masking as a security improvement measure, the i-WOTS algorithm does not apply bit masking in the development of the algorithm because bit masking operations comes with an additional cost of computation.

Other improvement method was to change the computation of the checksum by padding unused bits with ones instead of zeros and then applying the bit compression technique. Padding with ones improved signature verification time, while padding with zeros improved signature generation. The padding operations carried out in this work are a preventive measure such that an attacker cannot know the exact length of the plaintext message used. The *i-WOTS* used Uniform Transformation Format (UTF-8) encoding technique. The eight (8) means the 8-bit string is used at every message encoding. The UTF-8 encoding offers valuable properties that can reduce the costs related to the number of chaining function iterations of the developed algorithm and reduce key generation costs.

4. Algorithm Formulation

The i-WOTS algorithm uses the function given as: $f:\{0,1\}^s \rightarrow \{0,1\}^s$ and it is has the properties of one-wayness . The algorithm also need a cryptographic hash function $Hc:\{0,1\}^* \rightarrow \{0,1\}^s$.

Let $w \geq 2$.

Such that:

$$r1 = \left\lceil \frac{s}{w} \right\rceil;\ r2 = \left\lfloor \frac{\log_2 r1(w-1)}{\log_2 w} \right\rfloor + 1$$

Where: r = r₁ + r₂ (4.1)

The variable ‘r’ in the equation 4.1 is number of n-bytes string of i-WOTS secret key, publi~~ck~~ key and signature.

The variable *‘s’* is the hash value of the message, and ‘w’ is the winternitz parameter (w >1)
The signature key α is presented as:

$$Y = (y_{r-1},...,y_1,y_0) \in_R \{0,1\}^{(s,r)} \qquad (4.2)$$

Let y_i be the bit string selected evenly at random. This implies that the key for the verification Q is calculated by using function *f* on every bit sequence in the signature key for $(2^w - 1)$ number times. Therefore, the verification key Q is given as:

$$Q = (q_{r-1},...,q_1,q_2) \in \{0,1\}^{(s,r)} \qquad (4.3)$$

Where:

$$q_i = f^{2w-1}(y_i);\ 0 \leq i \leq r-1 \qquad (4.4)$$

The equations 4.1, 4.2 and 4.3 are the equations that are required in the key generation process. To compute the key generation, $r(2^w - 1)$ evaluations of the function *f* is required.

A). Signature generation

Let G equal to the message digest from $Hc(G) = d = (d_{s-1},...,d_0)$ is needed to be signed. The digest *d* may be padded with ones for ease of division by W. The string of *d* is sliced into *r₁* bit strings of $b_{r-1},...,b_r - r_1$ with the length to be the value of ‘W’. Then,

$$d = b_{r-1} \| b_{r-r1} \qquad (4.5)$$

The strings b_i are recognized with integers in the set $\{0,1,...,2^w - 1\}$ and the checksum is computed as:

$$Cs = \sum_{i=r-r1}^{r-1} (2^w - b_i) \qquad (4.6)$$

If the checksum $Cs \leq r_1 2^w$ implies that the checksum length in binary representation is less than taken the floor of $\lfloor \log_2 r_1 2^w \rfloor + 1 \ \ equals\ \ \lfloor \log_2 r_1 \rfloor + w + 1$ (4.7)

The equation 4.6 is padded with zeros so as enable the length of the extended string to be divided easily by ‘W’ equally. The next step is the splitting of the lengthy string into *r₂* blocks as $b_{r2-1},...,b_0$ of length *w;* such that: $Cs = b_{r2-1} \| .... \| b_0$. The signature of message ‘G’ is calculated as:

$$\alpha = (f^{br-1}(y_{r-1}),...,f^{b1}(y_1),\ f^{b0}(y_0)) \quad (4.8)$$

The signature generation requires $r(2^w - 1)$ evaluations of the function *f.*

Signature Verification

To verify the signature α , it is computed as: $\alpha = (\alpha_{r-1},...,\alpha_0)$. The bit strings in binary for $b_{r-1},...,b_0$ are computed by the verifier. The applying F, gives:

$$f^{2w-1-br-1}(\alpha_{s-1}),...,f^{2w-1-b0}(\alpha_0)) \overset{?}{=} (q_{s-1},...,q_0). \quad (4.9)$$

The signature is valid if the comparison holds in equation 4.35 or it is rejected.
The equation 4.35 holds for $i = r-1,...,0.$
Also note that to verify the signature, it requires $r(2^w - 1)$ evaluations of the function *f.*
The equations 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, and 4.9 are the basic mathematical equations in the formulation of the i-WOTS algorithm.

The flowcharts for the formulated i-WOTS algorithm is illustrated in the figures 1, 2, and 3. The flowchart in the figure 1 provides i-WOTS key generation process to obtain the public key and the secret key. The flowchart in figure 2 provides a clear steps from hashing the message to generate the final signature, while the figure 3 is the flowchart that provides i-WOTS signature verification process from hashing the message to verify the derived public key against the given public key.

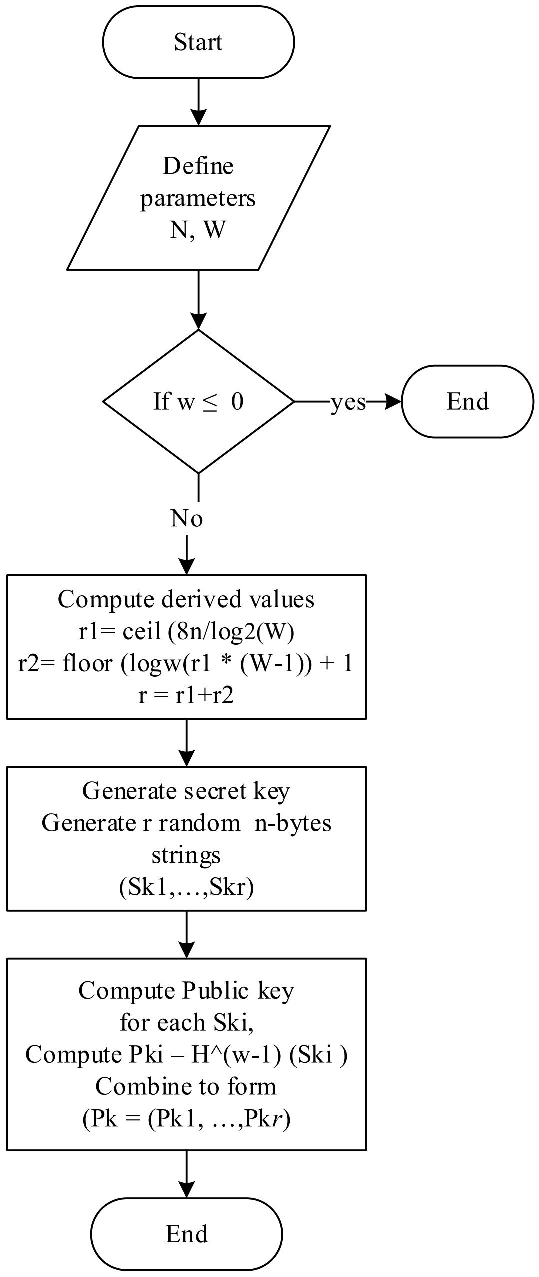
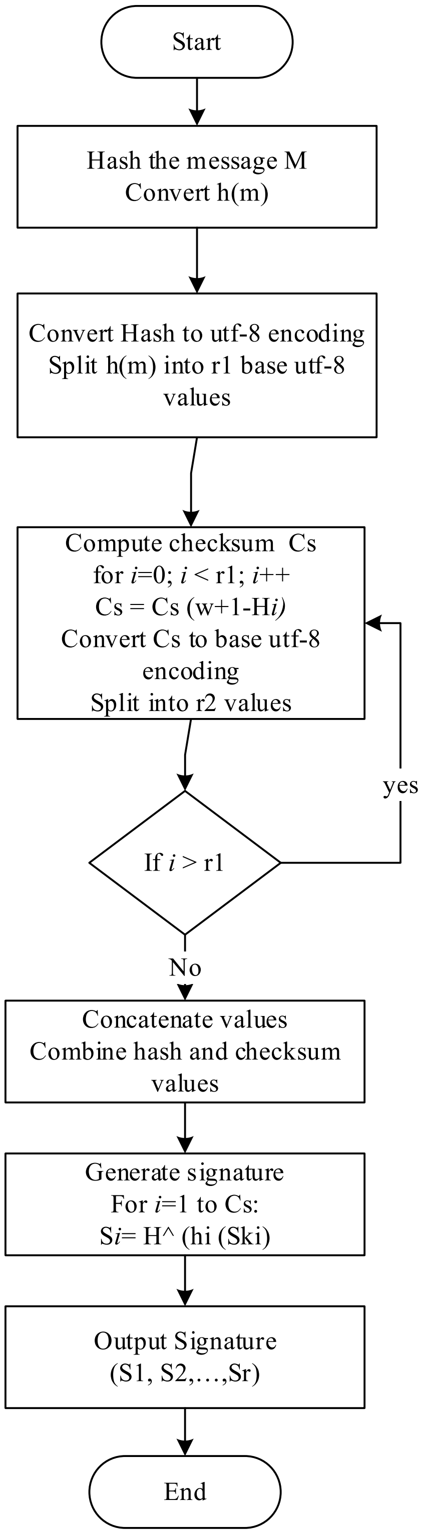

Start
Define parameters N, W
If w ≤ 0
yes
End
No
Compute derived values r1= ceil (8n/log2(W) r2= floor (logw(r1 * (W-1)) + 1 r = r1+r2
Generate secret key Generate r random n-bytes strings (Sk1,…,Skr)
Compute Public key for each Ski, Compute Pki – H^(w-1) (Ski) Combine to form (Pk = (Pk1, …,Pkr)
End

Figure 1. Key generation flowchart


Start
Hash the message M Convert h(m)
Convert Hash to utf-8 encoding Split h(m) into r1 base utf-8 values
Compute checksum Cs for i=0; i < r1; i++ Cs = Cs (w+1-Hi) Convert Cs to base utf-8 encoding Split into r2 values
yes
If i > r1
No
Concatenate values Combine hash and checksum values
Generate signature For i=1 to Cs: Si= H^ (hi (Ski)
Output Signature (S1, S2,…,Sr)
End

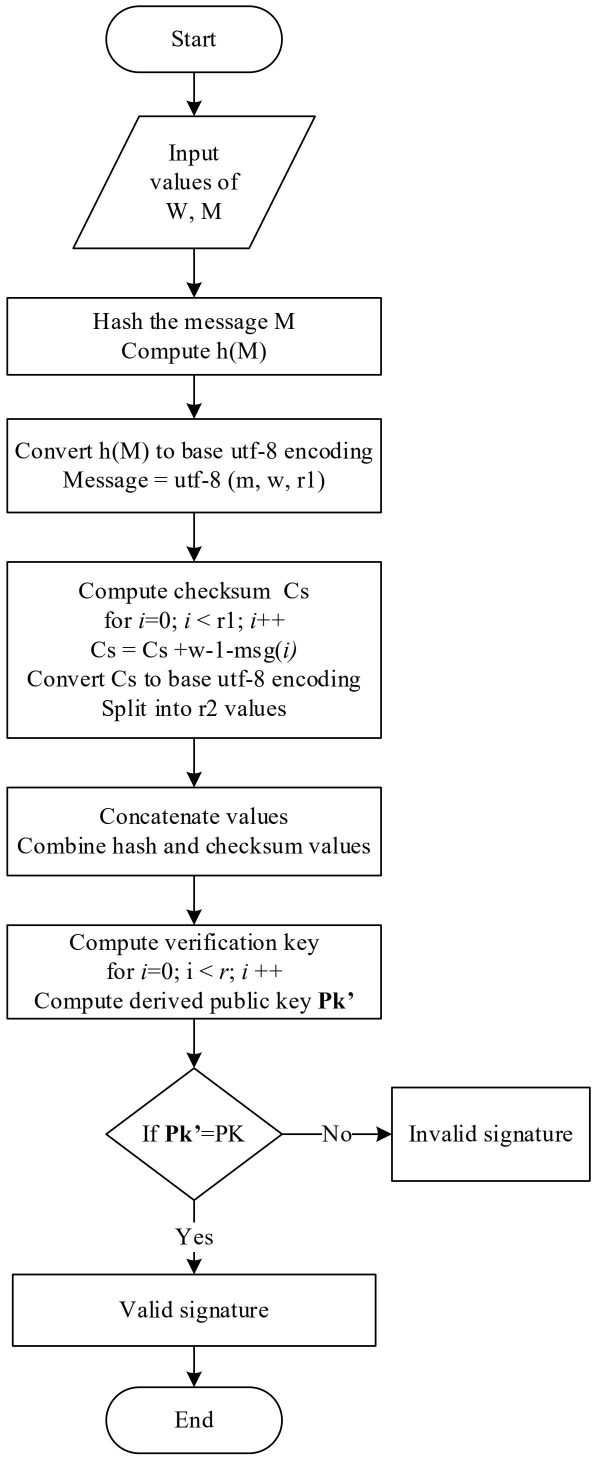


Figure 3 Signature verification flowchart

5.0 i-WOTS Algorithms

5.1 Secret Key Algorithm

The enhanced algorithm secret key is a collection of *n-byte* string lengths. The secret key is used to sign a single message at a time. The *n-byte* string should be randomly selected from the even spread or can be selected using a secure pseudorandom technique (i.e. using a pseudorandom number generator).

The algorithm 5.1 describes the secret key generation. The inputs include the seed and the i-WOTS parameter. The seed is a pseudorandom number generated using the PRF generator. The i-WOTS parameter (w) must be greater than 1. In the experiment, the value of w chosen was eight (8) and sixteen (16). The choice of these two values is to compare which among them gives the best output in terms of the time taken in key generation. The algorithm 3.1 is to return the i-WOTS secret key

Algorithm 5.1 i-WOTS_genS$_k$ Generating i-WOTS Secret key

```
Input:   Seed, i-WOTS parameter (w > 1)
Output:  i-WOTS secret key
    $for\ (i = 0;\ i < r;\ i++)\ \{$
    $initialize\ P_k(i)\ with\ a\ unifiorm\ random\ n-byte\ string;$
    $\}$
    $return\ S_k;$
```

5.2 Public Key Algorithm

The key pairs of the i-WOTS algorithm comprise *r* hashed chains of length w (w > 1). The value of *r* is computed as given in equation 4.1. The public key (pk) comprises of the last node of the hashed chains. The public key is a length of *r* collection of *n-byte* strings. The chaining function is applied to compute the hash chain. It is required that the hashed address of the OTS with the *seed SEED* should be made available by an algorithm. The *seed* used here is available to the public for the verifier to access. The algorithm 5.2 computes the public key. The inputs in this algorithm are the secret key, the address (ADRS) that is generated by the algorithm, and the seed. The secret key is to be kept secret and not to be known by the public.

Algorithm 5.2 i-WOTS_*genP$_k$* – Generating a i-WOTS PK from SK

```
Input:   i-WOTS secret key Sk, address ADRS, seed SEED
Output:  i-WOTS public key Pk

    $for\ (i = 0;\ i < r;\ i++)\ \{$
    $ADRS.setchainAddress(i);$
    $Pk(i) = chain\ (Sk[i], 0, w-1, SEED, ADRS);$
    $\{$
    $return\ Pk;$
```

5.3 Signature Generation Algorithm

The i-WOTS Signature generation is presented in algorithm 5.3. The inputs required in algorithm 5.3 include the message (m) to be hashed, the secret key and the public key are generated in algorithm 5.1 and 5.2, and the address (ADRS). The output of the algorithm is the i-WOTS signature. The operations executed by the algorithm are explained as comments lines.

Algorithm 5.3 i-WOTS_Sign

Input: Message (M), Sk.seed, pk.seed, ADRS
Output: i-WOTS Signature (Sign)

```
$Compute\ r\ from\ equation\ 4.1;$
  $m = 256,\ s = 256$
$//Convert\ message\ to\ utf\_8\ encoding$
$msg = utf\_8(m, w, r1)$
$//\ Compute\ checksum\ as\ in\ equation\ 4.6$
      $for\ i = 0;\ i < r1;\ i++$
$//\ Append\ 1\ to\ the\ computed\ checksum\ if\ the\ division\ does\ not\ give\ equal\ number\ of\ bits\ to\ be\ split$
$Cs = Cs + w - 1 - msg(i);$
$//\ Convert\ Cs\ to\ utf\_8\ encoding$
$Cs = Cs << (8 - ((r2 * \log(w))\ \%\ 8));$
$r2 - bytes - ceil((r2 * \log(w)\ /8);$
$msg = msg \| utf - 8\ (to\,byte(csum, r2\_bytes), r2);$
       $for\ i = 0;\ i < r;\ i++)$
$ADRS.setchainAddress[i];$
$Sk = PRF\ (Sk.seed, ADRS);$
$Sign\ [i] = chain(Sk, 0, msg[i], pk.seed, ADRS);$
$return\ \ Signature$
```

5.4 Signature Verification Algorithm

The algorithm 5.4 is the i-WOTS verification algorithm from the given message M and with the signature. In the algorithm, the inputs include the message with the secret seed value, the public seed, and lastly the address. The output is the i-WOTS signature. The operations explaining each operation command are written as comments within the algorithm for clarity.

Algorithm 5.4 i-WOTS_Verification

Input: Message (M), Sk.seed, pk.seed, ADRS
Output: i-WOTS Signature (Sign)

```
$w = 16$
$m = 256$
$compute\ r\ from\ equation\ 4.1$
$//\ message = SHA256(message)$
$//Convert\ message\ to\ utf-8\ encoding$
$msg = utf-8\ (m, w, r1)$
$//\ Compute\ checksum\ (Cs)\ using\ equation\ 4.6$
    $for\ i = 0;\ i < r1;\ i++$
$Cs = Cs + w - 1 - msg[i];$
$//\ Convert\ Cs\ to\ utf-8\ encoding$
$Cs = Cs << (8 - ((r2 * \log(w)) \%\ 8));$
$r2\_bytes = ceil((r2 * \log(w) / 8);$
$msg = msg || utf-8\ (to byte(Cs, r2\_bytes), w, r2);$
    $for\ i = 0;\ i < r;\ i++$
$ADRS.setchainAddress(i);$
$tmp[i] = chain(Sig[i], msg[i], w - 1 - msg[i], Pk.seed, ADRS);$
$eWOTSPkADRS.setType(eWOTS\_Pk);$
$eWOTSPkADRS.setKeyPairAddress(ADRS.getKeyPairAddress());$
$//\ check\ if\ decrypted\ value\ matches\ public\ key\ value$
$Pk\_Sig - T\_r(Pk.seed, eWOTSADRS, tmp);$
    $return\ Pk.Sign$
```

6. RESULTS AND DISCUSSION

The implementation results of the improved algorithm is discussed in this section. The computation time of the three parameters would be discussed. The Winternitz parameter "W" chosen is 16, while the hashing algorithm is SHA256. The Table 1 is the computation time of the improved algorithm as compared with the default algorithm (WOTS). In the table 6.1, the key generation time for the default algorithm was 5.77 seconds as against 0.711 seconds of the improved algorithm. In the same way, the signature generation of the default algorithm was 0.064 seconds while the improved algorithm recorded o.011 seconds. The verification time for the default algorithm was 0.057 seconds, while the verification time for the improved algorithm was 0.011 seconds. The overall total computation time for the improved algorithm was 0.73 seconds as compared with the default algorithm which was 5.89 seconds. The results shows that the i-WOTS algorithm recorded 87.61% as compared with the default algorithm.

Table 1 Algorithms Computation Time when w = 16 and Hashing Algorithm = SHA256

Parameter	Algorithm parameters	Default algorithm	Improved algorithm
W = 16 SHA256	Key generation	5.77	7.11×10^{-1}
	Signature generation	6.4×10^{-2}	

			1.1×10^{-2}
	Signature verification	5.7×10^{-2}	
			1.1×10^{-2}
Total execution time in (seconds)		**5.89 seconds**	7.3×10^{-1} **seconds**

The graphical representation of the Table 1 is the Figure 4

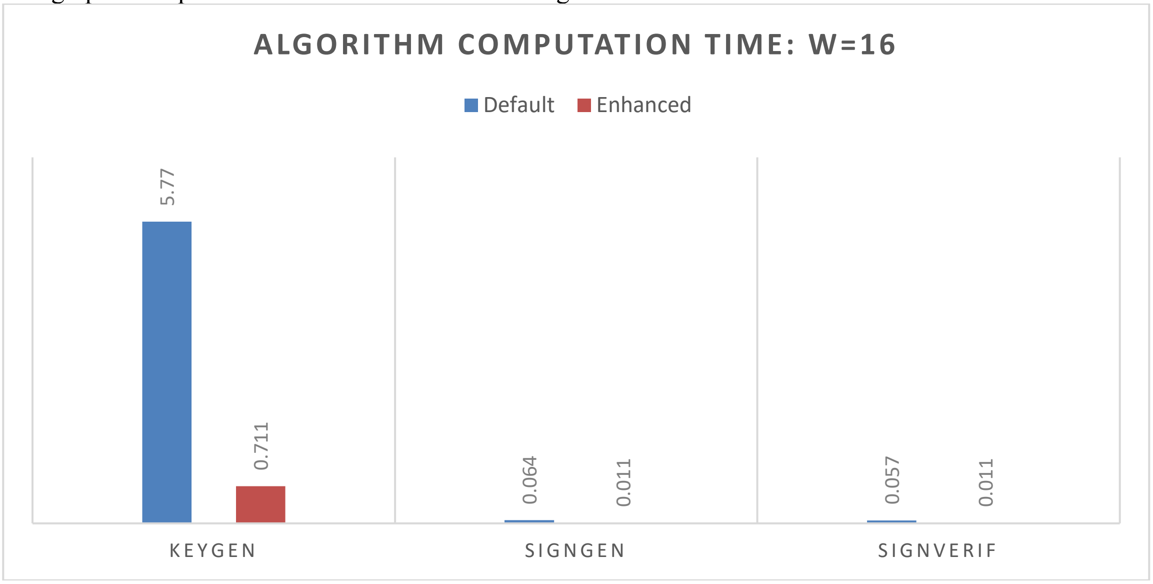


Figure 4Algorithm Computation Time (w = 16)

6.2 Performance Evaluation of i-WOTS Algorithm

The Table 2 is the summary that shows the performance of the three algorithms using the chosen parameters. The i-WOTS total execution time was 0.733 seconds as compared with the WOTS-S algorithm that has total time for execution of 0.861 seconds. This mean that the i-WOTS performed better than the WOTS-S with 14.87%. The total execution time of the i-WOTS as compared with the W-OTS and the WOTS+ algorithms.

Table 2 Summary of the Performance Analysis of i-WOTS Algorithm

Algorithm	KeyGen (s)	SignGen (s)	SignVerif (s)	Total execution time (s)
W-OTS	5.77	6.4×10^{-2}	5.7×10^{-2}	5.89
WOTS+	4.3×10^{-2}	7.45×10^{-1}	3.4×10^{-2}	8.22×10^{-1}
WOTS-S	8.12×10^{-1}	2.5×10^{-2}	2.4×10^{-2}	8.61×10^{-1}
i-WOTS	7.11×10^{-1}	1.1×10^{-2}	1.1×10^{-2}	7.33×10^{-1}

The summary of the Table 2 is represented in the graph in the Figure 5 for better understanding performance analysis.

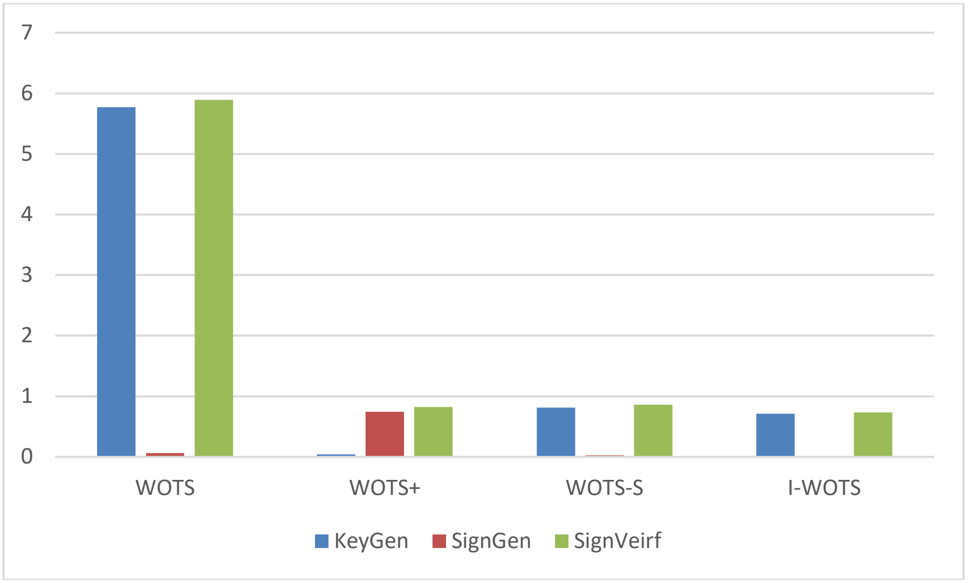


Figure 5Comparing i-WOTS with other OTS

7. The i-WOTS Algorithm Validation

The validation of the i-WOTS algorithm will be done using the big O notation. The space complexity of the i-WOTS was considered. The space complexity of the i-WOTS algorithm is subject to the message length to be signed, the winternitz parameter and the hashing algorithm used.

The Space complexity: $Sc = O(w * n)$

Where:

w: is the winternitz parameter

n: the number chains of the winternitz parameter

The i-WOTS hash function produces 256-bit outputs. If 'w' is chosen to be 16 and n=67, the space complexity $Sc = O(16 * 67) = O(1072)\, bytes$.

8. The i-WOTS Formal Security Proof

The security proof of the i-WOTS algorithm is built on the Chosen Plaintext Attack (CPA) model. The CPA is a cryptanalytic type of attack model that assumes that the adversary can find the cipher texts for some given plaintexts. The main goal for the attacker is to know the information that reduces the security of the encryption algorithm. The steps of a CPA attack process are as follows:

Step1: Attacker adaptively choose plaintext and examine their encryption techniques

Step2: At some point, attacker select messages m_0, m_1 such that $|m_0| = |m_1|$ and obtain a cipher text C_t

Step3: Attacker can choose more plaintexts and observe their encryption techniques

Step4: Attacker tries to guess whether m_0 *or* m_1 is encrypted or not.

Step5: The attacker repeat these process many times

The CPA model based on the improved algorithm is describe as follows:

It is assumed that the CPA model allows an attacker 'A' to query the signature of the chosen message $(C_m^0, C_m^1, \ldots, C_m^n)$. A signing oracle O_s responds to the attacker's queries, and returned a message and signature pair (C_m^A, Sig^A) provided that Sig^A are valid signatures of m^A *and* $m^A \notin (C_m^0, C_m^1, \ldots, C_m^n)$. In a secure CPA scheme, the probability of 'A' to be successful is very insignificant (ε).

9. CONCLUSION

In this research, an improved Winternitz One Time Signature (i-WOTS) was developed and implemented. The improvement as shown by the results is in the signature generation and verification. The i-WOTS algorithm performs better than the W-OTS+ algorithm when considering the signature creation and the time taken to verify the signature. The i-WOTS signature creation time is 44% better than the WOTS+. While in the signature verification time, the i-WOTS algorithm is 45.8% better than the WOTS+ algorithm. However, the WOTS+ algorithm outperformed the i-WOTS in terms of the time taken for key generation. The key generation time of WOTS+ is 6.05% better than the i-WOTS key generation time. The i-WOTS algorithm was validated using the Big O notation. The total space complexity of the i-WOTS was $Sc = O(16*67) = O(1072)\,bytes$. The results obtained for the total computational time as compared with other One Time Signatures was less. The space complexity performed better than the default algorithm. It is recommended that the i-WOTS algorithm could be applied on lightweight devices such as Internet of Things (IoT) because the computation time will be less with low memory consumption.

References

Bernstein, D. J., & Lange, T. (2017). Post-quantum cryptography. *Nature*, *549*(7671), (pp.188-194). https://doi.org/10.1038/nature23461

Bos, J. W., Hülsing, A., Renes, J., & van Vredendaal, C. (2021). Rapidly verifiable XMSS signatures. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 137-168.

Buchmann, J. Dahmen, E. Klintsevich, E., Okeya, K, and Vuillaume, C. (2007). "Merkle signatures with virtually unlimited signature capacity," in Applied Cryptography and Network Security, ser. LNCS, J. Katz and M. Yung, Eds., vol. 4521, Jun. 2007, (pp. 31–45).

Buchmann, J., Dahmen, E., Ereth, S., Hülsing, A., & Rückert, M. (2013). On the security of the Winternitz one-time signature scheme. *International Journal of Applied Cryptography*, *3*(1), 84-96. https://doi.org/10.1504/IJACT.2013.053435

de Oliveira, A. K. D., Lopez, J., & Cabral, R. (2017). High Performance of Hash-Based. *International Journal of Advanced Computer Science and Applications, Vol. 8, No. 3, 2017*

de Oliveira, A. K. D., & López, J. (2015, August). An Efficient Software Implementation of the Hash-Based Signature Scheme MSS and Its Variants. In *International Conference on Cryptology and Information Security in Latin America* (pp. 366-383). Springer, Cham.

Fedorov, A. K., Kiktenko, E. O., & Lvovsky, A. I. (2018). Quantum computers put blockchain security at risk. Nature (pp. 465-467). *doi: https://doi.org/10.1038/d41586-018-07449-z*

Fernández-Caramés, T. M., & Fraga-Lamas, P. (2020). Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks. *IEEE Access*, *8*, 21091-21116. *Future Generation Computer Systems*, *102*, 507-513.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (pp. 212-219).

Hülsing, A., Butin, D., Gazdag, S. L., Rijneveld, J., & Mohaisen, A. (2018). XMSS: eXtended Merkle signature scheme. In *RFC 8391*. IRTF. Retrieved from: https://www.rfc-editor.org/rfc/rfc8391.html

Joseph, D., Misoczki, R., Manzano, M., Tricot, J., Pinuaga, F. D., Lacombe, O. & Hansen, R.(2022). Transitioning organisations to post-quantum cryptography. *Nature*, *605*(7909), 237-243. https://doi.org/10.1038/s41586-022-04623-2

Kearney, J. J., & Perez-Delgado, C. A. (2021). Vulnerability of blockchain technologies to quantum attacks. *Array*, *10*, 100065. https://doi.org/10.1016/j.array.2021.100065.

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (2018). *Handbook of applied Cryptography*. CRC press.

Merkle, R. C. (1989, August). A certified digital signature. In *Conference on the Theory and Application of Cryptology* (pp. 218-238). Springer, New York, NY.

Nofer, M., Gomber, P., Hinz, O., & Schiereck, D. (2017). Blockchain. *Business & Information Systems Engineering*, *59*(3), 183-187. DOI 10.1007/s12599-017-0467-3

Pauls, F., Wittig, R., & Fettweis, G. (2019, July). A latency-optimized hash-based digital signature accelerator for the tactile Internet. In *International Conference on Embedded Computer Systems* (pp. 93-106). Springer, Cham.

Perin, L. P., Zambonin, G., Custódio, R., Moura, L., & Panario, D. (2021). Improved constant-sum encodings for hash-based signatures. *Journal of Cryptographic Engineering*, *11*(4), 329-351.

Perin, L. P., Zambonin, G., Martins, D. M. B., Custódio, R., & Martina, J. E. (2018, June). Tuning the Winternitz hash-based digital signature scheme. In *2018 IEEE Symposium on Computers and Communications (ISCC)* (pp. 00537-00542). IEEE.

Sattath, O. (2020). On the insecurity of quantum Bitcoin mining. *International Journal of Information Security*, *19*(3), 291-302. Download from https://doi.org/10.1007/s10207-020-00493-9

Shahid, F., Khan, A., Malik, S. U. R., & Choo, K. K. R. (2020). WOTS-S: A quantum secure compact signature scheme for distributed ledger.

Information Sciences, *539*, 229-249.
Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete
Signature Schemes. *International Journal of Advanced Computer Science and Applications*, *8*(3), p(421-432).
Wang, Z., Yu, H., Zhang, Z., Piao, J., & Liu, J. (2020). ECDSA weak randomness in Bitcoin. *Future Generation Computer Systems*, *102*, 507-513. https://doi.org/10.1016/j.future.2019.08.034