OSHA: A Next Generation One-way Secure Hash Algorithm

Ripon Patgiri, *Senior Member, IEEE*

Abstract—Secure hash functions are widely used cryptographic algorithms to secure diverse attacks. A one-way secure hash function is used in the various cryptographic area, for instance, digital signature. Notably, most of the hash functions provide security based on static parameters and publicly known operations. Consequently, it becomes easier to attack by the attackers because all parameters and operations are predefined. The publicly known parameters and predefined operations make the oracle regenerate the key even though it is a one-way secure hash function. Moreover, the sensitive data is mixed with the predefined constant where an oracle may find a way to discover the key. To address the above issues, we propose a novel one-way secure hash algorithm, OSHA for short, to protect sensitive data against attackers. OSHA depends on a pseudo-random number generator to generate a hash value. Particularly, OSHA mixes multiple pseudo-random numbers to produce a secure hash value. Furthermore, OSHA uses dynamic parameters, which is difficult for adversaries to guess. Unlike conventional secure hash algorithms, OSHA does not depend on fixed constants. It replaces the fixed constant with the pseudo-random numbers. Also, the input message is not mixed with the pseudo-random numbers; hence, there is no way to recover and reverse the process for the adversaries.

Index Terms—Hash function, SHA, Secure hash algorithms, Cryptography, Attacks, Cryptanalysis, Pseudo-random Number Generator, Algorithms.

1 Introduction

SECURE hash algorithms are used to solve a specific problem in certain domains, particularly, digital signature, password, SSH, Blockchain, TLS, PGP, SSL, IPsec, S/MiME, and other sensitive data. Secure hash algorithms are also used to protect passwords in our day-to-day life. The most famous cryptographically secure hash algorithms are the SHA2 and SHA3 families. However, there are preimage attacks [1], [2], cryptanalysis attacks [3] and collision attacks [4], [5]. Cryptanalysis is more powerful than other variants of attacks. Collision attacks are obvious, which can be expressed by the birthday paradox for any existing hash algorithms. The existing secure hash algorithms define constants and the number of rounds that are public and fixed. Moreover, message padding is required for the last block of the message. The existing secure hash design philosophy is based on static parameters, and as a result, these parameters are known to adversaries. In particular, the types of operations are fixed and known to adversaries. Furthermore, the message is used to derive a hash value. Hence, it makes it easier to attack the hash values.

The state-of-the-art secure hash algorithms are prone to preimage attacks [1], [2], second preimage attacks [1], [2], collision attacks [3], and cryptanalysis attacks [4], [5] due to static and public parameters. Diverse attacks on the state-of-the-art secure hash algorithms have already been reported, such as attacks on SHA1 [6], [7], attacks on SHA2 [8], attacks on SHA3 [9], attacks on BLAKE [10], and attacks on SHAKE [11]. Thus, a few research questions arise, which are outlined below-

- Q1 Can a single secure hash algorithm be used for various-sized hash value requirements? For instance, low-powered IoT devices.
- Q2 Can the predefined constants and operations be replaced, which are used by the state-of-the-art secure hash algorithms?
- Q3 Can the secure hash algorithm defeat diverse attacks?

SHAKE [12], [13], [14] addressed the question **Q1**. Since, there are diverse devices available that cannot process 256 bits; hence, it demands secure and variable-sized hash functions. Similarly, the emergence of Edge Computing also demands variable-sized hash function. In addition, the **Q2** and **Q3** create a serious security concerns. Moreover, the adversary knows all operations, constants, and parameters, making a weaker hash value. Therefore, we propose a novel and next-generation one-way secure hash algorithm, OSHA for short, to address the existing issues of secure hash algorithms. Our proposed algorithms take two inputs: secret key (input message) and seed value (however, the seed value is completely fixed and the public). Using these two inputs, OSHA generates a pseudo-random number to replace the fixed constants. The pseudo-random numbers are generated using the murmur hash function [15]. The total number of pseudo-random numbers is decided dynamically, wherein the total number is not known to the adversaries. Notably, OSHA calculates all possible parameters dynamically, including the total number of rounds, type of rotation, and the total number of rotations. In short, OSHA works on secret parameters and secret operations, which are calculated dynamically. The types of rotation and number of rotations change in each iteration. Furthermore, the new pseudo-random numbers are generated in each iteration.

• *Department of Computer Science & Engineering, National Institute of Technology Silchar, Cachar-788010, Assam, India*
E-mail: ripon@cse.nits.ac.in and URL: http://cs.nits.ac.in/rp/

Manuscript received Month 20YY; revised Month 20YY.

The existing ciphertext is XORed with the newly generated pseudo-random number in each round. Thus, OSHA creates unpredictability of the generated hash value. Therefore, we claim that OSHA is the first variant of a secure hash algorithm to use multiple pseudo-random numbers instead of predefined constants to the best of our knowledge.

This paper describes our proposed algorithm, OSHA, and compares OSHA with state-of-the-art secure hash algorithms. Moreover, we compare the features of OSHA with the state-of-the-art secure hash algorithm. OSHA is heavily dependent on the pseudo-random number generator, and henceforth, we enhance the pseudo-random number generator of existing work [16]. The enhanced pseudo-random number generator algorithm is tested in NIST SP 800-22 statistical test suite for randomness [17], [18], and results show excellent performance on the P-values and pass rates. Furthermore, we theoretically demonstrate the capability of our proposed work, and we show its strong resistance against preimage attacks, second preimage attacks, collision attacks, and cryptanalysis attacks. Thereupon, our claims are as follows-

- We devise a novel one-way secure hash algorithm, OSHA.
- It is the first variant to use a pseudo-random number instead of fixed and public constants.
- Security of OSHA depends on a pseudo-random number generator.
- All operations of OSHA are secret and dynamic.
- OSHA exhibits strong resistance against any possible attacks.
- It produces variable-sized secure hash values.
- OSHA can be adapted to keyed or keyless one-way secure hash function.

This article is organized as follows- Section 2 establishes the proposed system and provides an in-depth description. Section 3 analyzes the proposed system and compares it with existing state-of-the-art secure hash algorithms. In addition, it demonstrates the randomness analysis practically. Finally, Section 4 concludes the article.

2 OSHA: The Proposed Algorithm

We propose a novel and next generation **o**ne-way **s**ecure **h**ash **a**lgorithm, called OSHA. It extends the non-cryptographic string hash function, and it is used to generate a pseudo-random number to generate a hash value. The embodiment of OSHA is to use a pseudo-random number to produce a secure hash value. Pseudo-random numbers are highly unpredictable and secure. Accordingly, OSHA can provide better security than the existing state-of-the-art algorithm. Also, our proposed system is flexible, and it can be used for any bit size, for instance, 128-4096 or more. There is no restriction of bit sizes, unlike state-of-the-art secure hash functions. The proposed algorithm can be adapted to keyless and keyed secure hash functions depending on the applications, but both are one-way hash functions. The seed value of a keyless hash function is publicly available and fixed, while keyed hash function keeps the seed value as a secret key.

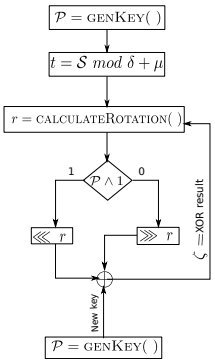

Fig. 1. Architecture of the proposed algorithm.

2.1 Description of proposed system

A pseudo-random number $\mathcal{P}$ is generated using an input message (secret key) $\mathcal{K}$ and a seed value $\mathcal{S}$, which is demonstrated in Figure 1. The $\mathcal{P}$ is circular shift rotated r times either left or right side, which is decided dynamically. The value of r changes in each iteration. It results ζ, and the ζ is XORed with a newly generated pseudo-random number $\mathcal{P}$. The pseudo-random number $\mathcal{P}$ is generated using a pseudo-random number generator. This process is repeated t times to generate a secure hash value, and the t is calculated dynamically.

Table 1 shows the required parameters and their states. All parameters are kept secret and generated dynamically. The seed value can be public or secret. There is no restriction on the seed value, and a user can input any number ≥ 4 digits. The seed value is made public and fixed. The rest values of the parameters are not known and computed at the run-time. In accordance, it is hard to retrieve the dynamically generated information by the adversaries. In addition, the input message and seed value are used to create a single bit. The input message and seed values are altered dynamically. The adversaries do not know dynamic parameters. It changes the value at run-time and in each iteration. OSHA has only two public and a static parameter which the bit size β of the hash value and seed value, and known to all.

2.2 Hash Value Generation

Algorithm 1 demonstrates generating a hash value of given message $\mathcal{K}$ and fixed seed value $\mathcal{S}$ in the OSHA algorithm. It uses a non-cryptographic string hash function to generate the pseudo-random number [15]. The $\mathcal{K}$ and $\mathcal{S}$ are used to generate a single bit of the first pseudo-random number. The pseudo-random number is used to replace the constants of the conventional hashing algorithms. The $\mathcal{K}$ and $\mathcal{S}$ are changed after generating the initial bit, and the initial message and seed value are discarded. The first generated pseudo-random number is rotated either left or right depending on the LSB bit of the pseudo-random number. The rotation's value r is calculated dynamically. The rotation process produces a new ciphertext, ζ. The ζ is XORed with a newly generated pseudo-random number $\mathcal{P}$. The pseudo-random number $\mathcal{P}$ is generated using a pseudo-random number generator 2. The pseudo-random number generator uses the murmur hash function. Murmur hash

TABLE 1
Parameters, descriptions and their state in OSHA algorithm.

Parameter	Description	State
$\mathcal{K}$	Secret Key- Input message	Secret, and Dynamic
$\mathcal{S}$	Public and fixed for keyless hash function	Public and Static
l	Length of the input string	Secret and Dynamic
β	Unrestricted bit size of hash value, for instance, $\beta = 4096$	Public and Static
t	Number of rounds	Secret, and Dynamic
r	Number of rotations	Secret, and Dynamic
Rotation type	Circular rotation, either left or right depending the last bit of the generated pseudo-random number	Secret, and Dynamic
$\mathcal{P}$	Newly generated pseudo-random number to replace constants	Secret, and Dynamic
ζ	Hash value in cipher form	Initially, it is secret and dynamic, but later, made it public.

Algorithm 1 Hash value generation using OSHA algorithm

procedure GENHASH($\mathcal{K},\ \beta$)
  $l = \text{LENGTH}(\mathcal{K})$
  $\mathcal{S} = Integer\ number$
  $\mathcal{S} = \mathcal{S} \oplus \beta$
  $\zeta = \text{GENPRNG}(\mathcal{K},\ l,\ \mathcal{S},\ \beta)$
  $\mathcal{K}' = \text{MURMUR}(\mathcal{K},\ l,\ \mathcal{S})$
  $t = (\mathcal{K}'\ mod\ \delta) + \mu$ ▷ For instance, $\mu = 5,\ \delta = 17$
  while $t \geq 1$ **do**
    $\mathcal{S} = \mathcal{S}\ \oplus\ \mathcal{K}'$
    $r = \mathcal{K}'\ mod\ \beta$
    if $P\ \wedge\ 1 = 1$ **then**
      $\zeta = \text{ROTATELEFT}(\zeta,\ r)$
    else
      $\zeta = \text{ROTATERIGHT}(\zeta,\ r)$
    end if
    $\mathcal{K} = \text{CONVERTTOSTRING}(\mathcal{K}')$
    $l = \text{LENGTH}(\mathcal{K})$
    $\mathcal{K}' = \text{MURMUR}(\mathcal{K},\ l,\ \mathcal{S})$
    $\mathcal{P} = \text{GENPRNG}(\mathcal{K},\ l,\ \mathcal{S},\ \beta)$
    $\zeta = \zeta\ \oplus\ \mathcal{P}$
    $t = t - 1$
  end while
  return ζ
end procedure

functions produce a 10-digits integer, and only a single LSB bit is recorded, and the rest are discarded. This process repeats t times to generate a secure hash value. The total number of iteration ranges between μ to δ. Moreover, the total number of rotations varies between 0 to $\beta - 1$.

2.3 Pseudo-Random Number Generator

OSHA depends on a pseudo-random number generator. The necessary conditions for the pseudo-random number generator are- consistent, secure, and statistically proven for randomness. Algorithm 2 demonstrates the generation of pseudo-random numbers. It uses the murmur hash function to generate a single bit. Conversely, the murmur hash function produces a 10-digits hash value, but a single LSB is considered in the $bin[]$ array, and the rest bits are discarded. It generates a β bits array, which is unpredictable and secure. Importantly, Algorithm 2 changes its parameters dynamically, which makes it hard to predict by the adversaries.

Algorithm 2 Pseudo-random number generator for pseudo-random number

procedure GENPRNG($\mathcal{K},\ l,\ \mathcal{S},\ \beta$)
  $i = 0$
  while $\beta \geq 1$ **do**
    $d = \text{MURMUR}(\mathcal{K},\ l,\ \mathcal{S})$
    $\mathcal{K} = d$
    $l = \text{LENGTH}(\mathcal{K})$
    $e = \text{MURMUR}(\mathcal{K},\ l,\ \mathcal{S})$
    $\mathcal{K} = \text{CONCATENATE}(d,\ e)$
    $l = \text{LENGTH}(\mathcal{K})$
    $\mathcal{S} = \mid d - e \mid$
    $bin[i] = d\ \wedge\ 1$
    $\beta = \beta - 1$
    $i = i + 1$
  end while
  return bin
end procedure

3 ANALYSIS

The adversaries know the rotation process and the total number of iteration in the conventional secure hash algorithms. As a consequence, it makes it easier to attack by adversaries. On the contrary, OSHA calculates all parameter dynamically making it hard to attack by the adversaries. The adversaries do not know whether to circular rotate left or right and how much rotation is required. Moreover, the adversary does not know how many iterations to be performed.

3.1 Time Complexity

The time complexity of Algorithm 1 depends on the bit size of the hash value; for instance, 1024. The bit size of the hash value is β. OSHA uses a bit array, and hence, it requires r time complexity to rotate the bit array. Additionally, it requires β time complexity to generate a pseudo-random number. Therefore, the time complexity of OSHA is $O(r+\beta)$ in each round. There are total t rounds in OSHA, thus, the total time complexity is $O(\beta + t(r + \beta) + r)$. The $r \leq \beta$, so, the total time complexity can be rewritten as $O(\beta + t\beta)$. Moreover, the t ranges from 5 to 17, which is a constant and small. Therefore, the total time complexity of OSHA is $O(\beta) \approx O(1)$. Now, we consider the message length l. As a consequence, the time complexity of Algorithm 1 becomes the length of the message. Consequently, we can conclude

the time complexity as $O(l)$. Hence, the time complexity depends on the input string's length.

3.2 Comparison with existing secure hash algorithm

Table 2 compares the state-of-the-art secure hash functions with OSHA. SHA family produces fixed-size output, whereas SHAKE, cSHAKE, and OSHA produce variable-sized output. SHA family, SHAKE, and cSHAKE perform fixed and predefined rounds, whereas OSHA can perform any number of rounds kept secret and calculated dynamically. Nonetheless, the minimum and the maximum number of rounds are public. SHA2 family uses modulus operation, but SHA3 family removes the modulus operation due to large integer calculation. Notably, SHA2, SHA3, SHAKE, and cSHAKE depends on the system architecture (little-endian and big-endian) due to bitwise operation. Conversely, OSHA does not depend on the system architecture because OSHA uses extra spaces $O(\beta)$ to store the bits, and thus, it is system independent. OSHA is the only variant to use a pseudo-random number to produce a hash value.

Table 3 shows the difference between state-of-the-art secure hash algorithms and OSHA. State-of-the-art secure hash algorithms use predefined constants and operations, which are public. Therefore, all operations and constants are known to adversaries too. OSHA uses secret and dynamic operations; for instance, rotation type is calculated dynamically. Furthermore, the number of rotations is calculated dynamically. As a result, there is no clue to adversaries to find the rotation type and number of rotations. In short, OSHA performs secret operations, which are calculated dynamically. On the contrary, the state-of-the-art secure hash algorithms use predefined operations and constant. OSHA generates the pseudo-random number dynamically instead of predefined constants.

3.3 Flexibility

To the best of our knowledge, SHAKE and OSHA provide flexibility in hash bit size; otherwise, the state-of-the-art secure hash algorithms can produce fixed bit size of the hash value. For example, SHA3-256 can produce 256 bits hash value while SHAKE and OSHA can produce any size of the output. A single algorithm works for 256 bits or 4096 bits, even higher bit sizes as shown in Table 4.

3.4 Outputs

Table 4 demonstrates the variable-sized output of OSHA, SHAKE128 [13], and SHAKE256 [14] for input word "OSHA". In addition, OSHA requires a seed value, and "98899" is used as a seed value. Depending on the requirements, the seed value can be made fixed and public or kept secret. SHAKE produces the same prefix different length for the same input; for instance, the prefix of 256 bits is 128 bits hash value. However, OSHA does not produce a similar prefix or suffix. It changes in changing of the bit sizes.

Notably, BLAKE is the fastest variant of secure hash algorithms [23]. The second fastest secure hash function is MD5 [23]. OSHA is slower than SHAKE because it does not depend on the predefined constants and operations. Also, OSHA uses a bit array for circular shift rotation; so, it is slower than BLAKE, but performance is similar to MD5 as shown in Table 6. Bit array makes OSHA a platform-independent secure hash algorithm.

Table 5 demonstrates the hash value of various secure hash algorithms. Also, it shows the fixed-sized hash value. On the contrary, OSHA does not restrict output size, which is similar to SHAKE128, and SHAKE256 [12]. SHA3-512 is restricted to 512 bits output size, and it cannot produce 256 or 1024 bits output. The 1024 or 2048 bits size output is not so costly for high-security requirements. Notably, the prefix of the SHAKE256 output for 256 bits, which is the same with 128 bits output size and it is shown in Table 4.

3.5 Performance

Table 6 shows the time taken to produce 1000 hash values. The BLAKE3 outperforms all, and it is the fastest secure hash algorithm. On the contrary, OSHA and MD5 generate 1000 hash values in 2 ms for 64 bits. BLAKE3 produces 1000 hash values in 2 ms for 256 bits while OSHA takes 5 ms for the same. SHAKE is slower than MD5, and hence, we exclude SHAKE in the comparison [23].

3.6 Irreversibility

Definition 1. *The function $f : A \mapsto B$ maps A to B, then the function f is said to be irreversible if the function exhibits $f : B \not\mapsto A$.*

OSHA is a one-way hash function, and hence, there is no way to regenerate the key. Therefore, OSHA follows Definition 1, and there is no way to regenerate the input from the output. The function $f : A \mapsto B$, i.e., OSHA transform any input A to output B. The input A contributes a single bit of B initially, and the pseudo-random numbers replace it. Consequently, there is no way to regenerate A from the output B. Let us assume that there exists a reversible function. The oracle need to reverse the function and regenerate the message from the hash value. Oracle needs to reverse hash value and eventually meet a single bit. Notably, it is impossible to find the input message from a single bit. Thus, OSHA guarantees that $f : B \not\mapsto A$, because it is impossible to regenerate A from B.

3.7 Irrecoverability

The function $f : A \mapsto B$, and the A is lost. OSHA guarantees $f : B \not\mapsto A$. Consequently, we cannot recover the lost input string. OSHA generates the output using a pseudo-random number generator; subsequently, it is highly unpredictable. As a result, the A is responsible for the initial bit. Thus, the A must be correct to regenerate the output B. An oracle can find reversibility; however, the oracle eventually finds the first bit but not the original string. On the contrary, conventional secure hash algorithms mix the input string with the predefined constant, where an oracle can find the reversibility of a hash value. As a consequence, input string is not recoverable from the hash value.

TABLE 2
Comparison with existing secure hash algorithm.

Name	**Output Size**	**Internal State**	**Block size**	**Rounds**	**Collision**	Operations
MD5	128	128	512	64	≤ 18	And, Xor, Rot, Add (mod 2^{32}), Or
SHA-0	160	160	512	80	< 34	And, Xor, Rot, Add (mod 2^{32}), Or
SHA-1	160	160	512	80	< 34	And, Xor, Rot, Add (mod 2^{32}), Or
SHA2-224 [19]	224	256	512	64	112	And, Xor, Rot, Add (mod 2^{32}), Or, Shr
SHA2-256 [19]	256	256	512	64	128	And, Xor, Rot, Add (mod 2^{32}), Or, Shr
SHA2-384 [19]	384	512	1024	80	192	And, Xor, Rot, Add (mod 2^{64}), Or, Shr
SHA2-512 [19]	256	512	1024	80	256	And, Xor, Rot, Add (mod 2^{64}), Or, Shr
SHA3-224 [12]	224	1600	1152	24	112	And, Xor, Rot, Not
SHA3-256 [12]	256	1600	1088	24	128	And, Xor, Rot, Not
SHA3-384 [12]	384	1600	832	24	192	And, Xor, Rot, Not
SHA3-512 [12]	512	1600	576	24	256	And, Xor, Rot, Not
SHAKE128 [12]	Unlimited	1600	1344	24	min(β/2, 128)	And, Xor, Rot, Not
SHAKE256 [12]	Unlimited	1600	1088	24	min(β/2, 256)	And, Xor, Rot, Not
cSHAKE128 [20]	Unlimited	1600	1344	24	min(β/2, 128)	And, Xor, Rot, Not
cSHAKE256 [20]	Unlimited	1600	1088	24	min(β/2, 256)	And, Xor, Rot, Not
BLAKE2s [21]	256	16 words of size 32 bits	512	10	128	
BLAKE2b [21]	256	16 words of size 64 bits	512	12	128	
BLAKE3 [22]	256	16 words of size 32 bits	512	7	128	
OSHA	Unilimited	–	64 characters but flexible	Flexible, secret, and Dynamic	$\approx \beta$	XOR, Rot, and genPRNG

TABLE 3
Difference between OSHA and state-of-the-art secure hash algorithms.

Parameters	**OSHA**	**State-of-the-art Secure Hash Algorithms**
Output size	Flexible	Fixed
Output	Output completely changes if desired output length changes for the same input	Some parts of the output are same even if desired output length of SHAKE128 and SHAKE256 change for the same input.
Rounds	Secret and Dynamic	Public and Fixed
Rotation type	Secret and Dynamic	Public and Fixed
Number of rotation	Secret and Dynamic	Public and Fixed
Mixture	Mixes with pseudo-random numbers	Mixes with predefined constants
Secret message	It contributes a single bit and define the bit patterns	Use to mix with predefined constants
Seed value	Public and fixed integer value	None
Constants	None	Public and Fixed
Pseudo-random numbers	Secret and Dynamic. It is a pseudo-random number.	None
Word size	Flexible	Fixed sizes
Padding with message	Not required	Required

3.8 Consistency

Consistency states that the output should be the same for the same input even if the platform changes. OSHA produces the same output for the same input parameters. OSHA does not depend on volatile variables. Therefore, it can produce a consistent result. Moreover, OSHA works on a bit array and random bits, and so, it can provide consistency irrespective of the system's architecture.

3.9 Rounds

Most of the conventional secure hash algorithm performs 64 rounds, which is fixed. OSHA performs μ to δ rounds of XOR, Rotation, and key generations (pseudo-random number generation). The rounds are dynamically generated between μ to δ to defend the adversaries; however, it is flexible. The user can set the total number of rounds as per their requirements to protect against the attacks. For instance, 2-17, 11-19, etc. Importantly, the δ should be a prime number. The difference between μ and δ should be significant enough to provide unpredictability; for instance,

TABLE 4
Outputs of OSHA, SHAKE128, and SHAKE256 for the input string "OSHA".

Bits	OSHA	SHAKE128	SHAKE256
16	b8a9	144e	9f6e
32	c655f80d	144e65ef	9f6e4af9
64	a68517979a06c690	144e65efe08651ca	9f6e4af9b5fdbeac
128	f1ec5c2da85661622589f6253a0d45a0	144e65efe08651ca40a9579648d4fcac	9f6e4af9b5fdbeacc748920658e9b894
256	ed4948e9382f486f4eec0f362b0410c06 ecc85fd50d492f9df02044a17eb0600	144e65efe08651ca40a9579648d4fcacf 088711566275ab8fb673b96b06c7a76	9f6e4af9b5fdbeacc748920658e9b8945 75852bc7499ba098e1513fced329367
512	c0d1f616fcebcd47c8daca7d9da4b08b5 a9396e80e734174387e4e3a9781cf0c7 3ff1b88d8e658f96857b7e6a52005f7e3 50c27224fe460a812f2130b51dffbd	144e65efe08651ca40a9579648d4fcacf 088711566275ab8fb673b96b06c7a76 6053a5dd64503da095f0094c687e12c9 af8124477f4765af904783c86aa015ff	9f6e4af9b5fdbeacc748920658e9b8945 75852bc7499ba098e1513fced329367c 943e7e61eb4863fa373b7ccb1acd2a39f 87a7c24eb7355c4607d1ecb480f76a
1024	b8c2e5fa87d15ea905ab6fce04b36041c 13f13b713abb4d187a0e817e7219f443 bb6d97ccb0bb783ac32eeaff858177260 b1fef795b31cb6254f10e33376a0f6384 20c62a7172bb1c2f8b50aa84a542ed8b 7413f588cf030ea140d53a2acfbdf8d64 40a2cf01f1c9e32beaacd41401efd4209 2407801430e36a938ae259320ef	144e65efe08651ca40a9579648d4fcacf 088711566275ab8fb673b96b06c7a76 6053a5dd64503da095f0094c687e12c9 af8124477f4765af904783c86aa015ffe 02040c6f3168d27a158c05706dd687bc ca44f132ba6b205ffab437053ff5ec844c 055670280522e032d71512c4d30eab8 d1956abe1f0fe2924858636a260e	9f6e4af9b5fdbeacc748920658e9b894 75852bc7499ba098e1513fced329367c 943e7e61eb4863fa373b7ccb1acd2a39f 87a7c24eb7355c4607d1ecb480f76ac3 07616825943b6e612874432dc4780eeb d1490b1b34ab28f208cfb7411a6497d2 9e8cbd5be9564f997de0dc4962b39450 258e17b714c09f78c85ff4fc8a11aa

TABLE 5
Output of various hash functions for the input string "OSHA"

Hash function's name	**Hash output**
MD5	7b95917312740e2be161a373be1bdce9
SHA256	7fde3b94d739f42f431d20fcac28017181462ae53873449995d45febe9ea8eb5
SHA3-256	c81957c8d5ab48bcf8971fe45580e98724d3b64e7f7780882154efb32912b757
SHA384	0aed40222131c9d872bdc76c20cf04082279ca28956ebf56fbe0c8be31384af9b1bfc4153 c1eb373646e9d114b733180
SHA3-384	0d1458a960fff9acf02844709e1b525b13f2c0513ed6558e61bbd29597ae110dab9542e1 dde20d0c7246598a8a8e4e6c
SHA512	590a860a95ada9ecd50541f7167d19caf87fb5c8aa3b1cca1fda7f12bb2af8feb91ff3a2d23 66a57047a3031bc2b392a3b077e30f8885f0c627e4671b1263692
SHA3-512	7a1fea1aa23d73491413a0a6bf21c8d325b302e7fb75843857c96a988a93bba70ab4cdd1 abda8a9dc1a23199a734e7fdf6e7c5a3fa613a7602cc17c82015f171

TABLE 6
Time taken in generating various hash values in milliseconds.

Hash function	Times in milliseconds
OSHA (64 bits)	2
MD5 (64 bits)	2
BLAKE3 (256 bits)	2
OSHA (256 bits)	5

2-17 is better than 11-19 because the difference between 2-17 is larger than 11-19. Notably, the minimum should be $\mu \geq 1$. On the contrary, if the minimum round is zero, it can also defend against many attacks because it depends on the pseudo-random numbers that are truly random and secure.

3.10 Collision resistance

Definition 2. *If there exists some functions such that $f : A \mapsto B$ and $f : C \mapsto B$ where $A \neq C$, then it is said to be collision.*

Definition 2 defines the collision where two hash values of two different input strings become the same. Generally, the collision probability of all hash functions is the same. The birthday paradox state that there is a collision probability in $2^{\frac{\beta}{2}}$ hash functions for β bits hash functions. If η items are hashed to find a collision, the collision probability is given using birthday paradox in Equation (1).

$$\rho = 1 - \frac{2^{\beta}!}{2^{\eta\beta}(2^{\beta} - \eta)!} \tag{1}$$

Solving Equation (1), we get Equation (2).

$$\begin{aligned} \rho &= 1 - e^{-\frac{\eta^2}{2^{\beta+1}}} \\ 1 - \rho &= e^{-\frac{\eta^2}{2^{\beta+1}}} \\ ln(1-\rho) &= -\frac{\eta^2}{2^{\beta+1}} \\ \eta^2 &= -2^{\beta+1}\, ln(1-\rho) \\ \eta &= 2^{\frac{\beta+1}{2}}\, \sqrt{-ln(1-\rho)} \end{aligned} \tag{2}$$

In Equation (2), we approximate $ln(1-\rho) = -\rho$, then we get Equation (3).

$$\eta = 2^{\frac{\beta+1}{2}}\, \sqrt{\rho} \tag{3}$$

Equation (3) gives us the probability of collision of any secure hash function. The η becomes enormous for 256-bits and onward. Equation (3) shows the collision probability of keyless OSHA. Notably, both keyless and keyed OSHA uses pseudo-random number which makes much harder for the attackers.

3.11 Preimage resistance

Definition 3. *Given a hash value B, a preimage attack finds a function such that $f: A \mapsto B$.*

Definition 3 defines preimage attack on the hash value. The hash value B is given, and the preimage attacker finds the input. The preimage attacks are successful in password guessing because of a weak password. Precisely, modern practice recommends a password of at least an alphabet, a digit, and a special symbol of string length eight. Still, there is a creation of a weak password, for instance, "abcd@1234". OSHA completely depends on the pseudo-random numbers other than state-of-the-art secure hash functions. If the pseudo-random number is weak, then OSHA cannot provide a strong preimage resistance. Particularly, our experimental results show that the generated pseudo-random number is secure. Therefore, keyless OSHA also provides strong resistance against preimage attacks. Notably, the applications of keyed and keyless OSHA are different, but both are secure.

Meet-in-the-middle [24] performs an exhaustive search on key spaces to achieve preimage attacks. The meet-in-the-middle has broken various secure hash algorithms [4], [5], [25] which try to perform preimage attacks. Nevertheless, this is an exhaustive search, and it takes huge computing resources. OSHA uses pseudo-random numbers, and thus, it does not follow any bit patterns. Thus, it can provide strong deterrence against meet-in-the-middle attacks.

3.12 Second preimage resistance

Definition 4. *Given a hash value B, a second preimage attack finds the functions $f: A \mapsto B$ and $f: C \mapsto B$ where $A \neq C$.*

Definition 3 defines second preimage attack. Given the hash value B to find two hash function that finds B for different inputs. Let us assume that $f: A \mapsto B$ and $f: C \mapsto B$ where $A \neq C$. OSHA depends on not only the input string but also the statistically proven pseudo-random numbers. Thus, it requires high-powered computing machinery to find the given hash value. Therefore, it is hard to find such a collision.

3.13 Cryptanalysis

There are diverse cryptanalysis attacks, particularly ciphertext-only, plaintext-only attacks, known-plaintext attacks, chosen-ciphertext attacks, chosen-plaintext attacks, adaptive chosen-ciphertext attacks, fault-injection attacks, differential cryptanalysis attacks, and linear cryptanalysis attacks. Cryptanalysis does not perform a brute-force search on the target. It performs in-depth analysis on the target and tries to find the fault/loophole to attacks on the hash values. Cryptanalysis is easier to perform if the parameters and constants are predefined. Predefined parameters and constants have hidden relations with the ciphertext. Therefore, the cryptanalysis tries to finds the relationship of all collected ciphertexts. On the contrary, OSHA does not have any relationship with the generated ciphertexts. Consequently, it is hard to perform cryptanalysis attacks.

3.14 Randomness testing

Table 7 demonstrates the randomness of Algorithm 2. Security of the OSHA depends on pseudo-random numbers, which are generated by Algorithm 2. The randomness of Algorithm 2 is tested in NITS SP 800-22. Table 7 demonstrates the P-values and pass rate of the generated bits using Algorithm 2. Initially, we have generated 10M random bits of the word "OSHA" and the number 98899. We have chosen a weak word to demonstrate the performance of our algorithm. The generated random bits are tested in NIST SP 800-22 statistical test suite [17], [18] for 32 bits, 64 bits, and 128 bits stream. NIST SP 800-22 test suit provides approximate entropy, frequency, block frequency, cumulative sums, runs, longest runs, rank, FFT, non-overlapping template, overlapping template, random excursions, random excursions variant, serial, linear complexity, and universal tests. The minimum pass rate of 32bits, 64 bits, and 128 bits stream is 0.96875, 0.9375, and 0.9765625, respectively. The minimum P-value of 32 bits, 64 bits, and 128 bits stream is 0.100508, 0.134686, and 0.015065, respectively. The P-value must be ≥ 0.001 to be considered as a random number. The maximum P-values of 32 bits, 64 bits, and 128 bits stream are 0.991468, 0.991468, and 0.788728, respectively. R. Patgiri [16] reported the highest P-values are 0.976060, 0.991468 and 0.941144 in 32 bits, 64 bits, and 128 bits streams, respectively. The minimum P-value of R. Patgiri [16] are 0.035174, 0.012043, 0.017912 in 32 bits, 64 bits, and 128 bits streams, respectively. Our proposed pseudo-random number generator clearly enhances and outperforms the pseudo-random number generator of R. Patgiri [16]. Similarly, R. Patgiri [16] reports test's minimum pass rates are 0.96875, 0.96875, and 0.9765625 in 32 bits, 64bits, and 128 bits stream, respectively; whereas our proposed pseudo-random number generator clearly surpasses the minimum pass rate.

4 Conclusion

In this paper, we have presented a one-way secure hash algorithm, OSHA for short, to produce a secure hash value and it can defend against diverse attacks. OSHA uses murmur hash functions to generate a single bit of a pseudo-random number. It uses multiple pseudo-random numbers to replace the predefined constants. Additionally, OSHA uses two variables as input, mainly secret message (key) and seed value, to generate a hash value. The input message and seed value contribute a single bit. Therefore, it does not use in the rest bit generation, but it defined the future bits. The seed value can be fixed and public or kept secret depending on the requirements of the hash function. A keyed one-way secure hash function does not share the seed value, for instance, generating a password hash value. Furthermore, OSHA can generate variable-sized hash values similar to SHAKE hash algorithms. OSHA calculates the parameters' value dynamically, and thus, parameters' values are not known to adversaries.

Moreover, the operation type is decided dynamically. Furthermore, OSHA performs an XOR operation with a newly generated pseudo-random number, but the original message is not used in the XORing process. Therefore, it provides truly one-way secure hash functions. Due to the dynamic property of OSHA, it provides strong resistance

TABLE 7
P-values and success rates of Algorithms 2 for 32, 64 and 128 bits in NIST SP 800-22.

Test name	32 bits		64 bits		128 bits	
	P-value	Pass rate	P-value	Pass rate	P-value	Pass rate
Approximate Entropy	0.100508	32/32	0.500934	64/64	0.350485	126/128
Frequency	0.862344	32/32	0.134686	64/64	0.178278	128/128
Block Frequency	0.949602	31/32	0.468595	63/64	0.054199	127/128
Cumulative sums	0.213309	32/32	0.324180	64/64	0.364146	128/128
Runs	0.213309	31/32	0.324180	60/64	0.619772	124/128
Longest runs	0.407091	31/32	0.350485	64/64	0.654467	127/128
Rank	0.671779	32/32	0.407091	64/64	0.222869	128/128
FFT	0.911413	32/32	0.500934	61/64	0.110952	127/128
Non-overlapping Template	0.991468	32/32	0.991468	64/64	0.788728	128/128
Overlapping Template	0.468595	32/32	0.862344	64/64	0.275709	127/128
Random Excursions	0.275709	13/13	0.162606	17/17	0.162606	15/15
Random Excursions Variant	0.637119	13/13	0.275709	17/17	0.275709	15/15
Serial	0.299251	32/32	0.671779	63/64	0.422034	128/128
Linear complexity	0.407091	31/32	0.911413	64/64	0.015065	125/128
Universal	0.534146	31/32	0.671779	64/64	0.350485	126/128

against diverse attacks, particularly preimage attacks, second preimage attacks, collision attacks, and cryptanalysis attacks. Significantly, there are diverse applications of keyed and keyless hash functions, for example, Edge Computing, IoT, Blockchain, Cloud Computing etc.

REFERENCES

[1] D. Khovratovich, C. Rechberger, and A. Savelieva, "Bicliques for preimages: Attacks on skein-512 and the sha-2 family," in *Fast Software Encryption*, A. Canteaut, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 244–263.

[2] T. Espitau, P.-A. Fouque, and P. Karpman, "Higher-Order Differential Meet-in-the-middle Preimage Attacks on SHA-1 and BLAKE," in *Advances in Cryptology – CRYPTO 2015*. Berlin, Germany: Springer, Aug 2015, pp. 683–701.

[3] A. Biryukov, M. Lamberger, F. Mendel, and I. Nikolić, "Second-Order Differential Collisions for Reduced SHA-256," in *Advances in Cryptology – ASIACRYPT 2011*. Berlin, Germany: Springer, Dec 2011, pp. 270–287.

[4] J. Guo, S. Ling, C. Rechberger, and H. Wang, "Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2," in *Advances in Cryptology - ASIACRYPT 2010*. Berlin, Germany: Springer, Dec 2010, pp. 56–75.

[5] J. Li, T. Isobe, and K. Shibutani, "Converting Meet-In-The-Middle Preimage Attack into Pseudo Collision Attack: Application to SHA-2," in *Fast Software Encryption*. Berlin, Germany: Springer, Mar 2012, pp. 264–286.

[6] M. Stevens, "New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis," in *Advances in Cryptology – EUROCRYPT 2013*. Berlin, Germany: Springer, May 2013, pp. 245–261.

[7] G. Leurent and T. Peyrin, "From Collisions to Chosen-Prefix Collisions Application to Full SHA-1," in *Advances in Cryptology – EUROCRYPT 2019*. Cham, Switzerland: Springer, Apr 2019, pp. 527–555.

[8] A. Hosoyamada and Y. Sasaki, "Quantum collision attacks on reduced SHA-256 and SHA-512," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 292, 2021. [Online]. Available: https://eprint.iacr.org/2021/292

[9] J. Guo, G. Liao, G. Liu, M. Liu, K. Qiao, and L. Song, "Practical Collision Attacks against Round-Reduced SHA-3," *J. Cryptology*, vol. 33, no. 1, pp. 228–270, Jan 2020.

[10] Y. Hao, "The Boomerang Attacks on BLAKE and BLAKE2," in *Information Security and Cryptology*. Cham, Switzerland: Springer, Mar 2015, pp. 286–310.

[11] T. Li and Y. Sun, "Preimage Attacks on Round-Reduced Keccak-224/256 via an Allocating Approach," in *Advances in Cryptology – EUROCRYPT 2019*. Cham, Switzerland: Springer, Apr 2019, pp. 556–584.

[12] O. Standards and N. I. Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," *CSRC | NIST*, Aug 2015. [Online]. Available: https://www.nist.gov/publications/sha-3-standard-permutation-based-hash-and-extendable-output-functions

[13] "Shake-128 Online," April 2021, [Online; accessed on April 2021]. [Online]. Available: https://emn178.github.io/online-tools/shake_128.html

[14] "Shake-256 Online," April 2021, [Online; accessed on April 2021]. [Online]. Available: https://emn178.github.io/online-tools/shake_256.html

[15] A. Appleby, "Murmurhash," Retrieved on December 2020 from https://sites.google.com/site/murmurhash/, 2008.

[16] R. Patgiri, "Stealth: A highly secured end-to-end symmetric communication protocol," Cryptology ePrint Archive, Report 2021/622, 2021, https://eprint.iacr.org/2021/622.

[17] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Booz-allen and hamilton inc mclean va, Tech. Rep., 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf

[18] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks *et al.*, *SP 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications*. National Institute of Standards & Technology, 2010. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-22/rev-1a/final

[19] "Announcing Approval of Federal Information Processing Standard (FIPS) 180-2, Secure Hash Standard; a Revision of FIPS 180-1," Aug 2002, [Online; accessed 23. May 2021]. [Online]. Available: https://www.federalregister.gov/documents/2002/08/26/02-21599/announcing-approval-of-federal-information-processing-standard-fips-180-2-secure-hash-standard-a

[20] J. Kelsey, S.-j. Change, and R. Perlner, "SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST SP 800-185, Dec. 2016. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf

[21] J. Guo, P. Karpman, I. Nikolić, L. Wang, and S. Wu, "Analysis of BLAKE2," in *Topics in Cryptology – CT-RSA 2014*. Cham, Switzerland: Springer, Feb 2014, pp. 402–423.

[22] Blake3-team, "BLAKE3-specs," April 2021, [Online; accessed April 2021]. [Online]. Available: https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf

[23] "BLAKE2," Nov 2020, [Online; accessed April 2021]. [Online]. Available: https://www.blake2.net

[24] W. Diffie and M. E. Hellman, "Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard," *Computer*, vol. 10, no. 6, pp. 74–84, Jun 1977.

[25] K. Aoki and Y. Sasaki, "Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1," in *Advances in Cryptology - CRYPTO 2009*. Berlin, Germany: Springer, Aug 2009, pp. 70–89.