# R-HASH: HASH FUNCTION USING RANDOM QUADRATIC POLYNOMIALS OVER GF(2)

Dhananjoy Dey[1], Noopur Shrotriya[1], Indranath Sengupta[2]

[1] SAG, Metcalfe House, Delhi-110 054, INDIA.
{dhananjoydey, noopurshrotriya}@sag.drdo.in

[2]Department of Mathematics, Jadavpur University, Kolkata,
WB 700 032, INDIA.
sengupta.indranath@gmail.com

## ABSTRACT

*In this paper we describe an improved version of HF-hash [7] viz. R-hash: Hash Function Using Random Quadratic Polynomials Over GF(2). The compression function of HF-hash consists of 32 polynomials with 64 variables over GF(2), which were taken from the first 32 polynomials of HFE challenge-1 by forcing last 16 variables as 0. The mode operation used in computing HF-hash was Merkle-Damgard. We have randomly selected 32 quadratic non-homogeneous polynomials having 64 variables over GF(2) in case of R-hash to improve the security of the compression function used in HF-hash. In designing R-hash, we have also changed the mode operation used in HF-hash, because of the theoretical weakness found in the Merkle-Damgard construction.*

*In this paper we will prove that R-hash is more secure than HF-hash and SHA-256 as well as we will show that it is also faster than HF-hash.*

## KEYWORDS

*Dedicated hash functions, differential attack, MQ problem, preimage attack.*

## 1. INTRODUCTION

After recent cryptanalytic attack on MD5 [2][15] [28] [31] and SHA-1 [5] [6] [23] [24] [29], the security of their successor, SHA-2 family [21], against all kinds of cryptanalytic attacks has become an important issue. Although many theoretical attacks [10], [18], [19], [20], [12], [26] on the reduced round of SHA-256 have been published during 2003 to 2008, but there is still no threat to the security of SHA-256. In 2007 NIST announced the SHA-3 competition [22] for selecting a new hash function, which would resist length extension attack and other theoretical weakness of Merkle-Damgard construction. The design principles of all submitted hash functions for SHA-3 competition are different from Merkle-Damgard construction. We can broadly categorise all the submitted hash functions according to their design principle in the following way: balanced Feistel network, unbalanced Feistel network, wide pipe design, key schedule, MDS matrix, output transformation, S-box and feedback register. NIST has already declared the

DOI : 10.5121/ijcsit.2012.4612 

SHA-3 hash function on 02 Oct 2012. NIST has selected Keccak [3] as the SHA-3 hash from the five SHA-3 finalists, viz., Blake [1], Grøstl [9], JH [27], Keccak [3] and Skein [8].

The compression function of HF-hash [7] was designed in such a way that it consists of the first 32 polynomials with first 64 variables taken from the polynomials of HFE challenge-1 by forcing last 16 variables of the 80 variables to 0. A new hash function *R-hash* has been designed by modifying *HF-hash* by taking random quadratic polynomials to improve the security as well as the efficiency of *HF-hash*. The compression function of *R-hash* depends on the following well-known facts:

- It is easy to compute the values of a random set of $m$ multivariate polynomials in $n$ variables over a finite field $\boldsymbol{F}$ viz., $(p_1(x_1, \cdots, x_n), \cdots, p_m(x_1, \cdots, x_n))$ for any fixed $(x_1, \cdots, x_n) \in F^n$.
- Solving random system of polynomial equations is an NP-hard problem[*] [11].

In designing *R-hash*, we have changed the padding procedure, from the one we have applied in *HF-hash*, to increase the size of the input to the hash function. We have also changed the Merkle-Damgard construction applied in HF-hash by applying double-pipe design to remove the multicollisions attack [13], length extension attack, fixed-point attack [16] and herding attack [14].

A complete description of *R-hash* and its analysis will be presented in the following subsequent sections.

## 2. R-HASH FUNCTION

*R-hash* function can take arbitrary length ($< 2^{64}$ 512-bit block) of input and gives 256 bits output, i.e. we can write $R-hash : (Z_2^{512})^* \rightarrow Z_2^{256}$. We have designed *R-hash* by changing the compression function as well as the mode operation. The compression function consists of 32 random polynomials with 64 variables of degree 2 over *GF(2)*. Since the number of coefficients of a polynomial of degree 2 with 64 variables over *GF(2)* is at most 2081, to generate 32 random polynomials with 64 variables of degree 2 over *GF(2)* requires 66592 random bits. These bits were generated in the following way:

(i) First we compute SHA-512 of a file containing "968bb576eeb70c6def469a6b4824907c47390ac1880ef1f948d8a1539090b3af28deb91db e00f37072e0366ba29f3e11a85bc41dc2492f7126d25a1489ae2c70".

(ii) Then we apply SHA-512 to the output of the above file.

(iii) We repeat the above process 131 times.

[*] It is true even if we restrict the total degree of these polynomials to at least 2.

The hash value of a message $M$ of length $L = 512 \times l + 8r$ bits can be computed in the following manner:

**Padding:** First we append 1 to the end of the message $M$ to indicate the termination of the message. Let $k$ be the number of zeros added for padding. First, 7-bit representation of $r$ bytes is appended to the end of $k$ zeros and then 64-bit representation of $l$ blocks is placed to the end of 7-bit representation of $r$ bytes. Now $k$ will be the smallest non-negative integer satisfying the following condition:

$$8r + 1 + k + 7 + 64 \equiv 0 \bmod 512$$

$$i.e.,\ k + 8r \equiv 440 \bmod 512$$

The padding procedure is shown in Figure 1. According to this padding procedure, we can compute the hash value of a message of length $\leq 2^9 \times (2^{64} - 1)$ bits.


Figure 1: Padding Procedure

**Parsing:** Let $l'$ be the length of the padded message. Divide the padded message into $n\ (= l'/512)$ 512-bit block i.e. sixteen 32-bit words instead of 448-bit block, which is applied in *HF-hash* to improve the efficiency. Let $M^{(i)}$ denote the i^th^ block of the padded message, where $1 \leq i \leq n$ and each word of i^th^ block is denoted by $M_j^{(i)}$ for $0 \leq j \leq 15$.

**Initial Value:** Take the first 256 bits initial value i.e., eight 32-bit words from the expansion of the fractional part of $\pi$ and hexadecimal representation of these eight words are given below:

$$H_0 = 243F6A88,\ H_1 = 85A308D3,\ H_2 = 13198A2E,\ H_3 = 03707344,$$
$$H_4 = A4093822,\ H_5 = 299F31D0,\ H_6 = 082EFA98,\ H_7 = EC4E6C89.$$

Thus $IV = H_0 \| H_1 \| \cdots \| H_7$.

**Hash Computation:** For each 512-bit block $M^{(1)}, M^{(2)}, \cdots, M^{(n)}$, the following four steps are executed for all the values of $i$ from 1 to $n$.

1. **Initialization:**
   The first chaining variable $CV_0 = H_0 \| H_1 \| \cdots \| H_7$ is the 256-bit initial value $IV$.

2. **Transformation:**
   Divide the input message block into two equal parts, i.e., $M^{(i)} = L_i \| R_i$. Transform the message block in the following way:

$$L'_i \leftarrow (L_i \oplus CV_{i-1} \oplus counter_i),$$

where $counter_i = i \bmod 2^{64}$. Xoring $counter_i$ removes the fixed point attack, because as the number of blocks increases, $counter_i$ will be different for each block. Now,

$$M^{(i)} = L'_i \,\|\, R_i.$$

We divide this $M^{(i)}$ into sixteen 32-bit words $M_0^{(i)}, M_1^{(i)}, \cdots, M_{15}^{(i)}$.

3. **Iteration:**

For j = 0 to 6

(a) For k = 0 to 7

$$H_k^j \leftarrow p(M_{2k}^{(i)} \,\|\, M_{2k+1}^{(i)})$$

(b) $R'_i \leftarrow (R_i \oplus (H_0^j \,\|\, H_1^j \,\|\cdots\|\, H_7^j))$

(c) For j = 0 to 15

$$M_j^{(i)} \leftarrow M_{j+2 \bmod 16}^{(i)}$$

where $p : Z_{2^{64}} \rightarrow Z_{2^{32}}$ be a function defined by

$$p(x) = 2^{31}.p_1(x_1,\cdots,x_{64}) + 2^{30}.p_2(x_1,\cdots,x_{64}) + \cdots + 1.p_{32}(x_1,\cdots,x_{64})$$

Since any element $x \in Z_{2^{64}}$ can be represented by $x_1x_2\cdots x_{64}$, where $x_1x_2\cdots x_{64}$ denotes the binary representation of $x$ in decreasing order of their significance. $p_i(x_1,\cdots,x_{64})$ denotes the $i^{th}$ polynomial with 64 variables.

For j = 7

(a) $H_k^j \leftarrow p(M_{2k}^{(i)} \,\|\, M_{2k+1}^{(i)})$

(b) $R_i''' \leftarrow (R''_i \oplus (H_0^j \,\|\, H_1^i \,\|\cdots\|\, H_7^j))$

$CV_i \leftarrow R'''_i$

The final hash value of the message $M$ will be $L_n$, to remove the length extension attack, multicollision attack and herding attack. Since $CV_i$ is the initial value for the block $M^{(i+1)}$ and $CV_n$ is not known; hence appending any extra block to compute the R-hash would not be possible.

The complete algorithm is given in Algorithm 1 and block diagram of R-hash is shown in Figure 2.

**Process of Implementation:** In order to compute *R-hash*(*M*), first the padding rule is applied and then the padded message is divided into 512-bit blocks. Now each 512-bit block is divided into sixteen 32-bit words and each 32-bit word is read in little endian format. For example, suppose we have to read an ASCII file with data '*abcd*', it will be read as 0x64636261.

**Test Value of HF-hash:** Test values of the three inputs are given below:

$$
\begin{aligned}
R-hash\,(a) &= 7A8FA11D\ 7E1E0B08\ EB55F19D\ 39C60488 \\
&\quad CBE2C601\ 92E87208\ 3E7DBE8A\ 3AAEFE38 \\
R-hash\,(ab) &= 69C08E89\ 728F2705\ DDBC05A8\ 2A687B97 \\
&\quad BE9A6F48\ E75BB73B\ 3FC17AD3\ 653F5E88 \\
R-hash\,(abc) &= AE1771AE\ 2726C492\ CB772115\ 7D8B82A9 \\
&\quad 849FAF09\ 5E0251BC\ 14DBBF42\ AE7AA8F0
\end{aligned}
$$

# 3. ANALYSIS OF R-HASH

In this section we will present the complete analysis of *R-hash*, which includes properties, efficiency, as well as the security analysis of this function.

## 3.1. Efficiency of R-hash Function

The following table gives a comparative study in the efficiency of *R-hash* with *HF-hash* on an Intel Core2Duo PC with P8400 chipset @ 2.26 Ghz processor with 2 GB RAM.

| **File Size** (in MB) | ***HF-hash*** (in sec) | ***R-hash*** (in Sec) |
|---|---|---|
| 1.46 | 35.27 | **12.25** |
| 4.84 | 118.48 | **40.72** |
| 7.79 | 188.67 | **64.80** |
| 14.51 | 351.26 | **120.63** |

Table 1: Comparative in the Efficiency of R-hash

This shows that *R-hash* is almost three times faster than HF-hash.

## 3.2. Preimage Resistance

To find preimage of *R-hash*, one has to solve 32 random quadratic multivariate polynomial equations with 64 variables over *GF(2)* which is an *MQ-problem*. If it is easy to find out preimage of *R-hash*, then *MQ-problem* is no longer an *NP-hard* problem, which is a contradiction. Thus *R-hash* is preimage resistant hash function.

## 3.3. Second Preimage Resistance

We know that collision resistance implies second preimage resistance. Therefore, the proof of collision resistance of *R-hash* gives the second preimage resistance of *R-hash*.

## 3.4. Collision Resistance

Since we have totally changed the design principle of the compression function in *R-hash*, therefore the differential attack applied for SHA-0 and SHA-1 by Chabaud and Joux in [4] and by Wang et al. in [30], [29] to find the collision is not applicable to *R-hash* function. All these attacks use the message expansion relation to find the collision,

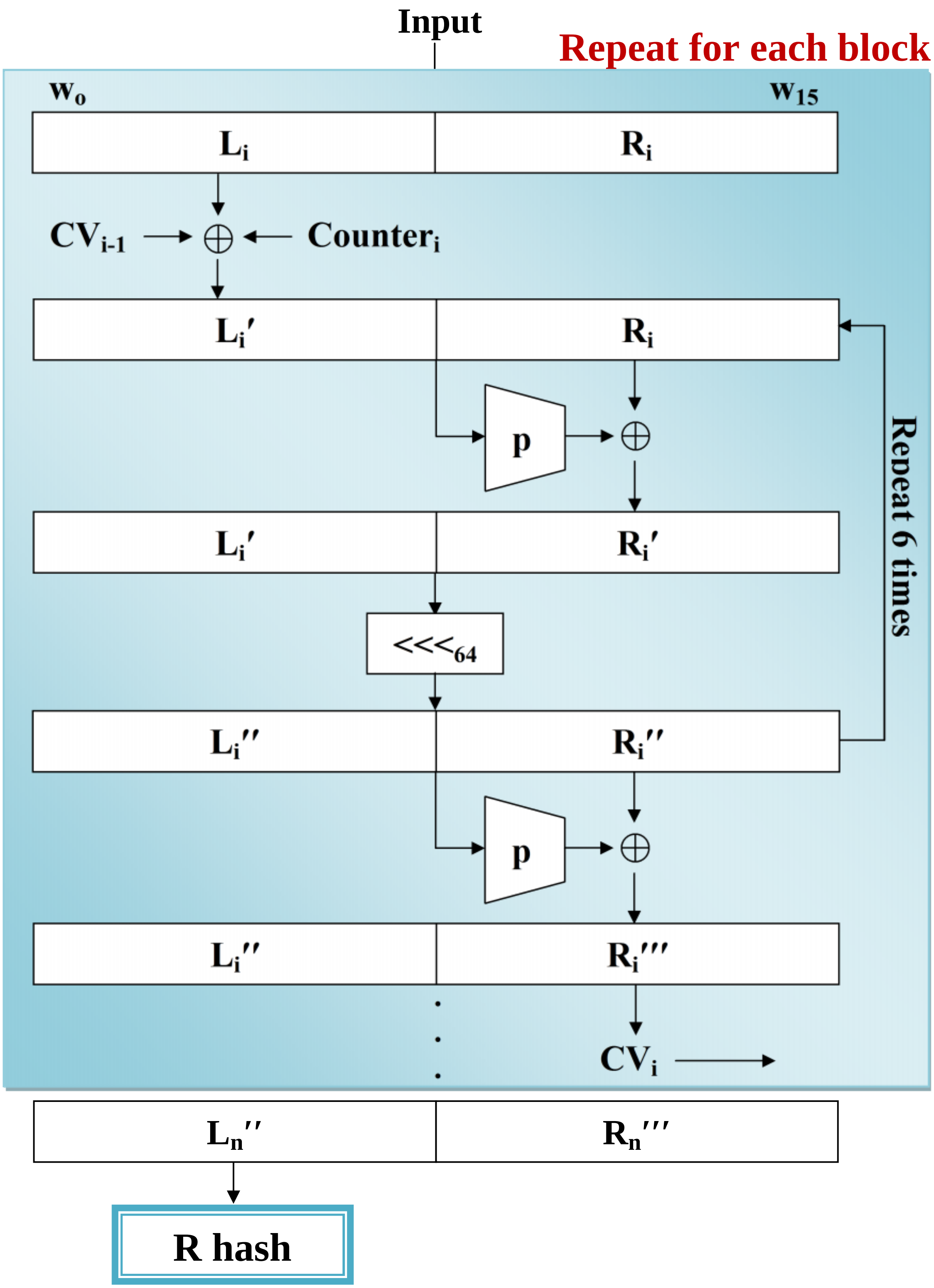


Figure 2: Block Diagram of R-hash

but in our design, we have not applied the message expansion algorithm. Hence, this hash function is collision resistance against the above methods.

Since the design of the compression function of R-hash is different from SHA-2 family, the cross dependence equation described by Sanadhya and Sarkar in [25] cannot be formed in R-hash. Thus this procedure too cannot be applied to our hash function for finding collisions.

Besides these, we have computed *R-hash* for a number of files by changing only one bit to the input. In most of the cases, we found that 17 bits changed after one round computation of *R-hash* and 51 bits changed after two rounds. So it would be difficult to control these bits as the number of rounds increases. Thus, differential attack to find the collision for *R-hash* would be difficult.

## 3.5. Avalanche Effect

We have taken an input file *M* consisting of 512 bits and computed *R-hash*(*M*). By changing the $i^{th}$ bit of *M*, the files $M_i$ have been generated, for $1 \le i \le 512$. Thus Hamming distance of each $M_i$ from *M* is exactly 1 for $1 \le i \le 512$. We then computed *R-hash*( $M_i$ ) for $1 \le i \le 512$ , computed the Hamming distances $d_i$ between *R-hash*(*M*) and *R-hash*( $M_i$ ), for $1 \le i \le 512$ and finally computed the distances between corresponding eight 32-bit words of the hash values. The following table shows the maximum, the minimum, the mode and the mean values of the above distances.

| Changes | $W_1$ | $W_2$ | $W_3$ | $W_4$ | $W_5$ | $W_6$ | $W_7$ | $W_8$ | *HF-hash* | **R-hash** |
|---|---|---|---|---|---|---|---|---|---|---|
| Max | 26 | 24 | 23 | 25 | 26 | 25 | 26 | 25 | 149 | **165** |
| Min | 9 | 6 | 6 | 8 | 7 | 8 | 8 | 7 | 103 | **104** |
| Mode | 15 | 16 | 15 | 16 | 15 | 15 | 17 | 16 | 132 | **131** |
| Mean | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 128 | **128** |

Table 2: Hamming Distances

To satisfy strict avalanche criterion, each $d_i$ should be 128 for $1 \le i \le 512$. But we have found that $d_i's$ were lying between 104 and 165 for the above files and in most of the cases $d_i = 131$. The observed deviation is acceptable so as to resist collision search using differential attack. The following table and figure show the distribution of the 512 files with respect to their differences (distance) in bits.

| Range of Distance | No. of Files | % *HF-hash* | **%** ***R-hash*** |
|---|---|---|---|
| $128 \pm 5$ | 257 | 47.99 | **50.20** |
| $128 \pm 10$ | 399 | 80.80 | **77.93** |
| $128 \pm 15$ | 479 | 93.97 | **93.56** |
| $128 \pm 20$ | 505 | 98.88 | **98.63** |

Table 3: Distribution of the Differences of R-hash by Changing a Single Bit

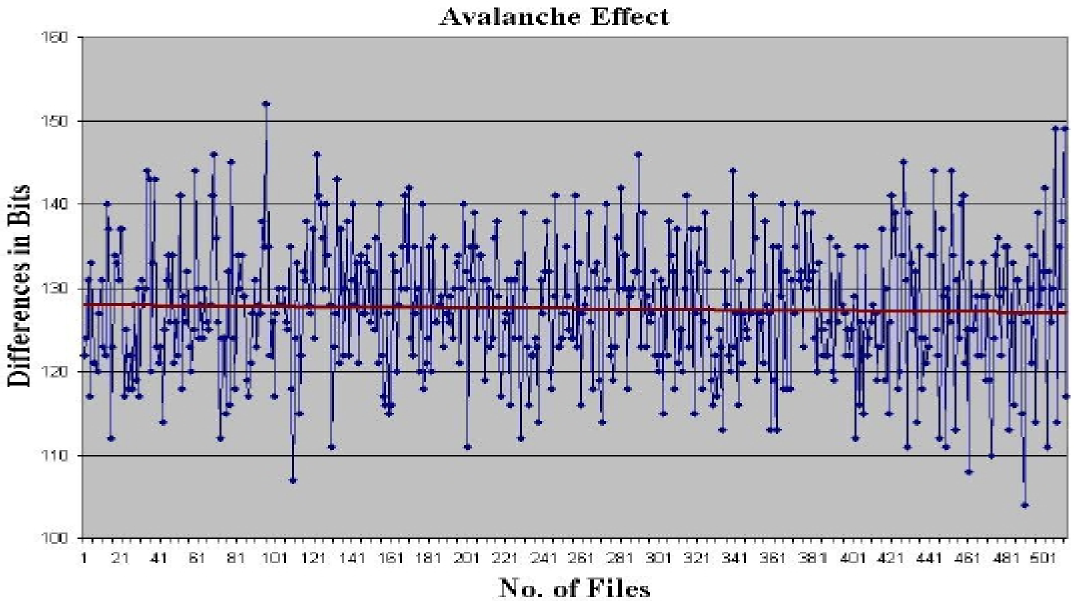


Figure 3: Distribution of the Differences of R-hash by Changing a Single Bit

### 3.6. Randomness Test

To conduct randomness test, we have generated a file consisting of 131328 bits by concatenating all the output of R-hash of the files $M, M_1, M_2, \cdots, M_{512}$. After that, we have divided 131328 bits into 64 blocks of length 2048 bits each, 32 blocks of length 4096 bits each, 16 blocks of length 8192 bits each, 8 blocks of length 16384 bits each, 4 blocks of length 32768 bits each, 2 blocks of length 65536 bits each and 1 block of the complete 131328 bits. Thus we have generated 127 blocks in total and conducted five basic randomness tests in these blocks. The concise result is shown in the following table.

| **Test** | **No. of Blocks** | **Passed** | **Failed** | **%** |
|---|---|---|---|---|
| Frequency | 127 | 127 | 0 | 100 |
| Serial | 127 | 127 | 0 | 100 |
| Poker-4 | 127 | 127 | 0 | 100 |
| Runs | 127 | 124 | 3 | 98.11 |
| Autocorrelation | 127 | 125 | 2 | 98.74 |

Table 4: Result of Randomness Test

### 3.7. The Bit-Variance Test

The bit variance test consists of measuring the impact of change in input message bits on the digest bits. More specifically, given an input message, all the small changes as well as the large changes of this input message bits are taken and the bits in the corresponding digest are evaluated for each such change. Afterwards, for each digest bit the probabilities of taking on the values of 1 and 0 are measured considering all the digests produced by applying input message bit changes. If $P_i(1) = P_i(0) = 1/2$ for all digest bits $i = 1, \cdots, 256$ then, the *R-hash* function has attained maximum performance in terms of the bit variance test [17]. The bit variance test actually measures the uniformity of each bit of the digest. Since it is computationally difficult to consider all input message bit changes, we have evaluated the results for only up to 513 files, viz. $M, M_1, M_2, \cdots, M_{512}$, which we have generated for conducting avalanche effect, and found the following results:

Number of digests = 513
Mean frequency of 1s (expected) = 256.50
Mean frequency of 1s (calculated) = 256.35

## 4. CONCLUSIONS

In this paper a dedicated hash function *R-hash* has been presented whose security is based on the *MQ-problem* over finite field. This hash function has the following advantages over *HF-hash*: it is much more efficient, it is secure against multicollision attack, fixed point attack, length extension attack and herding attack. Moreover, analysis of this hash function viz. avalanche effect, bit-variance test, randomness test as well as security proof are also described here. From these experimental results, it is clear that this hash function can also be used as a pseudo-random number generator because of the good randomness property of its output besides the other applications of cryptographic hash functions.

## ACKNOWLEDGEMENTS

The first two authors are thankful to Mrs Pratibha Yadav, SAG, for her support to carry out this work and are also very thankful to Dr. P. K. Saxena, Director SAG, for his encouragement to pursue this work. The authors are thankful to Mr. Bharat Lal Jangid, SAG, for carrying out the randomness tests of *R-hash*. The authors are also thankful to Dr. P. R. Mishra, SAG, for his critical suggestions and comments on the first draft of this paper.

## REFERENCES

[1] J. Aumasson, L. Henzen, W. Meier, R. Phan, "SHA-3 Proposal BLAKE (2009)", submission to NIST's SHA-3 competition.

[2] J. Black, M. Cochran & T. Highland, "A Study of the MD5 Attacks: Insights and Improvements", in Proceedings of Fast Software Encryption – FSE'06, LNCS 4047, Springer, 2006.

[3] G. Bertoni, J. Daemen, M. Peeters, G. Assche, "The KECCAK sponge function family (2009)", submission to NISTs SHA-3 competition.

[4] F. Chabaud & A. Joux, "Differential Collisions in SHA-0", advances in Cryptology - CRYPTO'98, LNCS 1462, Springer-Verlag, 1998.

[5] C. Canniere, F. Mendel & C. Rechberger, "Collisions for 70-step SHA-1: On the Full Cost of Collision Search", in Proceedings of SAC'07, LNCS 4876, Springer, 2007.

[6] C. Canniere & C. Rechberger, "Preimages for Reduced SHA-0 and SHA-1, advances in Cryptology - CRYPTO'08, LNCS 5157, Springer-Verlag, 2008.

[7] D. Dey, P. R. Mishra & I. Sengupta, “HF-hash: Hash Functions Using Restricted HFE Challenge-1, International Journal of Advanced Science and Technology Vol. 37, Dec 2011. Available online at http://www.sersc.org/journals/IJAST/Vol37/11.pdf

[8] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, J. Walker, “The Skein Hash Function Family (2009)”, submission to NIST’s SHA-3 competition.

[9] P. Gauravaram, L. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schlaffer, S. Thomsen, “Grøstl a SHA-3 Candidate (2009)”, submission to NIST’s SHA-3 competition.

[10] H. Gilbert & H. Handschuh, “Security Analysis of SHA-256 and Sisters”, in Proceedings of SAC’03, LNCS 3006, Springer, 2003.

[11] M. Garey & D. Johnson, “Computers and Intractability - A Guide to the Theory of NP-completeness”, Freeman and Co., 1979.

[12] S. Indesteege, F. Mendel, B. Preneel & C. Rechberger, “Collisions and other Non-Random Properties for Step-Reduced SHA-256”. Cryptology eprint Archive. Available online at http://eprint.iacr.org/2008/131.pdf

[13] A. Joux, “Multicollisions in Iterated Hash Functions - Application to Cascaded Constructions”, advances in Cryptology CRYPTO’04, LNCS 3152, Springer, 2004.

[14] J. Kelsey & T. Kohno, “Herding Hash Functions and the Nostradamus Attack”, advances in Cryptology EUROCRYPT’06, LNCS 4004, Springer-Verlag, 2006.

[15] V. Klima, “Tunnels in Hash Functions: MD5 Collisions within a Minute”, Cryptology ePrint Archive. Available online at http://eprint.iacr.org/2006/105.pdf

[16] J. Kelsey & B. Schneier, “Second Preimages on n-bit Hash Functions for Much Less Than 2n Work”, advances in Cryptology EUROCRYPT’05, LNCS 3494, Springer-Verlag, 2005.

[17] D.Karras & V. Zorkadis, “A Novel Suite of Tests for Evaluating One-Way Hash Functions for Electronic Commerce Applications”, IEEE, 2000.

[18] F. Mendel, N. Pramstaller, C. Rechberger & V. Rijmen, “Analysis of Step-Reduced SHA-256”, in Proceedings of FSE’06, LNCS 4047, Springer, 2006.

[19] F. Mendel, N. Pramstaller, C. Rechberger & V. Rijmen, “Analysis of Step-Reduced SHA-256”, Cryptology eprint Archive. Available online at http://eprint.iacr.org/2008/130.pdf

[20] I. Nikolic & A. Biryukov, “Collisions for Step-Reduced SHA-256” in Proceedings of FSE’08, Springer, 2008.

[21] National Institute of Technology, “Secure Hash Standard, FIPS Publication-180-2”, 2002. Available online at http://www.itl.nist.gov/fipspubs/

[22] National Institute of Standards and Technology, “Cryptographic Hash Project”, Available online at http://csrc.nist.gov/groups/ST/hash/index.html

[23] N. Pramstaller, C. Rechberger & V. Rijmen, “Exploiting Coding Theory for Collision Attacks on SHA-1”, in Cryptography and Coding, 10th IMA International Conference Proceedings, LNCS 3796, Springer, 2005.

[24] M. Sugita, M. Kawazoe & H. Imai, “Groebner Basis Based Cryptanalysis of SHA-1”, Cryptology ePrint Archive. Available online at http://eprint.iacr.org/2006/098.pdf

[25] S. Sanadhya & P. Sarkar, “Non-Linear Reduced Round Attacks Against SHA-2 Hash family”, Information Security and Privacy - ACISP 2008, LNCS, Springer, 2008.

[26] S. Sanadhya & P. Sarkar, “Attacking Step Reduced SHA-2 Family in a Unified Framework”, Cryptology eprint Archive. Available online at http://eprint.iacr.org/2008/271.pdf

[27] H. Wu, “The Hash Function JH (2009)”, submission to NIST’s SHA-3 competition.

[28] X. Wang & H. Yu, “How to Break MD5 and Other Hash Functions”, advances in Cryptology – EUROCRYPT’05, LNCS 3494, Springer, 2005.

[29] X. Wang, Y. Yin & H. Yu, “Finding Collisions in the Full SHA-1”, advances in Cryptology CRYPTO’05, LNCS 3621, pages 17 - 36, Springer, 2005.

[30] X. Wang, H. Yu & Y. Yin, “Efficient Collision Search Attacks on SHA-0”, in Advances in Cryptology - CRYPTO’05, LNCS 3621, Springer, 2005.

[31] T. Xie, D. Feng & F. Liu, “A New Collision Differential for MD5 with its Full Differential Path”, Cryptology eprint Archive. Available online at http://eprint.iacr.org/2008/230.pdf