Journal of Information Processing Systems, Vol.8, No.1, March 2012 http://dx.doi.org/10.3745/JIPS.2012.8.1.119

An Improved Fast and Secure Hash Algorithm

Siddharth Agarwal*, Abhinav Rungta*, R.Padmavathy*, Mayank Shankar* and Nipun Rajan*

Abstract—Recently, a fast and secure hash function SFHA – 256 has been proposed and claimed as more secure and as having a better performance than the SHA – 256. In this paper an improved version of SFHA – 256 is proposed and analyzed using two parameters, namely the avalanche effect and uniform deviation. The experimental results and further analysis ensures the performance of the newly proposed and improved SFHA-256. From the analysis it can be concluded that the newly proposed algorithm is more secure, efficient, and practical.

Keywords—SHA-256, SFHA-256, Improved SFHA-256

1. Introduction

The hash function H accepts the variable-sized message M as input and outputs a fixed-size representation H(M) of M, which is sometimes called a message digest [1]. I.B. Damgard et.el., discussed the construction of hash functions and presented an efficient and much more secure scheme with the combination of RSA system with the collision free hash function based on factoring [2]. Hash functions for message authentications are proposed in [3]. A universal one-way hash function family is discussed in [4].

SHA-1 is a cryptographic hash function published by the National Institute of Standards and Technology (NIST). The three SHA algorithms are SHA-0, SHA-1, and SHA-2. The SHA-0 algorithm was not used in many applications. On the other hand, SHA-2 differs from the SHA-1 hash function. SHA-1 is the most widely used hash function. Several widely-used security applications and protocols are based on SHA-1. In 2005, security flaws were identified in SHA-1 [5].

A prime motivation for the publication of the Secure Hash Algorithm was the Digital Signature Standard. The Digital Signature Algorithm (DSA) is a United States Federal Government standard or FIPS for digital signatures. It was proposed by the National Institute of Standards and Technology (NIST) in August 1991 for use in their Digital Signature Standard (DSS). The ElGamal signature scheme is a digital signature scheme that is based on the difficulty of computing discrete logarithms. It was described by Taher ElGamal in 1984 [6]. The Elliptic Curve Digital Signature Algorithm (ECDSA) is a variant of the Digital Signature Algorithm (DSA) that uses Elliptic curve cryptography [7, 8].

Recently, Hassan. M. Elkamchouchi et el., proposed a fast and secure hash function (SFHA -

Manuscript received June 30, 2011; accepted August 29, 2011.
Corresponding Author: R.Padmavathy

* Department of Computer Science and Engineering National Institute of Technology, Warangal (r_padma@rediffmail.com, rpadma@nitw.ac.in)

Copyright ⓒ 2012 KIPS (ISSN 1976-913X)

256) and claimed that SFHA-256 is more secure and that it has better performance than SHA-256 [9]. In the current study, an improved version of SFHA is proposed. Using two parameters, such as the avalanche effect and uniform deviation, the performance of the newly proposed algorithm is tested.

The rest of the paper is organized as follows: Section 2 presents the review of secure and fast hashing algorithm. Section 3 discusses the improvement over SFHA. Section 4 presents the analysis between SFHA and the newly proposed improved SFHA and Section 5 contains the concluding remarks.

2. Review of Secure and Fast Hashing Algorithm (SFHA-256)

A hash function accepts the variable-size message m as input and produces a fixed-size hash code H(m), which is also called a message digest, as output. The hash code is a function of all the bits of the message and provides an error detection capability, where a change to any bit or bits in the message results in a change to the hash code.

In this section, we will describe SFHA-256 [9]. These are the basic notations:

(1) +: addition mod 2^{32}

(2) $\oplus$: XOR, $\wedge$: AND, $\vee$: OR.

(3) A <<< s: s-bit left shift rotation a 32-bit string.

Input Block Length and Padding

The input message is processed by 512 bit blocks. SFHA-256 pads a message by appending a single bit 1 next to the least significant bit of the message, followed by zero or more bits of 0's until the length of the message is 448 modulo 512, and then appends the 64-bit original message length modulo 2^{64} to the message.

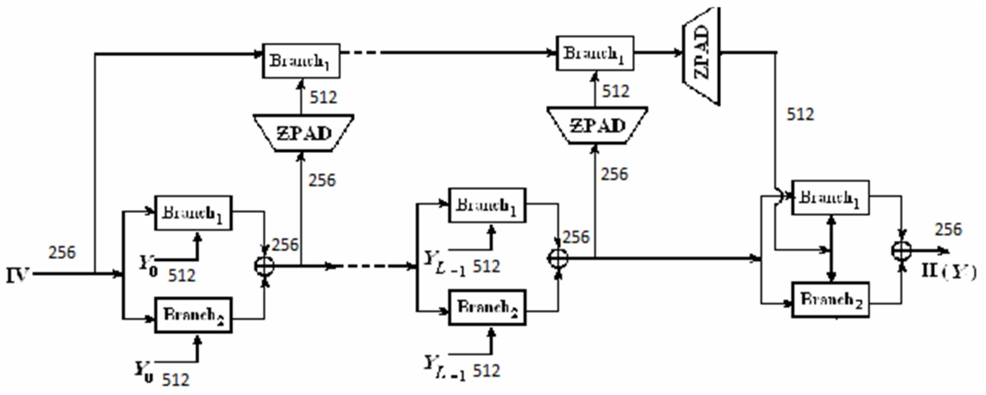

Fig. 1. Construction of SFHA-256

Compression Function

The SFHA-256 overall compression function consists of two compression functions: function f, which is iterated in the cascade chain and function f `, which is iterated in the accumulation chain. The function f consists of two parallel branch functions Branch1 and Branch2. So, the

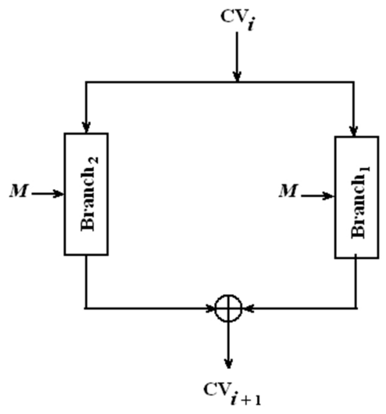

Fig. 2. SFHA-256 cascade chain compression function

attacker who tries to break the function should simultaneously aim for the two branches. The SFHA-256 compression function consists of two parallel branches. This means that SFHA-256 can be implemented in hardware and it is difficult to analyze the two branches simultaneously. The function f ` consists of a single branch of f. Let CV_i = [A, B, C, D, E, F, G, H] be the chaining variable of the compression function. It is initialized to IV_0 which is:

A_0 = 6a09e667 B_0 = bb67ae85 C_0 = 3c6ef372 D_0 = a54ff53a
E_0 = 510e527f F_0 = 9b05688c G_0 = 1f83d9ab H_0 = 5be0cd19

which are obtained by taking the fractional part of the square roots of the first eight primes in hexagonal values. Each successive 512-bit message block M is divided into sixteen 32-bit words M0, M1… M15 and the following computation is performed to update CV_i to CV_{i+1}

CV_{i+1} = Branch1 (CV_i , M) ⊕ Branch2 (CV_i , M) (1)

where M is the re-ordering of message words for the two branches as follows:
Branch1: M = (M_0, M_1… M_{15})
Branch2: M = (M_{15}, M_{14}… M_0)

Constants

The compression function of SFHA-256 uses sixteen constants for Branch1, which are ordered as follows:

β_0 = 428a2f98β_1 = 71374491 β_2 = b5c0fbcf β_3 = e9b5dba5
β_4 = 3956c25b β_5 = 59f111f1 β_6 = 923f82a4 β_7 = ab1c5ed5
β_8 = d807aa98 β_9 = 12835b01 β_{10} = 243185be β_{11} = 550c7dc3
β_{12} = 72be5d74 β_{13} = 80deb1fe β_{14} = 9bdc08a7 β_{15} = c19bf174

For Branch2 the order is reversed.

By using these constants we achieve the goal to disturb the attacker who tries to find good differential characteristics with a relatively high probability. So, we prefer the constants that represent the first thirty-two bits of the fractional parts of the cube roots of the first sixteen four prime

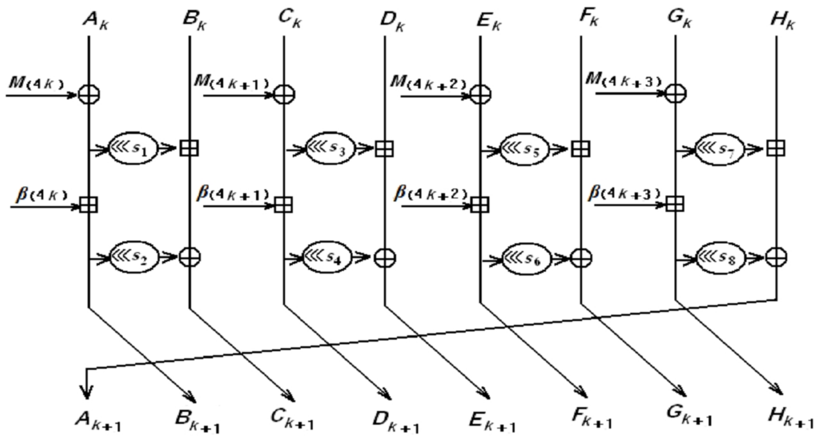

Fig. 3. Step Function in Compression Function

numbers.

Step Functions:

The input registers of the k-th step is divided into eight 32-bit words: (A_k ,B_k , C_k, D_k, E_k, F_k , G_k , H_k).

At the k + 1 step the following computations are done

$$A_{k+1} = [H_k + (G_k \oplus M_{(4k+3)})^{\lll s7}] \oplus [(G_k \oplus M_{(4k+3)}) + \beta_{(4k+3)}]^{\lll s8}$$
$$B_{k+1} = (A_k \oplus M_{(4k)}) + \beta_{(4k)}$$
$$C_{k+1} = [B_k + (A_k \oplus M_{(4k)})^{\lll s1}] \oplus [(A_k \oplus M_{(4k)}) + \beta_{(4k)}]^{\lll s2}$$
$$D_{k+1} = (C_k \oplus M_{(4k+1)}) + \beta_{(4k+1)}$$
$$E_{k+1} = [D_k + (C_k \oplus M_{(4k+1)}{}^{\lll s3}] \oplus [(C_k \oplus M_{(4k+1)}) + \beta_{(4k+1)}]^{\lll s4}$$
$$F_{k+1} = (E_k \oplus M_{(4k+2)}) + \beta_{(4k+2)}$$
$$G_{k+1} = [F_k + (E_k \oplus M_{(4k+2)}{}^{\lll s5}] \oplus [(E_k \oplus M_{(4k+2)}) + \beta_{(4k+2)}]^{\lll s6}$$
$$H_{k+1} = (G_k \oplus M_{(4k+3)}) + \beta_{(4k+3)}$$

SFHA-256 is more dynamic in the sense that in the step function the input controls what happens in the algorithm. In other words, the shift rotations starts with strong constants then these change as the algorithm moves forward and this will cause more complexity for someone who is trying to create a collision rather than keeping them constant.

Shift Rotations

Initially for k = 0 the amounts of shift rotations are given as:

$s_1 = 5$, $s_2 = 9$, $s_3 = 3$, $s_4 = 11$, $s_5 = 8$, $s_6 = 13$, $s_7 = 7$, $s_8 = 10$.

Then for the k = 1, 2, 3 the values of the shift rotations depend on the input (the content of the registers) as follows:

$s_1 \leftarrow A_k + B_k + C_k$, $s_2 \leftarrow B_k + C_k + D_k$, $s_3 \leftarrow C_k + D_k + E_k$
$s_4 \leftarrow D_k + E_k + F_k$, $s_5 \leftarrow E_k + F_k + G_k$, $s_6 \leftarrow F_k + G_k + H_k$
$s_7 \leftarrow G_k + H_k + A_k$, $s_8 \leftarrow H_k + A_k + B_k$

2.1 Comparative Analysis of SFHA-256 and SHA-256

- SHA-256 uses iterative M-D construction, whereas SFHA-256 makes use of a 3C-construction (modified M-D).
- SFHA-256 explicitly uses an accumulation chain to better resist collisions, while SHA-256 has no such provision.
- The SFHA-256 compression function consists of two parallel branches. This means that SFHA-256 can be implemented in hardware and it is difficult to analyze the two branches simultaneously.
- Unlike SHA-256, SFHA-256 does not use any boolean operations but only addition, XOR, and shift operations. This, in addition to reducing the total number of bit operation of the algorithm, distributes non-linearity among all blocks in a round.

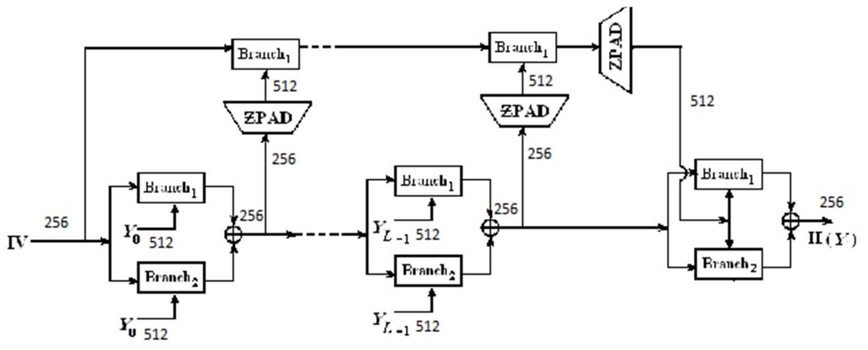

Fig. 4. Secure and Fast Hashing Algorithm (SFHA-256)

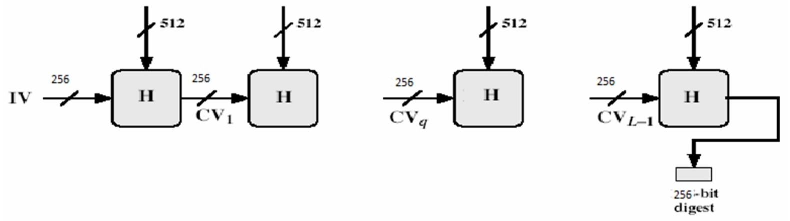

Fig. 5. Secure Hash Algorithm (SHA-256)

Shortcomings

The following shortcomings were observed in the design of SFHA-256 that potentially pose a significant security threat to the algorithm:

- The use of branch function Branch1 in accumulation function f', which has already been used in cascade function f, which is an obvious repetition.
- The use of the ZPAD function to expand the compressed bits of fixed length of 256 bits to 512 bits always appends the identical bit pattern.

The performance to the changes would be tested using the following criteria:

- The avalanche test to ensure a change in at least 50% of the bits for change in one bit of input message.

- The uniform distribution test to ensure minimum deviation from the ideal distribution of hashes of substantial numbers of random strings that have been distributed evenly among a fixed number of ranges.

3. IMPROVEMENTS

3.1 First Modification

The first modification consists of a change in the ZPAD function of the cascading chain. The ZPAD function is not recurrent as in SFHA – 256. The padding for subsequent 512 bit message blocks is differing in the cascading chain. The padding consists of appending Z with a bit 1 and some 0's followed by the binary encoded representation of the length of Z to make it a block of b bits. The position of the bit 1 right shifts by one position recursively. This padding technique reduces the redundancy of padding function. The padded block is given as input in the accumulation chain.

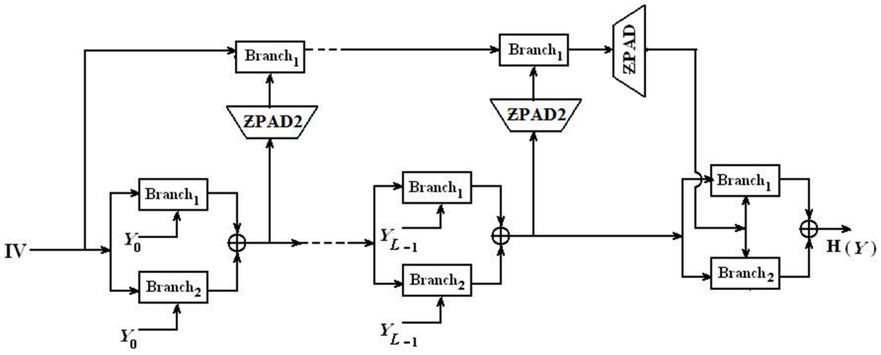

Fig. 6. First Modification

3.2 Second Modification

The second modification consists of a change in the branch function of the accumulation

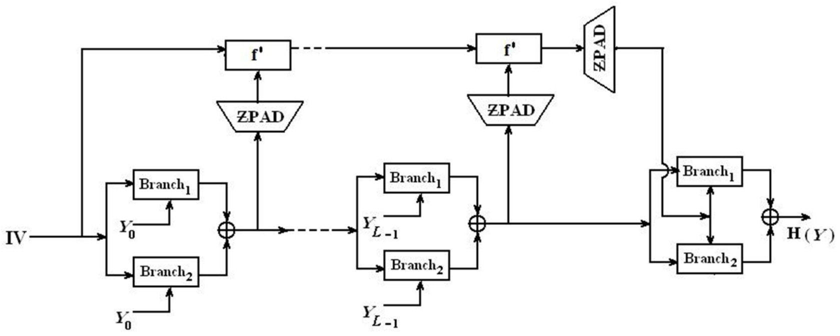

Fig. 7. Second Modification

chain. The implementation of this branch function in the accumulation chain was an obvious repetition to the branch function implemented in the cascading chain. If a cryptanalyst reverses one part of the hash coming from the cascading chain, then the hash from the accumulation chain is also broken. The new branch function f' uses the following step function:

Step Functions

The input registers of the k-th step is divided into eight 32-bit words: (A_k, B_k ,C_k, D_k, E_k, F_k , G_k , H_k).

At the k + 1step the following computations are done:

$A_{k+1} = (B_k \oplus M_{(4k)}) + \beta_{(4k)}$

$B_{k+1} = [C_k + (D_k \oplus M_{(4k+1)})^{<<<s5}] \oplus [(D_k \oplus M_{(4k+1)}) + \beta_{(4k+1)}]^{<<< s6}$

$C_{k+1} = (D_k \oplus M_{(4k+1)}) + \beta_{(4k+1)}$

$D_{k+1} = [E_k + (F_k \oplus M_{(4k+2)})^{<<<s}3] \oplus [(F_k \oplus M_{(4k+2)}) + \beta_{(4k+2)}]^{<<< s4}$

$E_{k+1} = (F_k \oplus M_{(4k+2)}) + \beta_{(4k+2)}$

$F_{k+1} = [G_k + (H_k \oplus M_{(4k+3)})^{<<<s}1] \oplus [(H_k \oplus M_{(4k+3)}) + \beta_{(4k+3)}]^{<<< s2}$

$G_{k+1} = (H_k \oplus M_{(4k+3)}) + \beta_{(4k+3)}$

$H_{k+1}= [A_k+ (B_k \oplus M_{(4k)})^{<<<s7}] \oplus [(B_k \oplus M_{(4k)}) + \beta_{(4k)}]^{<<< s8}$

Shift Rotations

Initially for k = 0 the amounts of shift rotations are given as:

$s_1 = 5$, $s_2 = 9$, $s_3 = 3$, $s_4 = 11$, $s_5 = 8$, $s_6 = 13$, $s_7 = 7$, $s_8 = 10$.

Then for the k = 1, 2, 3 the values of the shift rotations depend on the input (the content of the registers) as follows:

$s_8 \leftarrow A_k + B_k + C_k$, $s_7 \leftarrow B_k + C_k + D_k$, ,$s_6 \leftarrow C_k + D_k + E_k$

$s_5 \leftarrow D_k + E_k + F_k$, $s_4 \leftarrow E_k + F_k + G_k$, $s_3 \leftarrow F_k + G_k + H_k$

$s_2 \leftarrow G_k + H_k + A_k$, $s_1 \leftarrow H_k + A_k + B_k$

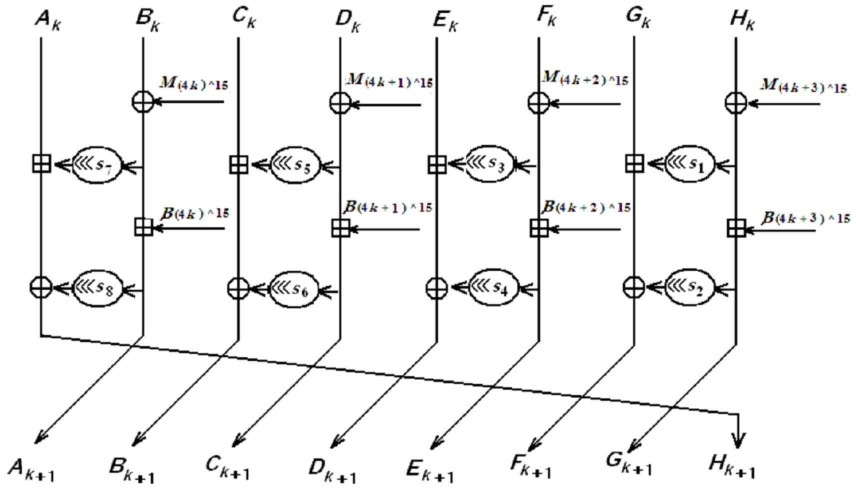

Fig. 8. Step functions

3.3 Third Modification

The third modification consists of adding a second level to the accumulation chain, as shown in the figure given below. We use the same branch function here as in the second modification. This modification uses a modified version of the accumulation chain in which the new compression function, f`, which is used in the second modification, is applied in a manner to accept its output as feedback. The output from the first application of f` is fed back along with $C_{i-1}$to be processed again by f` the output of which is XORed with the original output.

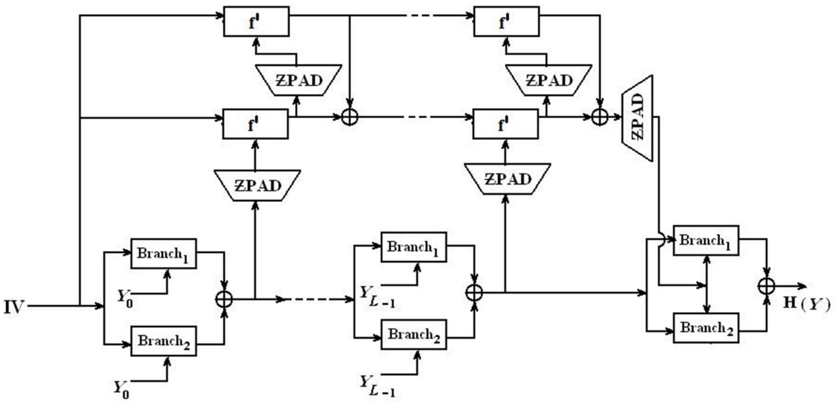

Fig. 9. Third Modification

4. Results

4.1 The Avalanche Effect and the Uniform Distribution Test

The avalanche effect is a desirable property of a hash function that signifies the probability that the output bit of a hash result will change when a bit in the input is changed. A very high or low probability of change of bits is undesirable as it gives an attacker a high probability of whether a given bit changes or not with a change of one bit in the input. A 0% probability of change of output bit means that the hash output does not change when the input differs by a single bit. In the same manner, a 100% probability of change of output means that the bit is bound to reverse on change of input by a single bit.

Strict Avalanche Criterion:

The strict avalanche criterion (SAC) is a generalization of the avalanche effect. It is satisfied if, whenever a single input bit is complemented, each of the output bits change with a 50% probability. The SAC builds on the concepts of completeness and avalanche.

The following tables give the average value of the percentage of bits that flipped upon the change of one bit in an input message over different message lengths.

Uniform Distribution Tests:

SFHA has an output of 256 bit, so it has 2^{256} possible hash value ranges from 0 to 2^{256}-1. For a perfect uniform distribution for this output, the average value of the hash values equals $[(2^{256}-1)/2] \cong 5.7896\times10^{76}$. Practically, it is impossible to obtain a perfect uniform distribution hash

Table 1. Average percentage of differing bits for different message lengths

Message Length	Probability of bit flip			
	SFHA-256	First Modification	Second Modification	Third Modification
1	50.244	50.244	50.049	48.633
2	50.903	49.634	50.879	49.292
3	50.439	49.658	49.74	49.984
4	49.915	50.037	50.635	49.438
5	51.025	49.834	49.814	50.342
6	51.034	50.578	50.822	50.366
7	50.405	50.356	50.314	49.819
8	50.134	49.518	50.378	49.463
9	49.821	50.212	50.092	49.501
10	50.137	50.039	50.166	49.897
11	50.568	50.164	49.996	49.374
12	50.106	50.269	49.878	50.077
13	50.143	49.598	49.73	50.413
14	49.676	50.774	50.499	49.69
15	50.189	49.948	50.055	50.098
16	50.04	50.568	49.924	50.22
17	49.54	49.635	49.79	49.943
18	50.418	49.995	49.609	50.203
19	49.789	49.653	50.198	49.933
20	50.042	49.675	50.076	49.583
21	49.937	50.395	49.988	50.2
22	49.956	49.936	49.88	50.244
23	49.938	49.824	50.284	49.943
24	50.405	50.429	50.248	50.014
25	49.979	50.045	49.803	50.023
26	49.822	49.961	50.077	50.131
27	50.212	49.873	49.495	49.839
28	50.08	50.054	50.221	49.89
29	50.005	49.897	50.283	50.263
30	49.777	50.137	50.236	50.171
31	49.929	49.865	50.12	49.965
32	50.159	49.768	50.049	49.857
Average	50.14896875	50.01790625	50.104	49.90028125

algorithm. Instead, the less the deviation from the uniform distribution the more secure the algorithm is.

Input:

The input provided to obtain the following results was a string of length 32 containing the English upper case alphabet in order and recurring on end (i.e. “ABCDEFGHIJKLMNOPQRS TUVWXYZABCDEF”).

For each message length, a substring of this string was provided starting from the first character.

Table 2. Average percentage of differing bits for different message lengths

Message Length	Deviation from ideal			
	SFHA-256	First Modification	Second Modification	Third Modification
1	-0.0361	-0.0361	0.3466	0.1911
2	-0.0189	0.0374	-0.2201	0.1368
3	0.1196	-0.0628	0.0635	0.0386
4	-0.0679	-0.0661	0.2033	0.1518
5	0.0783	0.0107	-0.0304	0.0414
6	0.1822	-0.0778	-0.0352	-0.0271
7	-0.0991	0.0906	0.0286	-0.0451
8	0.1172	-0.0785	-0.0177	0.0948
9	-0.0371	-0.1273	-0.0784	0.128
10	0.0139	0.0059	0.0968	0.1051
11	-0.0548	0.0149	0.0702	0.0609
12	0.139	-0.0058	-0.049	-0.0727
13	-0.0788	-0.0326	-0.0732	0.0604
14	0.0732	0.0678	-0.0041	0.1725
15	-0.075	-0.0749	-0.0749	0.0362
16	-0.0035	0.0038	0.0503	0.0175
17	-0.0397	0.0472	-0.0936	0.0228
18	0.06	-0.0983	-0.0156	-0.0625
19	-0.0645	-0.0372	-0.0105	-0.0061
20	-0.0169	0.036	-0.01	-0.0216
21	-0.0225	-0.0715	0.0216	-0.0199
22	0.094	0.0703	0.0567	0.0081
23	-0.0657	-0.0308	-0.0073	0.0795
24	-0.0615	0.068	0.0269	0.0603
25	-0.0395	-0.0073	-0.0086	-0.0178
26	0.0825	-0.0242	0.0062	0.0468
27	0.0453	0.0041	-0.0469	0.0028
28	0.0433	0.064	-0.0056	-0.0145
29	0.0595	0.0883	-0.0456	0.0302
30	0.0825	0.0043	0.0656	-0.0308
31	0.0387	-0.0127	0.0439	-0.0162
32	0.0376	0.0014	-0.0536	-0.0182
Average	0.064009375	0.04558125	0.061265625	0.057440625

Observed runtimes of the IBE based email system [10]

An encryption and subsequent decryption of the message, “Hello world,” was done and the average duration for the operations were recorded as given below:

MD5:14.320s

SFHA-256:14.303s

First Modification:14.883s

Second Modification::14.114s

Third Modification:15.187s

4.1.1 SFHA-256 vs First Modification

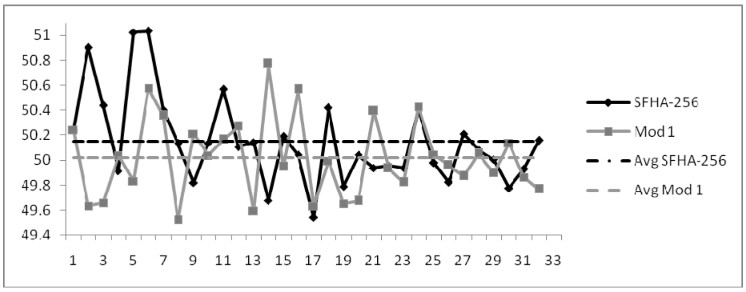

Fig. 10. Avalanche Effect (Percentage points vs. Message length)

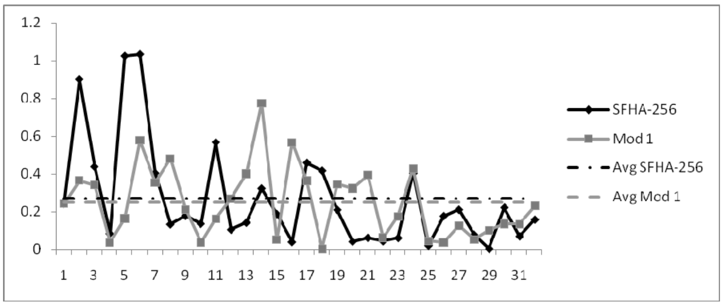

Fig. 11. Avalanche Effect - Deviation (Percentage points vs Message length)

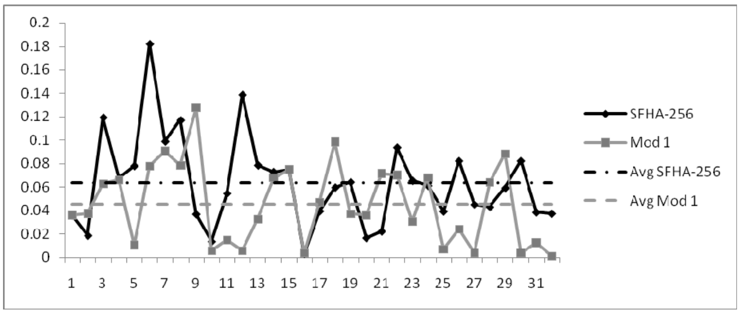

Fig. 12. Uniform Distribution Test - Deviation (Percentage points vs. Message length)

4.1.2 SFHA-256 vs Second Modification

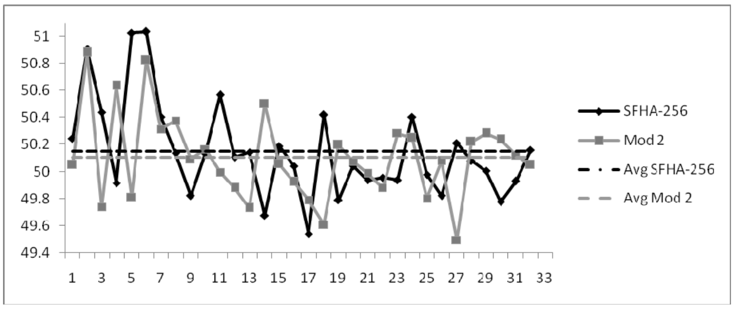

Fig. 13. Avalanche Effect (Percentage points vs. Message length)

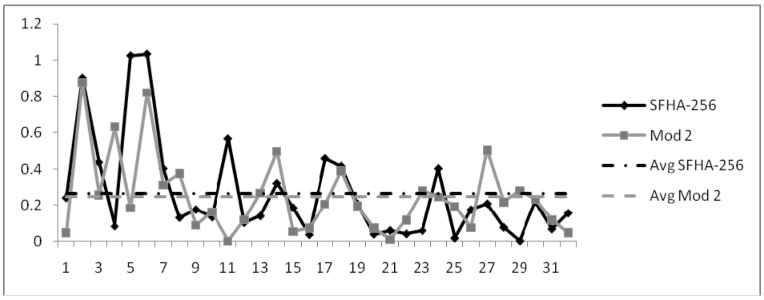

Fig. 14. Avalanche Effect - Deviation (Percentage points vs Message length)

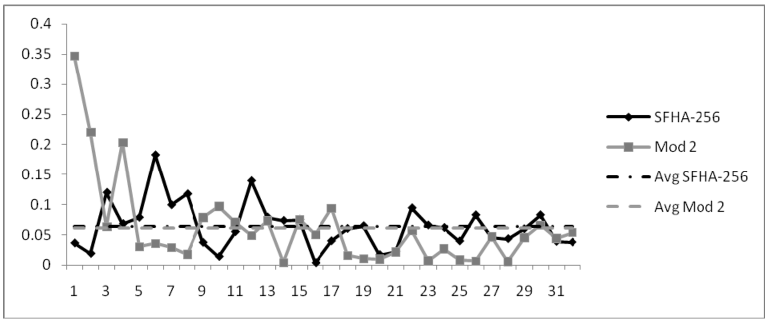

Fig. 15. Uniform Distribution Test - Deviation (Percentage points vs. Message length)

4.1.3 SFHA-256 vs Third Modification

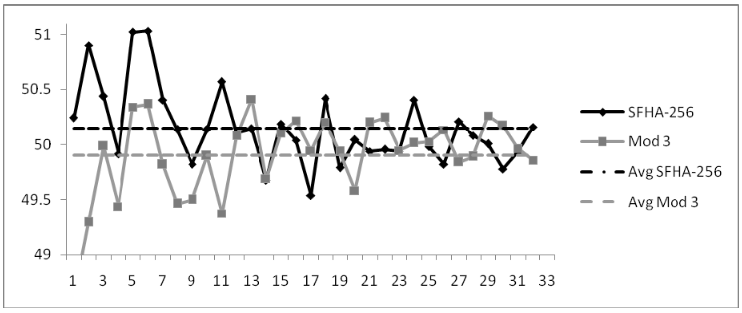

Fig. 16. Avalanche Effect (Percentage points vs. Message length)

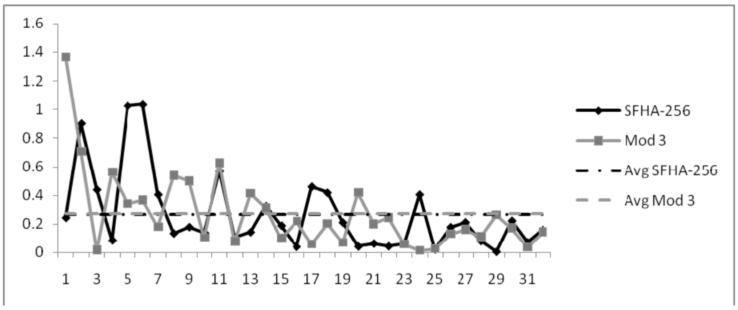

Fig. 17. Avalanche Effect - Deviation (Percentage points vs Message length)

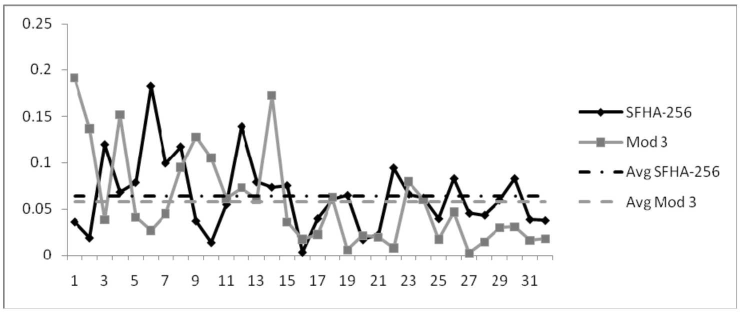

Fig. 18. Uniform Distribution Test - Deviation (Percentage points vs. Message length)

Future Work

The following areas can be pursued in the future to further improve upon the algorithm and its performance:

- Further modifications can be made to the ZPAD function or it could be completely replaced with another alternative expansion scheme.
- Different functions can be experimented with the accumulation, as in the second modification, to determine the most suitable candidate for the same.
- The running time that was increased due to a significant addition in the design scheme of the third modification can be decreased.
- The implementation of the hash function could be made to run in parallel instead of the serial order it currently follows.
- A comprehensive study of the performance of the email system utilizing the modified hash function can be made.

5. Conclusion

The analysis of the performance parameters that were recorded for the three modifications and the unmodified algorithm show that the modified algorithms have better characteristics as compared to the original unmodified Secure and Fast Hashing Algorithm.

From the data obtained experimentally and the subsequent analysis of the same, it can be conclusively said that the three modifications improve upon the original design of the SFHA-256 hashing algorithm.

References

[1] D. Davies and W. L. Price, *The application of digital signatures based on public key cryptosystems,* NPL Report DNACS 39/80, 1980.

[2] I.B. Damg°ard, "*Collision free hash functions and public key signature schemes*," *Advances in Cryptology, Proc. Eurocrypt'87*, LNCS 304, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp.203-216.

[3] C. Mitchell, D. Rush, and M. Walker, "A remark on hash functions for message authentication," *Computers & Security*, vol8, 1989, pp.517-524.

[4] M. Naor and M. Yung, "*Universal one-way hash functions and their cryptographic applications*," *Proc. 21st ACM Symposium on the Theory of Computing*, 1990, pp.387-394.
[5] http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html
[6] T. ElGamal , "A public key cryptosystem and a signature scheme based on discrete logarithms". *IEEE Transaction of Information Theory*, vol.31 (4), 1985, pp.469-472.
[7] Accredited Standards Committee X9, American National Standard X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), 2005.
[8] Certicom Research, Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography, Version 2.0, 2009.
[9] Hassan. M. Elkamchouchi, Mohamed E. Nasr, Roayat Ismail Abdelfatah, "*A New Secure and Fast Hashing Algorithm (SFHA-256)*", *25th National Radio Science Conference*, 2008.
[10] Dan Boneh, Matthew Franklin, "Identity-Based Encryption from the Weil Pairing", *SIAM Journal of Computing*, vol32(3), 2001, pp.586-615.

Siddharth Agarwal
He received a B.Tech degree in Computer Science and Engineering from the National Institute of Technology, Warangal. His research interests are Cryptography and network security.

Abhinav Rungta
He received a B.Tech degree in Computer Science and Engineering from the National Institute of Technology, Warangal. His research interests are Cryptography and network security.

R.Padmavathy
She received a PhD degree from University of Hyderabad, India. At present she is working as a faculty member at the National Institute of Technology, Warangal. Her research interests include Information security, Cryptology, and Network security.

Mayank Shankar
He received a B.Tech degree in Computer Science and Engineering from the National Institute of Technology, Warangal. His research interests are Cryptography and network security.

Nipun Rajan
He received a B.Tech degree in Computer Science and Engineering from the National Institute of Technology, Warangal. His research interests are Cryptography and network security.