International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6S, August 2019

Design Of Random Oracle For Block Iterated One-Way Ciphers Through Polynomial Functions

P. Karthik, P. Shanthibala

Abstract— ***The security aspect on provably secure hash function relies upon the design principles of Random Oracle (RO). Poor design principles make the said function vulnerable to hash attacks. The conventional keyless provably secure hash functions MD-5, SHA-2 and SHA-3 use RO or Sponge principles for the design and construction of hash function. They use bitwise operators AND, OR, XOR and modulo arithmetic for processing the input blocks. These operators are simple to use and they are efficient in terms of achieving quick response time. At the same time the repeated use of them in the input blocks may invite hash collisions. The proposed design advocates the use of multi variable higher-order polynomial function for the design of round function in RO. The new design paradigm derogates the use of bitwise operators at block level processing and hardens the internal structure of RO with higher- order polynomial function to ensure better security. The results prove that, the new prototype helps the block iterated hash function to exhibit strong random behavior even for a small bit flip in the input. Therefore, performing differential analysis on the proposed design is very hard than ever before.***

Index terms: Block Iterated Hash Functions, Polynomials for sparse mapping of hashes, Random Oracle design through Polynomials, Enhancing hash security using polynomials, Avalanche effect in hash design using polynomials, Provably Secure Block Iterated Hash Functions

I. INTRODUCTION

Data security is a prime concern in modern computing for the impact it creates on individuals' lives due to identity theft [1][2]. The user information is purely personal and sensitive and importantly the remote data breaching have to be addressed properly with lot of care. This is not an easy task to perform because the data is not under the physical control of the owner. Provably secure keyless function finds an appropriate solution to this problem. It quickly recognizes the data breaching in a remote server and finds a quick remedy. Provably secure hash function takes arbitrary length input string {0,1}m and maps it to a fixed size output {0,1}n[3][4]. The values such as m and n represent the size of input and output respectively. The size of the input m is normally greater than the output size n. As a result, the hash function performs compression on input data in order to produce fixed length output. The hash output is otherwise known to be in different names as such hash or digest or Modification Detection Code (MDC), and it is generally presumed as a short form of representation of larger binary string. Under this stage, the input and output mapping of cryptographic hash function comes one to one. Actually the achievement of one to one mapping on compression function is not possible due to Pigeon-Hole principle [5]. Cryptographic hash function tries to over overcome this problem by performing sparse mapping over the output domain. This property enables the hash function, does not have correlation between outputs correspond to different inputs. Therefore, performing differential analysis against this algorithm will be difficult. Figure-1 demonstrates the mappings of cryptographic hash function.

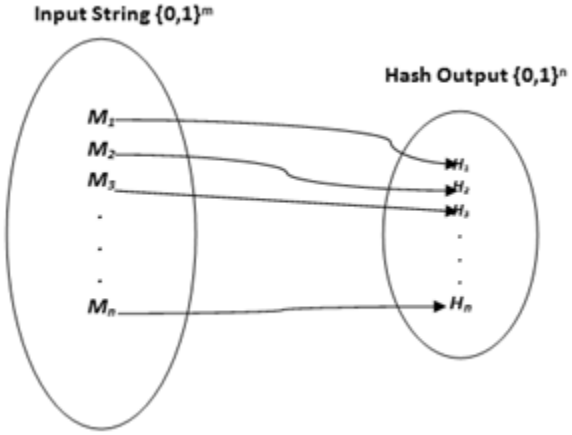

Fig: 1 One to One mapping of cryptographic hash function

According to Bart Preneel (1993b) and Timo Bartkewitz (2009), the hash function should adhere some of the key properties of provably secure hash function for cryptographic use [6][7]. The properties are given as follows:

- Collision Resistance- For any two different messages x and y , H(x) # H(y).
- Pre-Image Resistance- For a given hash value H(x) an adversary should not find y such that x # y and H(x)=H(y)

Manuscript published on 30 August 2019.
* Correspondence Author (s)
P.Karthik, Research Scholar, Department of Computer Science, School of Engineering and Technology, Pondicherry University, Puducherry, Tamilnadu, India.(email Id: thooralonly@gmail.com)
Dr. P. Shanthibala Assistant Professor, Department of Computer Science, School of Engineering and Technology, Pondicherry University,Puducherry, Tamilnadu, India (email Id: shanthibala.cs@gmail.com)

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC-BY-NC-ND license http://creativecommons.org/licenses/by-nc-nd/4.0/

- Second Pre-Image Resistance- It is computationally infeasible for an adversary to find the second pre-image y of x such that, x#y and H(x) =H(y).

The proposed design hereafter referred as Block Iterated One-way Polynomial Cipher (BIOPC), identifies the structural weakness of the conventional algorithms MD5, SHA-160, SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384 and SHA3-512. It advocates the use of polynomial function for enhancing the security of block iterated one way hash function. BIOPC works on 1024 bits blocks to produce 512 bits output. The blocks are applied to higher-order polynomial function of degree 128, to produce intermediate results of sizes between 270 and 290 nibbles using the wide-pipe principle as suggested by Stefan Lucks (2004)[8]. The intermediate outputs are then truncated to 256 nibbles and are subsequently applied to the next block for modifying the block elements. This process will be continued till all the blocks are processed. The final hash of BIOPC is the hash output of the last block.

The paper is summarized as follows: Section 2 describes the related work, Section 3 highlights the design principles of BIOPC, Section 4 advocates the merits of BIOPC through experimental analysis, Section 5 deals with discussion and Section 6 conveys concluding remarks of future enhancements.

II. RELATED WORK

It was first time in history Ivan Bjerre Damgard and Ralph C Merkle, the colonizers of provably secure hash function, proved the world provably secure properties through mathematical deductions[9][10]. The conventional provably secure keyless hash functions such as MD4, MD5, SHA-224, SHA-256, SHA-384 and SHA-512 used Merkle-Damgard(MD) construction principles for the design of hash function.

Ronald Rivest invented MD4 algorithm using MD construction principles in 1992 [11]. Rivest used 3 auxiliary functions with 4 initialization vectors (IVs) of 32 bits each to produce 128 bits hash output. MD4 was operated on 512 bits blocks to produce 128 bits hash output. In the same year, Rivest came up with another algorithm called MD5 [12]. The new MD5 used 4 auxiliary functions instead of 3 and it produced 128 bits hash output. It was believed that, MD5 was more secure than its predecessor MD4. During the year 2001, D Eastlake et al., discovered a new 160 bits SHA-1 algorithm [13]. SHA-1 used 5 auxiliary functions of 32 bits each to produce 160 bits hash output. The use of MD4, MD5 and SHA-1 was prohibited from cryptographic use after 2004. This was because, they were identified as producers of hash collisions [14][15].

During the year 2004-2005, series of attacks were reported against SHA-1 algorithm. SHA-1 was previously approved by the US based National Institute of Standards and Technology (NIST), as a new standard for hash functions that time. The attacks on SHA-1 forced NIST to set a new standard for provably secure hash function through open competition [16]. The open competition was announced during 2007 and after several rounds of scrutiny, NIST was selected Keccak as the winner of the competition in October 2012 [17]. SHA-3 has been a new standard for all the cryptographic hash functions from October 2012 [18].

III. DESIGN PRINCIPLES

BIOPC applies higher-order polynomial function into the design of RO using the principles of block iterated hash functions. Polynomial function of higher-degree is chosen because, finding roots of higher-degree polynomial function is extremely difficult and it does not have standard solution [19]. This design uses two variables polynomial function as follows:

$$P(x,y)= C_0x^n+C_1 x^{n-1} y^1+....+C_ny^n \text{-----> Eq(1)}$$

The use of two variables in the polynomial function prevents hash collisions due to truncation of trialing bits in the hash output. BIOPC generates hash values in two different levels as follows:

- Pre-Processing
- Hash generation through Polynomial Round Function

A. *Pre-Processing*

Pre-Processing is the first level of hash generation process. In this level the input message is pre-processed at first, before it is applied into the polynomial function for block level processing. This is performed through MD-strengthening at tail end of the message to be processed [22]. Here, the block size is fixed as 1024 bits. BIOPC differs from conventional MD-Strengthening by not reserving the last 64 bits for storing the length attribute of the input message. Rather, it performs MD-Strengthening by dynamically calculating the padding bits at runtime. BIOPC uses Equation 2, to perform message padding as follows:

$$padlen=128-(dblen+blen+1) \ MOD \ 128 \text{ ------> Eq(2)}$$

Here, Dblen represents the number of elements present in the input string and blen implies number of bytes which are required to represent the length attribute of the input string. The first byte of the padding is set with 0x80 and the remaining bytes are filled with zeros. The byte values of blen are stored at the tail end of the last block. The dynamic padding of BIOPC enables it to process messages of size greater than 264 bits. After message being strengthened, BIOPC applies IVs to modify the input elements. This step is being performed to introduce near random behavior for the RO model. BIOPC uses an array of 8 IVs to process the message blocks. Each element of the IV is of 128 bits in length and the concatenation of all IVs yield to 1024 bits. The concatenated IVs are finally XORed with message block to produce an unintelligible junk of input bytes of 1024 bits per block. Table 1 shows the IV elements of BIOPC

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

	INITIALIZATION VECTOR	Bits
1	426a1ef027197015fce2c52fbfd6158b	128
2	9a5b10b097c823de31d7ae698fdf17f1	128
3	1c7f9b9dc21dc2f164e7c6302742f791	128
4	2c42ccdd727f0051c8d49997d905ac73	128
5	2721bd77b4e7862f68a711e2f61d9329	128
6	2b7e7f63b2c382592e194c394af41f5f	128
7	3423d0bb18ec36dd36b6a74ebf799213	128
8	6a90f18822175cf43721dbe3da9df414	128

Table 1: IVs Used in BIOPC

The Original String Value is :

Received: from SLUAVA.slu.edu (sluvca.slu.edu) by Sierra.Stanford.EDU with SMTP (5.67b8/25-eef) id AA11430; Tue, 30 Aug 1994 Length is : 128 Bytes

Byte Values Before MD Strengthening

The Length of the data is : 128

13 13 10 82 101 99 101 105 118 101 100 58 32 102 114 111 109 32 83 76 85 65 86 65 46 115 108 117 46 101 100 117 32 40 115 108 117 118 99 97 46 115 108 117 46 101 100 117 41 32 98 121 32 83 105 101 114 97 46 83 116 97 110 102 111 114 100 46 69 68 85 32 119 105 116 104 32 83 77 84 80 32 40 53 46 54 55 98 56 47 50 53 45 101 101 102 41 32 105 100 32 65 65 49 49 52 51 48 59 32 84 117 101 44 32 51 48 32 65 117 103 32 49 57 57 52 32

The Length of the data is : 128

79 103 20 -94 66 122 21 124 -118 -121 -95 21 -97 -80 103 -28 -9 123 67 -4 -62 -119 117 -97 31 -92 -62 28 -95 -70 115 -124 60 87 -24 -15 -73 107 -95 -112 74 -108 -86 69 9 39 -109 -28 5 98 -82 -92 82 44 105 52 -70 -90 -8 -71 -118 113 -51 29 65 78 -49 19 -102 -94 -62 122 72 -48 120 -106 -98 61 -64 100 127 46 95 75 -121 -1 9 -76 110 76 33 99 11 127 -39 122 58 82 10 -16 -46 124 -52 119 -100 7 -121 -109 125 -113 66 -78 71 31 -11 -35 -88 17 39 124 -75 66 70 -5 -46 -29 -92 -64 52 -46 30 49 4 66 -21 -98 -122 -28 59 124 123 83 -66 -126 -115 77 67 -34 4 -29 46 2 -65 -100 88 -91 -9 -6 -62 -79 104 75 98 22 18 -7 4 123 -58 58 -11 -51 49 -5 -30 -2 51 -113 -13 115 -72 67 -72 94 44 -100 -8 -58 4 -24 94 -14 35 -120 89 -101 -82 79 -32 10 57 26 -109 50 -5 32 -75 -114 101 -28 55 -82 -10 -100 -16 -59 -19 20 -30 60 94 -61 -78 101 36 111 -49 -20 118 88 112 75 37 -61 41 -121 41 94 -125 -21 -27 -124 122 23 99 29 -122 -37 -90 -42 -44 -23 -41 -17 50 66 -26

Fig 2: Elements of the input array after MD strengthening

The junk of bits is produced by BIOPC for the sample input has been shown in Figure 2 for analysis. The elements of IVs are circularly rotated for every block to be processed. Each element in the IV array is circularly rotated in one position on right and the IV element itself is circularly rotated to 5 bits on left. The objective of the rotation is to generate different IVs for every individual block to be processed.

B. Hash Generation Through

Polynomial Round Function

This is a second level of hash generative process. Level-1 converts the input string into unintelligible junk of byte array such a way that, the size of the array appears to be in multiples of block size. The individual block of strengthened array is then applied to two variables polynomial function at degree 128, in order to establish intermediate hash outputs. Intermediate hashes are then being used here to modify the next block elements to be processed. To perform this operation, the polynomial

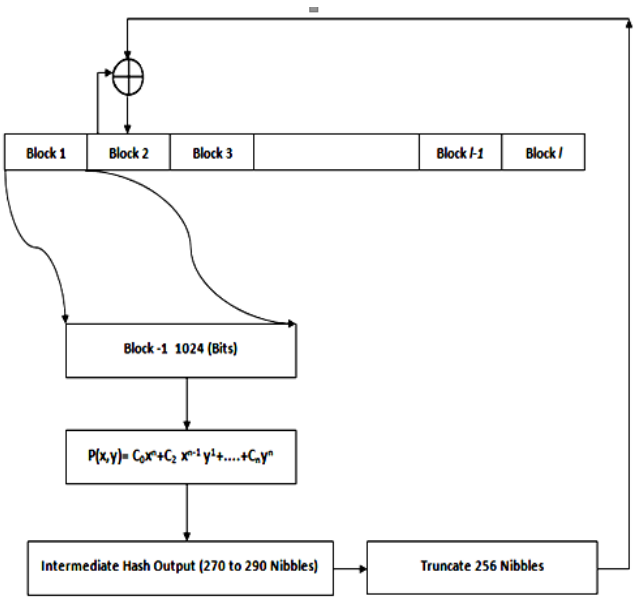

Fig 3: Working principle of polynomial Round Function

round function uses 2 prime numbers in place of the polynomial variables x and y . It uses the bytes of the individual block as the co-efficient for the polynomial function. The intermediate hash output of the polynomial function usually contains 270-290 nibbles. In order to achieve such size, the polynomial variables are assigned values in the range between 300 and 600. Figure 3 illustrates the working principle of polynomial round function for Block-1.

Randomness is a desirable property of provably secure hash function. BIOPC achieves randomness in the intermediate hashes by, shuffling the nibbles of the intermediate output evenly across the length of the hash. This operation is performed at 2 steps namely truncation and shuffling.

a.. Truncation

```
PROCEDURE TRUNCATE -256 (IHash: ARRAY [1...290] )
  BEGIN
  // Count the nibbles of IHash array.
  hlen= Length of IHash;
  //Define a character array for 1st and 2nd haves.
  hc: ARRAY[1...128]
  hc1: ARRAY[1...128] of characters.
  thc: ARRAY[1...256] of characters.
  int mid=hlen/2; //Calculate the mid position
  // Copy the first 128 bytes from mid to hlen
  for(int i=mid+1,j=0;i<hlen &&j<128;i++,j++){
     hc1[j]=IHash[i]; }
  // Copy the first 128 bytes from mid to 0
  for(int i=mid,j=0;i>0 && j<128; i--,j++) {
        hc[j]=IHash[i]; }
  END-TRUNCATE-256;
```

Fig 4: Truncation

© Copyright: All rights reserved.

Truncation module truncates the required block output size of 256 nibbles by, taking two halves of 128 nibbles each from the mid position of the intermediate hash output on either direction. The truncated two halves then store up separately in different arrays for shuffling. The pseudo code given in Figure 4 illustrates the working principle of this module.

b. Shuffling

Under Shuffling, the truncated halves of the intermediate hash output are shuffled into the entire length of the hash output. BIOPC performs shuffling using the nibbles of the intermediate hash halves. The pseudo code given in Figure 5 demonstrates the internal shuffling of intermediate hashes at block level. The final hash of the BIOPC is obtained from the intermediate result of block Ml. In this block, the two halves hardly shuffle and they get XORed to produce 128 nibbles of hash output.

```
PROCEDURE SHUFFLING(hc: ARRAY[1...128], hc1:
ARRAY [1...128])

BEGIN

    // For Every Iteration Copy 2 nibbles to the array by
    //taking one nibble from each halve at a time

    for(int x=0,y=0;x<256 && y<128;x=x+2,y=y+1)
    {
      thc[x]=hc1[y];
      thc[x+1]=hc[y];
    }

END SHUFFLING;
```

Fig 5: Shuffling

IV. EXPERIMENTAL ANALYSIS & RESULTS

In experimental analysis BIOPC is subjected to exhaustive experimental analysis on the key attributes of provably secure keyless hash function namely Collision resistance, Pre-Image resistance and Second pre-image resistance. The desirable property of strict avalanche criterion puts on test by comparing the nibbles with the reference value for possible match. At end BIOPC is compared to existing provably secure keyless hash functions SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384 and SHA3 -512, using various data sets. All the experiments are conducted for analysis using Intel(R) Core(TM) i3 6006U CPU @ 2.00 GHz 2.00 GHz processor with Windows-10 64 bits operating system. The algorithm is fully designed and tested using JDK 10.0.1.

A. ANALYSIS ON COLLISION AND PRE-IMAGE RESISTANCE (MODIFYING THE INDIVIDUAL BYTES FOR ALL POSSIBLE VALUES)

On the basis of analysis it has been identified that, BIOPC is not producing collisions when it is subjected to collision resistance and pre-image resistance by modifying the individual bytes of the input string for all possible values.

S.No	No.Of Input Blocks Taken	Data Size (Bytes)	Data Size (Bits)	No.Of Hash Comparisons made (Worst Case)	No of Hash Collisions Found	Findings
1	< 1 Block	64	512	16320	0	BIOPC did not Produce any Collision and shows Strong Collision and Pre-Image Resistance Properties.
2	1	128	1024	32640	0	
3	2	256	2048	65280	0	
4	4	512	4096	130560	0	
5	8	1024	8192	261120	0	
6	10	1280	10240	326400	0	
7	12	1536	12288	391680	0	
8	16	2048	16384	522240	0	
9	24	3072	24576	783360	0	
10	32	4096	32768	1044480	0	
Total Number of Hash Comparisons made				3574080		

Table 2: Response on Collision and Pre-Image resistance test for all possible byte values

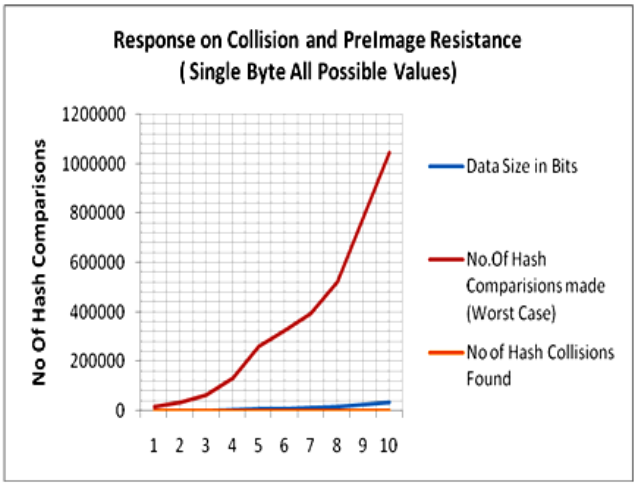

Fig 6: Graphical response on Collision and Pre-Image resistance test

Table 2 presents the response of this test for various data sets. Figure 6 reflects the graphical response of Table-2 entries. Figure 7 shows the sample screen shot for collision and pre-image resistance test, for all possible values of individual byte on 4K data.

Fig 7: Sample response on Collision and Pre-Image resistance test for 4K data

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

B. ANALYSIS ON COLLISION AND PRE-IMAGE RESISTANCE (FOR ALL POSSIBLE TWO BYTES INTERCHANGES)

A test on Collision resistance and Pre-Image resistance is conducted through all possible two bytes interchanges for the given input string. Table 3 presents the response of BIOPC on this test. Figure 8 graphically represents the outcome of this test. Figure 9 shows the sample response of BIOPC on all possible two bytes interchanging of 1280 bytes of input data.

S.No	No of Input Blocks	Data Size (Bytes)	Data Size (Bits)	No.Of Hash Comparisons made (Worst Case)	No of Hash Collisions Found	Findings
1	< 1 Block	64	512	2016	0	BIOPC did not produce any Collisions and shows Strong Collision and Pre-Image Resistance Properties.
2	I	128	1024	8128	0	
3	2	256	2048	32640	0	
4	3	384	3072	73536	0	
5	4	512	4096	130816	0	
6	5	640	5120	204480	0	
7	8	1024	8192	523776	0	
8	10	1280	10240	818560	0	
9	12	1536	12288	1178880	0	
10	16	2048	16384	2096128	0	
Total Number of Hash Comparisons made				5068960		

Table 3: Response on Collision and Pre-Image resistance test for all two bytes interchanges

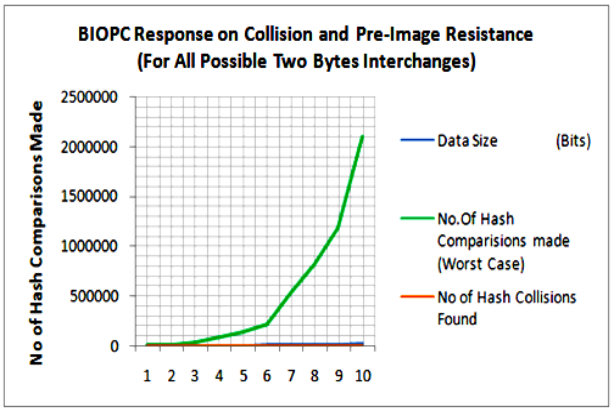

Fig 8: Graphical response on all possible two bytes interchanges

Fig 9: Sample response on Collision and Pre-Image resistance test for 1280 bytes

C. ANALYSIS ON COLLISION AND SECOND PRE-IMAGE RESISTANCE

In this test, BIOPC is supplied with two randomly selected input strings from the domain of 100 thousands of random samples. The sample input strings are varying in length in the range between 1 and 100 thousands of bytes. The response of the hash function is recorded for hash matches.

S.No	No. of Samples Taken	No.Of Hash Comparisons Made	Collision Count	Findings
1	1000	500	0	BIOPC did not produce any Second Preimage Collision and shows strong Second Pre-Image Resistance.
2	2000	1000	0	
3	3000	1500	0	
4	4000	2000	0	
5	5000	2500	0	
6	8000	4000	0	
7	10000	5000	0	
8	20000	10000	0	
9	30000	15000	0	
10	100000	50000	0	
Total	**183000**	**91500**	**0**	

Table 4: BIOPC response on Collision and Second Pre-Image resistance test

Table 4 shows the response of Collision and Second Pre-Image resistance test. Figure 10 presents the graphical response of BIOPC for Collision and Second Pre-Image resistance test.

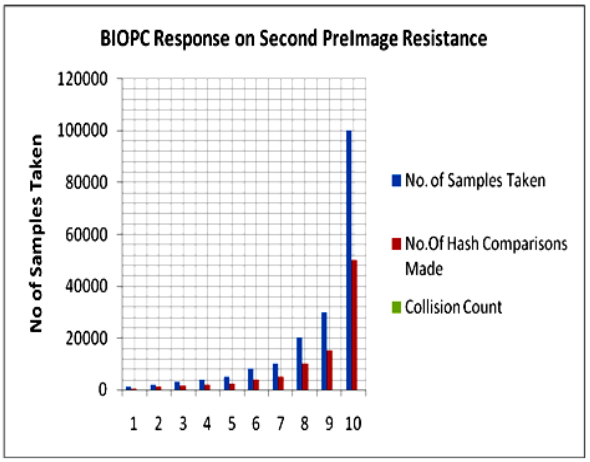

Fig 10: Graphical response on Collision and Second Pre-Image resistance test.

© Copyright: All rights reserved.

Figure 11 shows the sample response of this test on 30000 randomly selected samples.

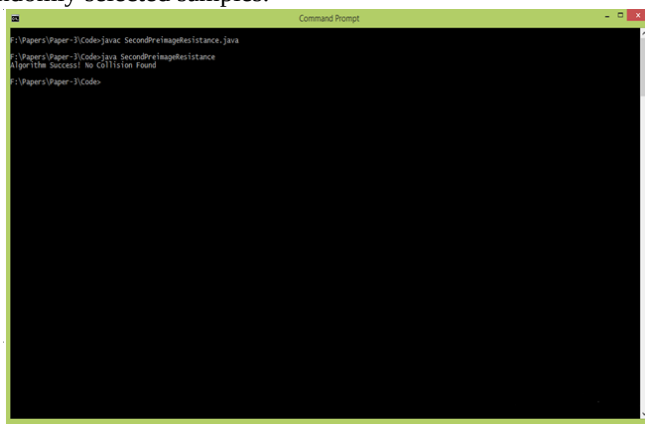

Fig 11: Response of BIOPC on Collision and Second Pre-Image resistance test for 30000 samples

D. AVALANCHE RESPONSE OF BIOPC

This is a desirable property of cryptographic hash function. According to the property, a small change in the input string affects substantial number of output bits of the hash function [20]. A.F. Webster et al. (1985) suggested that, hash function would meet strict avalanche criterion for it to be considered for cryptographic use. To realize this property, the hash function should affect more than 50 % of the output bits, for a small change in the input string [21].

S.No	Data Size in Bytes	Avalanche Effect of BIOPC						
		No. of Bytes Modified						
		1 B	2B	4B	8B	10B	20B	>20B
1	64 B	90.63	95.31	93.75	96.86	94.53	94.53	95.31
2	128 B	93.75	93.75	94.53	94.53	92.18	91.4	92.97
3	256 B	94.53	96.86	95.31	95.31	89.06	93.75	94.53
4	512 B	97.65	95.31	92.97	91.41	90.63	94.53	94.53
5	1k	92.18	98.43	93.75	95.31	96.09	92.19	93.75
6	2k	94.53	92.19	96.09	93.75	95.31	95.31	92.97
7	4k	93.75	93.75	92.97	95.31	92.97	92.19	92.19
8	8k	95.31	95.31	97.65	93.75	95.31	92.19	92.97
9	16k	93.75	95.31	96.09	92.97	95.31	96.88	91.41
10	20 k	96.88	92.97	96.875	96.875	92.97	93.75	89.84
11	25k	95.31	91.4	94.53	92.97	96.88	96.09	96.88
12	> 25k	95.31	96.09	91.4	94.53	94.53	95.31	91.41
Average %		94.47	94.72	94.66	94.46	93.81	94.01	93.23

Table 5: Avalanche response of BIOPC

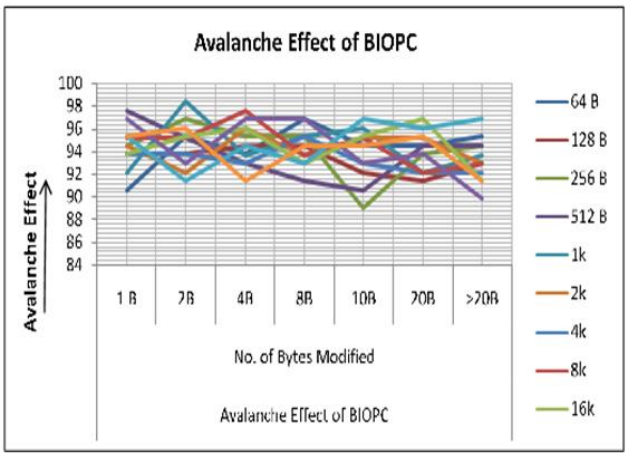

Fig 12: Graphical view of avalanche response

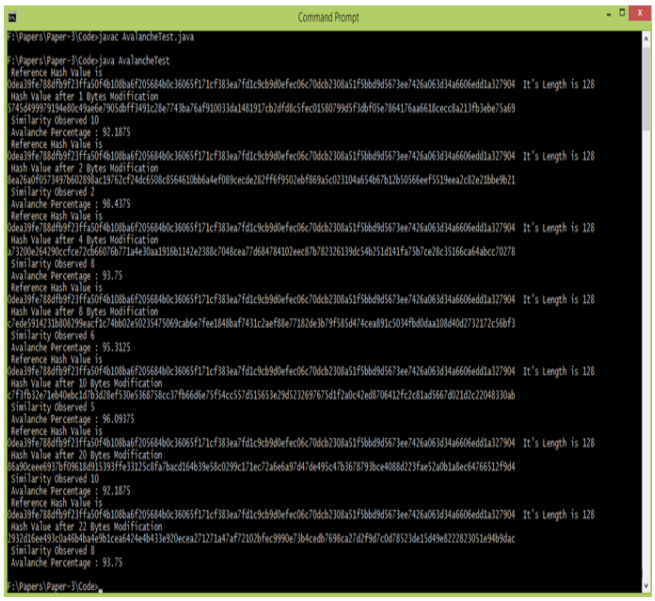

Fig 13: Avalanche response of BIOPC for 1K data

Table 5 presents the response of BIOPC on avalanche effect. Figure 12 shows the graphical response of BIOPC. Figure 13 confronts the sample response of the algorithm on avalanche effect.

E. RUNTIME ANALYSIS

In this experiment the runtime performance of BIOPC is compared to the conventional keyless cryptographic functions SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384 and SHA3 -512. The response time of the aforementioned algorithms is derived from the mean of 25 samples from each data set. Table 6 presents the runtime response of conventional keyless hash functions in Milliseconds. Figure 14 showcases the runtime performance of BIOPC in graphical format. Figure 15 gives the sample response of BIOPC for 2K data.

Table 6: Runtime time analysis for various data sets

S.No	Data Size in Bytes	Runtime Comparison of Keyless Hash Function in Milliseconds								
		SHA-2 Family				SHA-3 Family				BIOPC-512
		SHA-224	SHA-256	SHA-384	SHA-512	SHA3-224	SHA3-256	SHA3-384	SHA3-512	
1	64 B	1.84	0	0	0	9.36	0.6	0.64	0	7.48
2	128 B	1.24	0.6	0.64	0	10	0.6	0.64	0	8.08
3	256 B	1.24	0	0.64	0	11.24	0.64	0	0.64	12.48
4	512 B	0.64	0	0	0.64	8.72	0.6	0	0	14.4
5	1 K	1.24	0.64	0.64	0	8.76	0	1.28	0	19.96
6	2 K	1.88	0.6	0	0.64	9.36	0.64	0	0	30.6
7	4 K	1.28	0	0.64	0.64	10	0	0	0.64	49.92
8	6 K	1.24	0	1.28	1.2	11.28	0.64	0	1.88	67.6
9	8 K	1.2	0	0	1.04	10.64	0.6	1.28	0	82.48
10	10 K	0.64	0	0.6	1.28	10.04	0	0	1.24	95.12
11	20K	2.52	0.64	0.64	1.24	10.64	0	1.24	2.48	138.16

Retrieval Number: F11960886S19/2019©BEIESP
DOI:10.35940/ijeat.F1196.0886S19
Journal Website: www.ijeat.org

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.

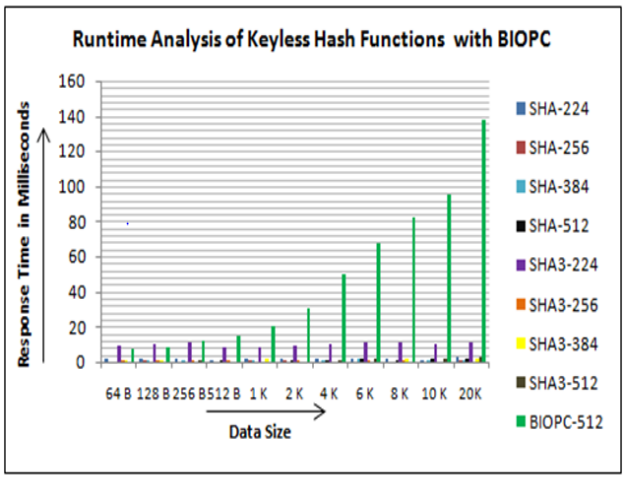

Fig 14: Graphical view on runtime time analysis

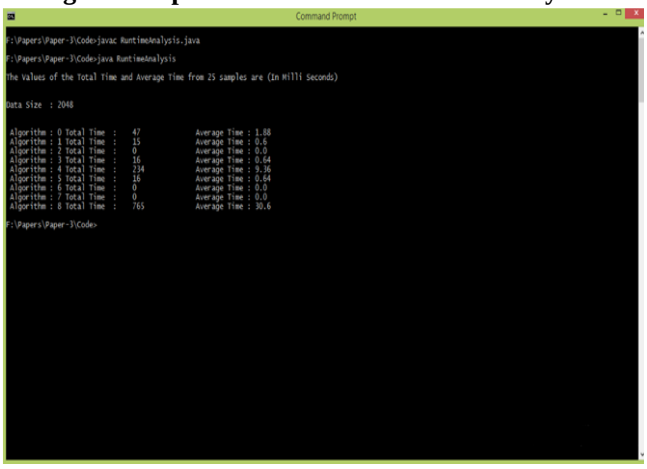

Fig 15: Sample response of BIOPC on runtime analysis of 2K data

V. DISCUSSION

The conventional algorithms SHA-224, SHA-256, SHA-384, SHA-512, SHA3-224, SHA3-256, SHA3-384 and SHA3-512 use simple bitwise AND, OR, XOR, Complement and modulo arithmetic operators, for processing the data at block level. These operators help the said algorithms to work efficiently with quick response time. At the same time, they are traceable when parameters involved in the block ciphers are known. Hence, they are poor in terms of security and vulnerable to block level attacks. According to Bart Preneel et al., the iterated hash functions that perform bitwise operations at block level are vulnerable to direct attack, permutation attack, forward attack, backward attack and fixed point attack, if any one of the input parameters and the output is identifiable [4]. The partial breaking of SHA-2 and SHA-3 algorithms witness this fact and yet the said algorithms are vulnerable to block level attacks [23][24][25]. BIOPC prevents the block level attacks by generating intermediate hashes through two variables polynomial function. The higher-order polynomial function is the natural one-way function and the use of it at block level strongly prevents any adversary to decipher the codes generated by the blocks. In addition to it, BIOPC uses only the safe schemes among the 64 available schemes [4]. It is sound enough to say the combined merits of the two variables polynomial function and the safe schemes virtually block the entire avenue for an adversary to launch any attack at block level.

The exhaustive experimental analysis which is performed on Collision resistance, Pre-Image resistance and Second Pre-Image resistance proves that, BIOPC is a provably secure hash function. Adding to the above the avalanche test that is conducted on BIOPC shows that, the average worst case of change in the output bits is at 93.23 %. This is far and away the strict avalanche criterion for provably secure hash function. Despite these merits, the runtime performance of BIOPC is low and it linearly increases with respect to data size. This happens due to the use of higher-order polynomial function. The research is still in progress at our side and we try to expedite possibilities to see through possible reconstruction of round function without compromising the security.

REFERENCES

1. Romanosky, Sasha, Rahul Telang, and Alessandro Acquisti. "Do data breach disclosure laws reduce identity theft?." Journal of Policy Analysis and Management 30.2 (2011): 256-286.
2. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.
3. Buchmann, Johannes A. "Cryptographic hash functions." Introduction to Cryptography. Springer, New York, NY, 2004. 235-248.
4. Preneel, Bart, René Govaerts, and Joos Vandewalle. "Hash functions based on block ciphers: A synthetic approach." Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1993a.
5. Ajtai, Miklós. "The complexity of the pigeonhole principle." Foundations of Computer Science, 1988., 29th Annual Symposium on. IEEE, 1988.
6. Preneel, Bart. Analysis and design of cryptographic hash functions. Diss. Katholieke Universiteit te Leuven, 1993b.
7. Bartkewitz, Timo. "Building hash functions from block ciphers, their security and implementation properties." Ruhr-University Bochum (2009).
8. Lucks, Stefan. "Design Principles for Iterated Hash Functions." IACR Cryptology ePrint Archive 2004 (2004): 253.
9. Dr.S.P.Anandaraj " Research Opportunities and Challenges of a Security Concerns associated with Big Data in Cloud Computing", IEEE International conference on ISMAC IoT in Social, Mobile, Analytics and Cloud published in IEEE Explore, DOI: 10.1109/I-SMAC.2017.8058278, Electronic ISBN: 978-1-5090-3243-3,Print ISBN: 978-1-5090-3242-6, 5 on 10th to 11th Feb, 2017
10. Damgård, Ivan Bjerre. "A design principle for hash functions." Conference on the Theory and Application of Cryptology. Springer, New York, NY, 1989.
11. Merkle, Ralph c. "one way hash functions and des." conference on the theory and application of cryptology. Springer, new york, ny, 1989.
12. Rivest, Ronald. The MD4 message-digest algorithm. No. RFC 1320. 1992.
13. Rivest, Ronald. The MD5 message-digest algorithm. No. RFC 1321. 1992.
14. Eastlake 3rd, D., and Paul Jones. US secure hash algorithm 1 (SHA1). No. RFC 3174. 2001.

© Copyright: All rights reserved.

15. Wang, Xiaoyun, et al. "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD." IACR Cryptology ePrint Archive 2004 (2004): 199.
16. Wang, Xiaoyun, Yiqun Lisa Yin, and Hongbo Yu. "Finding collisions in the full SHA-1." Annual international cryptology conference. Springer, Berlin, Heidelberg, 2005.
17. https://csrc.nist.gov/projects/hash-functions/sha-3-project
18. Bertoni, Guido, et al. "Keccak." Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2013.
19. Kam, John B., and George I. Davida. "Structured design of substitution-permutation encryption networks." IEEE Transactions on Computers 10 (1979): 747-753.
20. Dr. S.P.Anandaraj, "Spot Detection For Morphological Convolution Using DNA Microarray" , International Journal of Pure and Applied Mathematics, ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version), SCIE 2013 Impact Factor= 7.19, Vol.118, Issue 14,pp.1-7,Feb 2018
21. Lai, Xucjia, and James L. Massey. "Hash functions based on block ciphers." Workshop on the Theory and Application of of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1992.
22. Sanadhya, Somitra Kumar, and Palash Sarkar. "New collision attacks against up to 24-step SHA-2." International conference on cryptology in India. Springer, Berlin, Heidelberg, 2008.
23. Aoki, Kazumaro, et al. "Preimages for step-reduced SHA-2." International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2009.
24. Dinur, Itai, Orr Dunkelman, and Adi Shamir. "Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials." International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 2013..

AUTHORS PROFILE

P. Karthik, Research Scholar, Department of Computer Science, School of Engineering and Technology, Pondicherry University, Puducherry.

Email Id: thooralonly@gmail.com

Dr. P.Shanthi Bala Assistant Professor, Department of Computer Science, School of Engineering and Technology, Pondicherry University, Puducherry.

Email Id: shanthibala.cs@gmail.com

© Copyright: All rights reserved.