OSHA: A General-purpose One-way Secure Hash Algorithm

Ripon Patgiri$^{[0000-0002-9899-9152]}$

National Institute of Technology Silchar
Cachar-788010, Assam, India
`ripon@cse.nits.ac.in/rp/`
`http://cs.nits.ac.in/rp/`

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, password protection. However, most of the hash functions provide security based on static parameters and publicly known operations. Therefore, 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 key (sensitive data) is mixed with the predefined constant where an oracle may find a way to discover the key. To address the above issues of the secure hash functions, we propose a novel and 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 private key. Moreover, OSHA mixes multiple private keys to generate a 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 private keys. Also, the key is not mixed with the private keys; hence, there is no way to recover and reverse the process for the adversaries.

Keywords: Hash function · SHA · Secure hash algorithm · Cryptography · Attacks · Cryptanalysis

1 Introduction

Secure hash algorithms are used to solve a specific problem in certain domains, particularly, Password, SSH, Blockchain, TLS, PGP, SSL, IPsec, S/MiME, and other sensitive data. Secure hash algorithms are used to protect passwords in our day-to-day life. The most famous cryptographically secure hash algorithms are SHA2 and SHA3 families. However, there are preimage attacks [17,10], cryptanalysis attacks [7] and collision attacks [13,19]. 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 therefore, these parameters are known to adversaries. Moreover, 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.

Existing state-of-the-art secure hash algorithms are prone to preimage attacks [17,10], second preimage attacks [17,10], collision attacks [7], and crypt-analysis attacks [13,19] due to static and public parameters. Diverse reports on attacks have already been published, such as attacks on SHA1 [24,18], attacks on SHA2 [15], attacks on SHA3 [12], attacks on BLAKE [14], and attacks on SHAKE [20]. 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?
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 [23,3,4] addressed the question **Q1**. The **Q2** and **Q3** create a serious security concerns. Moreover, the adversary knows all operations, constants, and parameters, which makes a weaker hash value. Therefore, we propose a 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 and semi-secret seed value (however, the seed value can be completely kept secret). Using these two secret inputs, OSHA generates a pseudo-random number (term as a private key) to replace the fixed constants. The private keys are generated using the murmur hash function [5]. The total number of private keys is decided dynamically, and therefore, it is not known to the adversaries. Moreover, 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 private keys are generated in each iteration. The existing ciphertext is XORed with the newly generated private key in each round. Thus, OSHA creates unpredictability of the generated hash value. OSHA is the first variant of a secure hash algorithm to use multiple private keys instead of predefined constants to the best of our knowledge.

This paper describes the OSHA algorithms and compares OSHA with state-of-the-art secure hash algorithms; however, we have excluded performance comparison between OSHA and compares OSHA with the state-of-the-art secure hash algorithm. OSHA heavily dependent on the pseudo-random number generator, and therefore, we enhance the pseudo-random number generator of existing work [21]. The enhanced pseudo-random number generator algorithm is tested in NIST SP 800-22 statistical test suite for randomness [22,6], and results show excellent performance on the P-values and pass rates. Moreover, 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.

This paper 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. Moreover, it demonstrates the randomness analysis practically. Section 4 discusses on OSHA algorithms. Finally, Section 5 concludes the paper.

2 OSHA: The proposed algorithm

We propose a novel one-way secure hash algorithm called OSHA. OSHA depends on the non-cryptographic string hash function. The non-cryptographic string hash function is used to generate a pseudo-random number to generate a hash value. OSHA is the first secure hash algorithm to use a pseudo-random number to produce hash value to the best of our knowledge. Pseudo-random numbers are highly unpredictable and secure. Therefore, 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.

Our assumption of the proposed algorithm is as follows- we assume two secret keys, particularly the passphrase and the number. The number is used as the initial seed value. The number may be semi-secret greater than four digits, for instance, date of birth in the form of ddmmyyyy, mmddyyyy, yyyyddmm, or yyyy, i.e., it is easy to remember for the user but difficult for adversaries.

2.1 Description of proposed system

Figure 1 demonstrates the working mechanism of our proposed system. Firstly, a private key $\mathcal{P}$ is generated using a secret key $\mathcal{K}$ and a semi-secret seed value $\mathcal{S}$. 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 private key $\mathcal{P}$. The private key $\mathcal{P}$ is generated using a pseudo-random number generator. This process is repeated t times to generate a 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. However, the seed value is semi-secret because it can be the date of birth, zip code, year, phone number, etc., which easier to remember. However, there no restriction on the seed value. A user can input any number ≥ 4 digits. The value of the parameters is not known and computed at the run-time. Therefore, it is hard to retrieve the dynamically generated information by the adversaries. Moreover, the secret key and semi-secret seed value are used to generate a single bit. However, the secret key and seed value define future bit patterns. The secret key and seed value are altered dynamically. The adversaries do not know dynamic parameters. It changes the value at run-time and each iteration. OSHA has only one public and a static parameter which the bit size β of the hash value, and known to all.

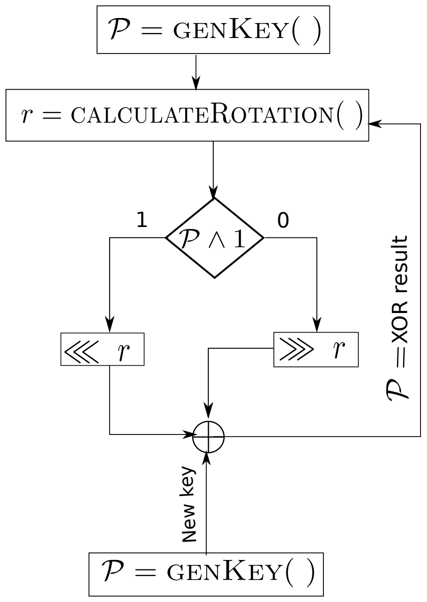
$\mathcal{P} = \textsc{genKey}(\ )$
$r = \textsc{calculateRotation}(\ )$
1
0
$\mathcal{P} \wedge 1$
$\lll r$
$\ggg r$
$\mathcal{P} =$ XOR result
New key
$\mathcal{P} = \textsc{genKey}(\ )$

Fig. 1. Architecture of the proposed algorithm

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

Parameter	**Description**	**State**
$\mathcal{K}$	Secret Key- Password	Secret, and Dynamic
$\mathcal{S}$	Semi-secret- Date of Birth, Zip code, Year, etc. which is greater than 4 digits and less than integer range	Semi-secret, and Dynamic
l	Length of the input secret key	Secret and Dynamic
β	Unrestricted bit size of hash value, for instance, $\beta = 4096$	Public and Static
t	Number of rounds	Semi-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 private key	Secret, and Dynamic
$\mathcal{P}$	Newly generated private key	Secret, and Dynamic
ζ	Hash value in cipher form	Secret, and Dynamic

2.2 Proposed algorithm

Algorithm 1 demonstrates generating a hash value of given key $\mathcal{K}$ and semi-secret seed value $\mathcal{S}$ in the OSHA algorithm. It uses a non-cryptographic string hash function to generate the pseudo-random number [5]. The $\mathcal{K}$ and $\mathcal{S}$ are used to generate a single bit of the first private key. The $\mathcal{K}$ and $\mathcal{S}$ are changed after generating the initial bit, and the initial key and seed value are discarded later.

Algorithm 1 Hash value generation using OSHA algorithm

```
1: procedure GENHASH($\mathcal{K}$, $l$, $\mathcal{S}$, $\beta$)
2:     $seed = seed \oplus \beta$
3:     $\mathcal{P}$ = GENPRNG($\mathcal{K}$, $l$, $\mathcal{S}$, $\beta$)
4:     $\mu = 5,\ \delta = 57$
5:     $t = (\mathcal{S}\ mod\ \delta) + \mu$
6:     while $t \geq 1$ do
7:         $\mathcal{K}$ = MURMUR($\mathcal{K}$, $l$, $\mathcal{S}$)
8:         $\mathcal{S} = \mathcal{S} \oplus \mathcal{K}$
9:         $l$ = LENGTH($\mathcal{K}$)
10:        $r = \mathcal{K}\ mod\ \beta$
11:        if $P \wedge 1 = 1$ then
12:            $\zeta$ = ROTATELEFT($\zeta$, $r$)
13:        else
14:            $\zeta$ = ROTATERIGHT($\zeta$, $r$)
15:        end if
16:        $\mathcal{P}$ = GENPRNG($\mathcal{K}$, $l$, $\mathcal{S}$, $\beta$)
17:        $\zeta = \zeta \oplus \mathcal{P}$
18:        $t = t - 1$
19:    end while
20:    $\alpha$ = CONVERTINTOHEXADECIMAL($\zeta$, $\beta$)
21:    return $\alpha$
22: end procedure
```

The first generated private key is rotated either left or right depending on the LSB bit of the private key. The rotation's value r is calculated dynamically. The rotation process results ζ. The ζ is XORed with a newly generated private key $\mathcal{P}$. The private key $\mathcal{P}$ is generated using a pseudo-random number generator. The pseudo-random number generator uses the murmur hash function. Murmur hash functions produce a 10-digits integer; however, 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 10 to γ, and it is calculated dynamically. 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. However, 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 bit array, which is unpredictable and secure. Moreover, Algorithm 2 changes its parameters dynamically, which makes it hard to predict by the adversaries.

Algorithm 2 Pseudo-random number generator for private key

```
1: procedure genKey($\mathcal{K}$, $l$, $\mathcal{S}$, $\beta$)
2:     $i = 0$
3:     while $\beta \geq 1$ do
4:         $d =$ Murmur($\mathcal{K}$, $l$, $\mathcal{S}$)
5:         $\mathcal{K} = d$
6:         $l =$ Length($\mathcal{K}$)
7:         $e =$ Murmur($\mathcal{K}$, $l$, $\mathcal{S}$)
8:         $\mathcal{K} =$ Concatenate($d$, $e$)
9:         $l =$ Length($\mathcal{K}$)
10:        $\mathcal{S} = \mid d - e \mid$
11:        $bin[i] = d \wedge 1$
12:        $\beta = \beta - 1$
13:        $i = i + 1$
14:    end while
15:    return $bin$
16: end procedure
```

3 Analysis

The adversaries know the process of rotation and the total number of iteration in the secure hash algorithms. Therefore, it makes it easy to attack by adversaries. On the contrary, OSHA algorithm calculates all parameter dynamically which makes 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 are to perform.

3.1 Time Complexity

The time complexity of OSHA 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 therefore, it requires r time complexity to rotate the bit array. Moreover, 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 $r \leq \beta$, therefore, the total time complexity can be rewritten as $O(\beta + t\beta)$. Moreover, the t ranges from 5 to 61, which is a constant and small. Therefore, the total time complexity of OSHA is $O(\beta) \approx O(1)$. Therefore, the time complexity depends on the bit size of the hash function's output.

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 size output. SHA family, SHAKE, and cSHAKE perform fixed and predefined rounds, whereas OSHA can perform any number of rounds which is

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 [1]	224	256	512	64	112	And, Xor, Rot, Add (mod 2^{32}), Or, Shr
SHA2-256 [1]	256	256	512	64	128	And, Xor, Rot, Add (mod 2^{32}), Or, Shr
SHA2-384 [1]	384	512	1024	80	192	And, Xor, Rot, Add (mod 2^{64}), Or, Shr
SHA2-512 [1]	256	512	1024	80	256	And, Xor, Rot, Add (mod 2^{64}), Or, Shr
SHA3-224 [23]	224	1600	1152	24	112	And, Xor, Rot, Not
SHA3-256 [23]	256	1600	1088	24	128	And, Xor, Rot, Not
SHA3-384 [23]	384	1600	832	24	192	And, Xor, Rot, Not
SHA3-512 [23]	512	1600	576	24	256	And, Xor, Rot, Not
SHAKE128 [23]	Unlimited	1600	1344	24	min($\beta/2$, 128)	And, Xor, Rot, Not
SHAKE256 [23]	Unlimited	1600	1088	24	min($\beta/2$, 256)	And, Xor, Rot, Not
cSHAKE128 [16]	Unlimited	1600	1344	24	min($\beta/2$, 128)	And, Xor, Rot, Not
cSHAKE256 [16]	Unlimited	1600	1088	24	min($\beta/2$, 256)	And, Xor, Rot, Not
BLAKE2s [11]	256	16 words of size 32 bits	512	10	128	
BLAKE2b [11]	256	16 words of size 64 bits	512	12	128	
BLAKE3 [8]	256	16 words of size 32 bits	512	7	128	
OSHA	Unilimited	–	Flexible	Flexible, secret, and Dynamic	$\frac{\beta}{2}$	XOR, Rot, and Key-Gen

kept secret and calculated dynamically. However, the minimum and the maximum number of rounds are public. SHA2 family uses modulus operation; however, SHA3 family removes the modulus operation due to large integer calculation. Moreover, SHA2, SHA3, SHAKE, and cSHAKE depends on the system architecture (little-endian and big-endian) due to bitwise operation; however, 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. 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 Key	It contributes a single bit and define the bit patterns	Use to mix with predefined constants
Seed value	Semi-secret integer value	None
Constants	None	Public and Fixed
Private keys	Secret and Dynamic	None
Word size	Any sizes	Fixed sizes
Padding with message	Not required	Required

Table 3 shows the difference between state-of-the-art secure hash algorithms and OSHA. State-of-the-art secure hash algorithms use predefined constant and operation, which is public. Therefore, all operations and constants are known to adversaries too. OSHA uses secret and dynamic operations; for instance, rotation type is calculated dynamically. Moreover, the number of rotations is calculated dynamically. Therefore, there is no clue to adversaries to find the rotation type and number rotation. 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 size.

3.4 Outputs

Table 4 demonstrates the variable-sized output of OSHA, SHAKE128 [3], and SHAKE256 [4] for input word "ieee". Moreover, OSHA requires seed value, and "1982" is used as a seed value. SHAKE produces the same prefix; 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 [2]; however, SHAKE is faster than SHA3. 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; therefore, it is slower than other secure hash algorithms. Bit array makes OSHA a platform-independent secure hash algorithm. But OSHA can provide strong resistance against any possible attacks.

3.5 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 therefore, 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. Therefore, there is no way to regenerate A from the output B. Let us assume that there exists a reversible function. The reversible function can regenerate the first bit of the hash value, and it is impossible to find the input key from a single bit. Thus, OSHA guarantees that $f: B \not\mapsto A$, because it is impossible to regenerate A from B.

3.6 Irrecoverability

The function $f: A \mapsto B$, and the A is lost. OSHA guarantees $f: B \not\mapsto A$. Therefore, we cannot recover the lost input string. OSHA generates the output using a pseudo-random number generator; therefore, it is highly unpredictable. Moreover, the A is responsible for the initial bit and defines future bit patterns. 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.

Table 4. Outputs of OSHA, SHAKE128, and SHAKE256 for the input “ieee”.

Bits	OSHA	SHAKE128	SHAKE256
16	351b	63f6	140f
32	26db221b	63f6416e	140f8322
64	4ad1e6f1b28e82e0	63f6416e1cf1ecff	140f83226bb36e2c
128	5006ef06df2bb690a6229813077f641d	63f6416e1cf1ecff63a52a8c8316eaf1	140f83226bb36e2cbc46630c114940db
256	1a82a1cc551fb90f5bdd10e056f09e35 5858ffb3dafd5c589ad38c5903156864	63f6416e1cf1ecff63a52a8c8316eaf12b c659f75542275fb4add18576e24d44	140f83226bb36e2cbc46630c114940db f39ec957cbd1f76edbf58748082d303e
512	112896744e66525bb15e0f12e0a34762 dde9c78cbcdf817a0400bfc77a11ca9d 120b9594069b325f2c9a486e0fd2e89d 4af77812013311dcb55414e3485aae2e	63f6416e1cf1ecff63a52a8c8316eaf12b c659f75542275fb4add18576e24d440d dc06ccd9bb24c78ffa855a2bc730f87b 7274a93adda5774465d186498d6b0d	140f83226bb36e2cbc46630c114940db f39ec957cbd1f76edbf58748082d303e 5e326dbf66a021bfe1240b27cbf5144f e4d7b80a8e1334a1c7383b6f99795a0f
1024	d30c354417c509dc3493ca04158cee0 87ec5adc6138f2d37f5ba5fa49c962d8 5cd39a9e7c4a2bcf7eacc180046721c1 8f4223200b3482198ecc9a09c4d7a45 abb33b500951988be09175b2dfae714 344c091ee85cfa580aecdc9353c93c78 e5dde9eed96c0f1b342260bbd78d6ae faf0121113fe58a3a45ae4ac8ec9bcf8 17d2	63f6416e1cf1ecff63a52a8c8316eaf12b c659f75542275fb4add18576e24d440 ddc06ccd9bb24c78ffa855a2bc730f87 b7274a93adda5774465d186498d6b0 ddeb3fae3ed8e3504611a947c37c0980 d9f3d4682eef23f0ec3badef120b0375 d18d2c0031f1155de6082b33d559d1b 59a2d0b18197ba3ab4c107b0427e12 9fb6	140f83226bb36e2cbc46630c114940d bf39ec957cbd1f76edbf58748082d303 e5e326dbf66a021bfe1240b27cbf5144 fe4d7b80a8e1334a1c7383b6f99795a 0fd4220f9a83803e6a8a03ce00fda054 1a0626f76598fb1de47a0d564d2108e 0ecef697759020cf7ad522b22162c970 8d824e1e55cbda71669f3de86a7be7c 20f4

3.7 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 therefore, it can provide consistency irrespective of the system's architecture.

3.8 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. The rounds are dynamically generated between μ to δ to depend on the adversaries; however, it is flexible. The total number of rounds can be set by the user as per their requirements, for instance, 5-62, 10-100, etc., to better protect against attack. However, the δ should be a prime number. The difference should be significant enough to provide unpredictability; for instance, 5-57 is better than 23-61 because the difference between 5-57 is larger than 23-61. However, the minimum should be $\mu \geq 5$. 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.

3.9 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 become the same using different input. 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. However, OSHA uses two secret keys, and therefore, the combination of the two secret keys is $\binom{\eta}{2} = \frac{\eta(\eta-1)}{2}$. The probability of picking a correct pair is $\frac{2}{\eta(\eta-1)}$. The probability of not picking a correct pair is $(1 - \frac{2}{\eta(\eta-1)})$. The η is large, and therefore, we approximate the probability $\frac{2}{\eta(\eta-1)} \approx 0$; thus, the probability of not picking a correct pair is 1. With this approximation, we can rewrite Equation (1), and thus the probability of collision becomes 0, which is given in Equation (4).

$$\begin{aligned} \rho &\approx 1 - \frac{2^{\beta}!}{2^{\beta}(2^{\beta}-1)!} \\ \rho &\approx 1 - \frac{2^{\beta}}{2^{\beta}} \\ \rho &\approx 0 \end{aligned} \tag{4}$$

However, Equation (4) is an approximation of the probability, and it shows the difficulties in getting collision attacks.

Therefore, OSHA does not restrict output size similar to SHAKE128, and SHAKE256 [23]. 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 is the same with 128 bits output size; for example, if SHAKE outputs E7 for 8 bits, then it outputs E75A for 16 bits.

3.10 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. However, 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 provides strong security even if there is a weak password because OSHA uses a semi-secret seed value. However, the seed value can be completely a secret by providing an unpredictable number greater than four-digit. Therefore, OSHA provides strong resistance against preimage attacks.

Meet-in-the-middle [9] performs an exhaustive search on key spaces to achieve preimage attacks. The meet-in-the-middle has broken various secure hash algorithms citeAoki,Guo,Li which tries to perform preimage attack. However, this is an exhaustive search, and it takes huge computing resources. OSHA uses two keys; one is the secret key, and the other is the seed value. A meet-in-the-middle attack must perform a search on both keys. Therefore, OSHA can provide strong resistance against meet-in-the-middle attacks.

3.11 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 seed value. Therefore, it requires four inputs to find the given hash value. Therefore, it is hard to find such a collision.

3.12 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. 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 ciphertexts. Consequently, it is hard to perform cryptanalysis attacks.

3.13 Randomness testing

Table 5 demonstrates the randomness of Algorithm 2. The randomness of Algorithm 1 is tested in NITS SP 800-22. Table 5 demonstrates the P-values and pass rate of the generated bits using Algorithm 2. Initially, we have generated 10M random bits of the word "ieee" and the number 1982. The generated random bits are tested in NIST SP 800-22 statistical test suite [22,6] 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, 0.984375, and 0.9921875, respectively. The minimum P-value of 32 bits, 64 bits, and 128 bits stream is 0.043745, 0.275709, and 0.078086, 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 0.999896, 0.964295, and 0.964295, respectively.

Table 5. 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	0.949602	32/32	0.437274	63/64	0.078086	127/128
Frequency	0.739918	32/32	0.468595	64/64	0.964295	127/128
Block Frequency	0.468595	32/32	0.568055	64/64	0.031497	127/128
Cumulative sums	0.043745	32/32	0.637119	64/64	0.437274	127/128
Runs	0.671779	32/32	0.378138	64/64	0.242986	128/128
Longest runs	0.804337	32/32	0.275709	64/64	0.086458	128/128
Rank	0.949602	32/32	0.407091	64/64	0.204076	128/128
FFT	0.178278	31/32	0.324180	63/64	0.070445	128/128
Non-overlapping Template	0.999896	32/32	0.964295	64/64	0.957319	128/128
Overlapping Template	0.299251	32/32	0.437274	63/64	0.407091	128/128
Random Excursions	0.834308	11/11	0.739918	12/12	0.834308	13/13
Random Excursions Variant	0.834308	11/11	0.739918	12/12	0.637119	13/13
Serial	0.862344	32/32	0.324180	63/64	0.568055	128/128
Linear complexity	0.739918	32/32	0.739918	63/64	0.568055	128/128
Universal	0.602458	31/32	0.299251	64/64	0.204076	128/128

4 Discussion

OSHA is slower than state-of-the-art secure hash algorithms due to bit array and private keys, but it can be faster than any other variants of the secure hash algorithms by using zero rounds. Can a zero round prevent all kinds of possible attacks to the hash value? Let us assume that zero round cannot protect the possible attacks on the hash value. A zero round in OSHA performs a pseudo-random key generation and performs circular shift rotation. We know that random numbers are cryptographically secure, and therefore, a zero round can provide good security on the hash value. If a zero round can provide high security, then why should OSHA performs more rounds? More rounds can provide higher unpredictability. Therefore, OSHA keeps all operations are calculated dynamically, and OSHA removes predefined constants.

5 Conclusion

In this paper, we have presented a one-way secure hash algorithm, OSHA for short, to protect against diverse attacks on the hash value. OSHA uses murmur hash functions to generate a single bit of a private key. It uses multiple private keys to replace the predefined constants. Moreover, OSHA uses two secrets, mainly secret message (key) and secret seed value to generate a hash value. The secret key and seed value contribute a single bit, and the two secret values define the rest bits. OSHA can generate variable-sized hash values similar to SHAKE hash algorithms. OSHA calculates the parameters' value dynamically, and therefore, parameters' values are secret. Moreover, the operation type is decided dynamically. Furthermore, OSHA performs XOR operation with newly generated private keys, 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 against diverse attacks, particularly preimage attacks, second preimage attacks, collision attacks, and cryptanalysis attacks.

References

1. Announcing Approval of Federal Information Processing Standard (FIPS) 180-2, Secure Hash Standard; a Revision of FIPS 180-1 (Aug 2002), `https://www.federalregister.gov/documents/2002/08/26/02-21599/announcing-approval-of-federal-information-processing-standard-fips-180-2-secure-hash-standard-a`, [Online; accessed 23. May 2021]
2. BLAKE2 (Nov 2020), `https://www.blake2.net`, [Online; accessed April 2021]
3. Shake-128 Online (April 2021), `https://emn178.github.io/online-tools/shake_128.html`, [Online; accessed on April 2021]
4. Shake-256 Online (April 2021), `https://emn178.github.io/online-tools/shake_256.html`, [Online; accessed on April 2021]
5. Appleby, A.: Murmurhash. Retrieved on December 2020 from https://sites.google.com/site/murmurhash/ (2008)
6. Bassham III, L.E., Rukhin, A.L., Soto, J., Nechvatal, J.R., Smid, M.E., Barker, E.B., Leigh, S.D., Levenson, M., Vangel, M., Banks, D.L., 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), `https://csrc.nist.gov/publications/detail/sp/800-22/rev-1a/final`
7. Biryukov, A., Lamberger, M., Mendel, F., Nikolić, I.: Second-Order Differential Collisions for Reduced SHA-256. In: Advances in Cryptology – ASIACRYPT 2011, pp. 270–287. Springer, Berlin, Germany (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_15
8. Blake3-team: BLAKE3-specs (April 2021), `https://github.com/BLAKE3-team/BLAKE3-specs/blob/master/blake3.pdf`, [Online; accessed April 2021]
9. Diffie, W., Hellman, M.E.: Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer **10**(6), 74–84 (Jun 1977). https://doi.org/10.1109/C-M.1977.217750
10. Espitau, T., Fouque, P.A., Karpman, P.: Higher-Order Differential Meet-in-the-middle Preimage Attacks on SHA-1 and BLAKE. In: Advances in Cryptology – CRYPTO 2015, pp. 683–701. Springer, Berlin, Germany (Aug 2015). https://doi.org/10.1007/978-3-662-47989-6_33
11. Guo, J., Karpman, P., Nikolić, I., Wang, L., Wu, S.: Analysis of BLAKE2. In: Topics in Cryptology – CT-RSA 2014, pp. 402–423. Springer, Cham, Switzerland (Feb 2014). https://doi.org/10.1007/978-3-319-04852-9_21

12. Guo, J., Liao, G., Liu, G., Liu, M., Qiao, K., Song, L.: Practical Collision Attacks against Round-Reduced SHA-3. J. Cryptology **33**(1), 228–270 (Jan 2020). https://doi.org/10.1007/s00145-019-09313-3
13. Guo, J., Ling, S., Rechberger, C., Wang, H.: 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, pp. 56–75. Springer, Berlin, Germany (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_4
14. Hao, Y.: The Boomerang Attacks on BLAKE and BLAKE2. In: Information Security and Cryptology, pp. 286–310. Springer, Cham, Switzerland (Mar 2015). https://doi.org/10.1007/978-3-319-16745-9_16
15. Hosoyamada, A., Sasaki, Y.: Quantum collision attacks on reduced SHA-256 and SHA-512. IACR Cryptol. ePrint Arch. **2021**, 292 (2021), `https://eprint.iacr.org/2021/292`
16. Kelsey, J., Change, S.j., Perlner, R.: SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash. Tech. Rep. NIST SP 800-185, National Institute of Standards and Technology, Gaithersburg, MD (Dec 2016). https://doi.org/10.6028/NIST.SP.800-185, `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-185.pdf`
17. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for preimages: Attacks on skein-512 and the sha-2 family. In: Canteaut, A. (ed.) Fast Software Encryption. pp. 244–263. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
18. Leurent, G., Peyrin, T.: From Collisions to Chosen-Prefix Collisions Application to Full SHA-1. In: Advances in Cryptology – EUROCRYPT 2019, pp. 527–555. Springer, Cham, Switzerland (Apr 2019). https://doi.org/10.1007/978-3-030-17659-4_18
19. Li, J., Isobe, T., Shibutani, K.: Converting Meet-In-The-Middle Preimage Attack into Pseudo Collision Attack: Application to SHA-2. In: Fast Software Encryption, pp. 264–286. Springer, Berlin, Germany (Mar 2012). https://doi.org/10.1007/978-3-642-34047-5_16
20. Li, T., Sun, Y.: Preimage Attacks on Round-Reduced Keccak-224/256 via an Allocating Approach. In: Advances in Cryptology – EUROCRYPT 2019, pp. 556–584. Springer, Cham, Switzerland (Apr 2019). https://doi.org/10.1007/978-3-030-17659-4_19
21. Patgiri, R.: Stealth: A highly secured end-to-end symmetric communication protocol. Cryptology ePrint Archive, Report 2021/622 (2021), `https://eprint.iacr.org/2021/622`
22. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. Tech. rep., Booz-allen and hamilton inc mclean va (2001), `https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf`
23. Standards, O., Technology, N.I.: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. CSRC | NIST (Aug 2015). https://doi.org/10.6028/NIST.FIPS.202, `https://www.nist.gov/publications/sha-3-standard-permutation-based-hash-and-extendable-output-functions`
24. Stevens, M.: New Collision Attacks on SHA-1 Based on Optimal Joint Local-Collision Analysis. In: Advances in Cryptology – EUROCRYPT 2013, pp. 245–261. Springer, Berlin, Germany (May 2013). https://doi.org/10.1007/978-3-642-38348-9_15