International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.2, Mar-Apr 2012 pp-462-470 ISSN: 2249-6645

GB-hash: Hash Functions Using Groebner Basis

Dhananjoy Dey[1], Prasanna Raghaw Mishra[1], Indranath Sengupta[2]
[1]SAG, DRDO, Metcalfe House, Delhi - 110 054, INDIA.
[2]Department of Mathematics, Jadavpur University, Kolkata, WB - 700 032, INDIA.

Abstract

In this paper we present an improved version of HF-hash [DMS] viz., GB-hash: Hash Functions Using Groebner Basis. In case of HF-hash, the compression function consists of 32 polynomials with 64 variables, which were taken from the first 32 polynomials of hidden field equations challenge-1 by forcing last 16 variables as 0. In GB-hash we have designed the compression function in such way that these 32 polynomials with 64 variables form a minimal Groebner basis of the ideal generated by them with respect to graded lexicographical (grlex) ordering as well as with respect to graded reverse lexicographical (grevlex) ordering. In this paper we will prove that GB-hash is more secure than HF-hash as well as it is little bit faster than HF-hash.

Keywords: *Dedicated hash functions, differential attack, Groebner basis, preimage attack.*

1 Introduction

After recent cryptanalytic attack on MD5 [WY] and SHA-1 [WYY], the security of their successor, SHA-2 family [NIST], against all kinds of cryptanalytic attacks has become an important issue. Although many attacks [GH], [MPRR], [MPRR1], [NB], [IMPR], [SS] on the reduced round of SHA-256 are published between 2003 to 2008, but no result gives any practical threat to the security of SHA-256 till now. In the mean time NIST announced SHA-3 competition in 2007 and the final SHA-3 candidate will be declared by the second quarter of this year. All hash functions submitted for the SHA-3 competition [NIST1] are divided on the following broad category: *balanced Feistel network, unbalanced Feistel network, wide pipe design, key schedule, MDS matrix, output transformation, S-box* and *feedback register*. But it is still an important issue to analyse the hash function based on the design principle of MD4 family.

We have already designed a cryptographic hash function viz. HF-hash [DMS] in which we have designed the compression function consisting of 32 polynomials with 64 variables which were taken from the first 32 polynomials of hidden field equations challenge-1 by forcing last 16 variables as 0. The leading monomials of 32 polynomials with respect to graded lexicographical ordering used in HF-hash are the following:

$$x_1x_2, x_1x_3, x_1x_2, x_1x_3, x_1x_3, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_3, x_1x_6, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_2, x_1x_4, x_1x_2,$$
$$x_1x_2, x_1x_3, x_1x_3, x_1x_5, ,x_1x_2, x_1x_7, x_1x_2, x_1x_2, x_1x_2, x_1x_3, x_1x_2, x_1x_2, x_1x_2, x_1x_3.$$

Therefore, there are only six different leading monomials viz. $x_1x_2, x_1x_3, x_1x_4, x_1x_5, x_1x_6$ & x_1x_7.

To improve the design of the compression function of HF-hash function, we have designed a new hash function viz. GB-hash. The leading monomials of the compression function of 32 polynomials with respect to *grlex* ordering as well as with respect to *grevlex* ordering used in GB-hash are given below:

$$x_1x_2, x_3x_4, x_5x_6, x_7x_8, x_9x_{10}, x_{11}x_{12}, x_{13}x_{14}, x_{15}x_{16}, x_{17}x_{18}, x_{19}x_{20}, x_{21}x_{22}, x_{23}x_{24}, x_{25}x_{26}, x_{27}x_{28}, x_{29}x_{30},$$
$$x_{31}x_{32}, x_{33}x_{34}, x_{35}x_{36}, x_{37}x_{38}, x_{39}x_{40}, x_{41}x_{42}, x_{43}x_{44}, x_{45}x_{46}, x_{47}x_{48}, x_{49}x_{50}, x_{51}x_{52}, x_{53}x_{54}, x_{55}x_{56}, x_{57}x_{58},$$
$$x_{59}x_{60}, x_{61}x_{62}, x_{63}x_{63}.$$

These 32 polynomials form a minimal Groebner basis for the ideal they generate with respect to grlex ordering as well as with respect to grevlex ordering. But if any one wants to solve the system of equations formed by these polynomials, (s)he cannot reduce the number of polynomials <8 with respect to any monomial ordering with the assumption that they form a Groebner basis. So the number of equations cannot be reduced to less than 8.

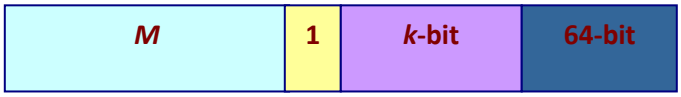
In this paper we prove that GB-hash is more secure than HF-hash with respect to the preimage resistance as well as the collision search attack in the subsequent sections.

2 GB-hash

GB-hash function can take arbitrary length $(<2^{64})$ of input and gives 256 bits output. We have designed GB-hash by changing the compression function. The compression function is designed by taking a set of 32 polynomials with 64 variables over *GF(2)* which form a minimal Groebner basis of an ideal they generate with respect to *grlex* or *grevlex* ordering; where as the compression function of HF-hash consists of 32 polynomials with 64 variables which were taken from the first 32 polynomials of hidden field equations challenge-1 by forcing last 16 variables as 0. For computation of GB-hash, we have taken the padding and parsing procedure, initial value and the 64 constants are the same as HF-hash. For completeness of the algorithm, the computation of hash value of a message *M* of length *l* bits is given below:

Padding: First we append 1 to the end of the message *M*. Let *k* be the number of zeros added for padding. The 64-bit representation of *l* is appended to the end of *k* zeros. The padded message M is shown in the following figure. Now *k* will be the smallest positive integer satisfying the following condition:

$$l+1+k+64\equiv 0 \bmod 448$$
$$i.e.,\quad k+l\equiv 383 \bmod 448$$



Padded Message

Parsing: Let l' be the length of the padded message. Divide the padded message into $n(=l'/448)$ 448-bit block i.e. fourteen 32-bit words. Let $M^{(i)}$ denote the i^{th} block of the padded message, where $1\le i\le n$ and each word of i^{th} block is denoted by $M_j^{(i)}\ for 1\le j\le 14.$

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

$$h_0^{(0)}=243F6A88, h_1^{(0)}=85A308D3, h_2^{(0)}=13198A2E, h_3^{(0)}=03707344,$$
$$h_4^{(0)}=A4093822, h_5^{(0)}=299F31D0, h_6^{(0)}=082EFA98, h_7^{(0)}=EC4E6C89.$$

Hash Computation: For each 448-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**

$$H_j = h_j^{(i-1)}\ \ for\ 0\le j\le 7.$$

2. **Expansion**

i. $W_0 \leftarrow H_0$

ii. $W_j \leftarrow M_j^{(i)},\ for\ 1\le j\le 14$

iii. $W_{15} \leftarrow H_7$

iv. $W_j \leftarrow rotl_3(W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-1}),\ for\ 16 \le j \le 63,$ where $rotl_k$ denotes the left rotation by *k*.

This is the expansion of the message blocks without padding. In the last block we apply padding rule. If $(l+1) > 384$ bits, then we have two extra blocks in the padded message. Otherwise we have one extra block in the padded message. In both the cases, we apply the following expansion rule for the last block so that the length of the message appears in the end of the padded message.

i. $W_0 \leftarrow H_0$

ii. $W_1 \leftarrow H_7$

iii. $W_j \leftarrow M_j^{(i)},\ for\ 2 \le j \le 15$

iv. $W_j \leftarrow rotl_3(W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-1}),\ for\ 16 \le j \le 63$

3. Iteration for *j* = 0 to 63

i. $T_1 \leftarrow H_1 + H_2 + p(H_3 \| H_0) + K_j$ [1]

ii. $T_2 \leftarrow H_4 + H_5 + p(H_7 \| H_6) + W_j$

iii. $H_7 \leftarrow H_6$

iv. $H_6 \leftarrow H_5$

v. $H_5 \leftarrow H_4$

vi. $H_4 \leftarrow rotl_5(H_3 + T_2)$

vii. $H_3 \leftarrow H_2$

viii. $H_2 \leftarrow H_1$

ix. $H_1 \leftarrow H_0$

x. $H_0 \leftarrow T_1 + T_2$, where T_1 and T_2 are two temporary variables and $p : Z_{2^{64}} \rightarrow Z_{2^{32}}$ 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 bits of x in decreasing order of their significance. The list of polynomials $p_i(x_1,\cdots,x_{64})$ for $1 \le i \le 32$ is given in https://docs.google.com/file/d/0ByA1ZE-dqRLQaDFvRWZGZHVUNFNpS1otWjdTVk8tZw/edit

The 64 constants K_j are taken from the fractional part of e and are given below:

[1] The operation || denotes the concatenation and + denotes the addition *mod* 2^{32}.

K_0 = AC211BEC	K_1 = 5FEFE110	K_2 = 112276F8	K_3 = 8AE122A4
K_4 = 18B3488B	K_5 = 00921A36	K_6 = 40C045F8	K_7 = C8C0A3DA
K_8 = C4ABF676	K_9 = 6A68C750	K_{10} = A37AFE0F	K_{11} = 732806F3
K_{12} = 25722CB7	K_{13} = 3FF43825	K_{14} = ACDF96D7	K_{15} = 9B53BCD3
K_{16} = E34950DE	K_{17} = D9780CCB	K_{18} = 8B5F9BB7	K_{19} = 3D1182ED
K_{20} = 1921B44A	K_{21} = 7003F30D	K_{22} = 42657E31	K_{23} = 231E7B55
K_{24} = 91E3A28E	K_{25} = 95CD4AB0	K_{26} = 0A0AC2E3	K_{27} = FCDEBE5E
K_{28} = FCF1E321	K_{29} = 1D136560	K_{30} = 2974BF63	K_{31} = 70963992
K_{32} = 4F5B5107	K_{33} = 0072C0C1	K_{34} = C99F3C1D	K_{35} = C56598D9
K_{36} = 77A1D027	K_{37} = 36675FB6	K_{38} = A40C34E8	K_{39} = 46764EAD
K_{40} = F8823861	K_{41} = 19F66E64	K_{42} = 87E10299	K_{43} = 4311C8C2
K_{44} = 07C102B9	K_{45} = 9F4EC8CE	K_{46} = 29D81EBA	K_{47} = 992744F9
K_{48} = 4CDA6790	K_{49} = 13DA5357	K_{50} = BA6D7772	K_{51} = 80673F08
K_{52} = B049EE4C	K_{53} = 839F8647	K_{54} = 736F658B	K_{55} = EBE90F9B
K_{56} = FA6DC4D1	K_{57} = E951630E	K_{58} = AFC453E4	K_{59} = 159B7483
K_{60} = 45EABF9D	K_{61} = 4292A60E	K_{62} = 17AA0ABD	K_{63} = 94E81C30

4. Intermediate Hash Value

The i^{th} intermediate hash value

$$h^{(i)} = h_0^{(i)} \| h_1^{(i)} \| h_2^{(i)} \| h_3^{(i)} \| h_4^{(i)} \| h_5^{(i)} \| h_6^{(i)} \| h_7^{(i)},$$

where $h_j^{(i)} = H_j \quad for\ 0 \le j \le 7.$ This $h^{(i)}$ will be the initial value for the message block $M^{(i+1)}$.

The final hash value of the message M will be

$$h_0^{(n)} \| h_1^{(n)} \| h_2^{(n)} \| h_3^{(n)} \| h_4^{(n)} \| h_5^{(n)} \| h_6^{(n)} \| h_7^{(n)},$$

where $h_i^{(n)} = H_i \quad for\ \ 0 \le i \le 7.$

Process of Implementation: In order to compute GB-hash(M), first the padding rule is applied and then the padded message is divided into 448-bit blocks. Now each 448-bit block is divided into fourteen 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 GB-hash

Test values of the three inputs are given below:

GB-hash(a) = *f1887394 23fab8a8 0512448e 43d6755e da90c8d0 c38c38d0 db7ab991 4645e099*

GB-hash(ab) = *b302d927 033fd17e 1e2ff903 839e4b35 1feb55e2 fadd9f8b dca0adbf 1c719df9*

GB-hash(abc) = *59d647e2 765243b3 49d01559 8392ba30 476c5c65 dfacc415 a7a9de8c 794e8bb9*

3. Analysis of GB-hash

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

3.1 Properties of GB-hash

This subsection describes the properties of GB-hash required for cryptographic applications.

i. **Easy to compute:** For any given value *x* it is easy to compute GB-hash(*x*) and the efficiency of this hash function is given in section 3.2.

ii. **One-wayness:** Suppose one knows the GB-hash(*x*) for an input *x*. Now to find the value of *x*, (s)he has to solve the system of polynomial equations consisting of 32 polynomials with 64 variables given in the site for each round operation. Since this system of equations is underdefined, the XL [CKPS] method or any variant of XL [YC] cannot be applied to solve this system.

We will prove that this system of equations cannot be solve in polynomial time using the method described for solving underdefined system of equation in [KPG].

Proposition 3.1. Let *G* denote the set of polynomials $\{p_1, p_2, \cdots, p_{32}\}$, where $p_i's$ are defined above and *G* generates an ideal *I*. Then

(i) *G* is a Groebner basis for *I* with respect to the monomial order grlex as well as with respect to grevlex.

(ii) If G' is a non-empty subset of G and if G' is a Groebner basis for *I* with respect to some monomial order, then $\#G' \geq 8.$

Proof 3.1.

(i) The leading term (lt) of $p_1, p_2, \cdots, p_{32}$ *in G with respect to the monomial order grlex as well as with respect to grevlex are* x_1x_2, x_3x_4, x_5x_6, x_7x_8, x_9x_{10}, $x_{11}x_{12}$, $x_{13}x_{14}$, $x_{15}x_{16}$, $x_{17}x_{18}$, $x_{19}x_{20}$, $x_{21}x_{22}$, $x_{23}x_{24}$, $x_{25}x_{26}$, $x_{27}x_{28}$, $x_{29}x_{30}$, $x_{31}x_{32}$, $x_{33}x_{34}$, $x_{35}x_{36}$, $x_{37}x_{38}$, $x_{39}x_{40}$, $x_{41}x_{42}$, $x_{43}x_{44}$, $x_{45}x_{46}$, $x_{47}x_{48}$, $x_{49}x_{50}$, $x_{51}x_{52}$, $x_{53}x_{54}$, $x_{55}x_{56}$, $x_{57}x_{58}$, $x_{59}x_{60}$, $x_{61}x_{62}$, $x_{63}x_{64}$ *respectively.*

Since $lt(p_i)\times lt(p_j)=lcm(lt(p_i),lt(p_j))$ *i.e.* $lt(p_i)$ *and* $lt(p_j)$ *are relatively prime then S-polynomial* $S(p_i, p_j)$ *reduces to zero for* $i \neq j \,\&\, i, j \in \{1,2,\cdots,32\}.$

This shows that the set of polynomials $\{p_1, p_2, \cdots, p_{32}\}$ *forms a Groebner basis with respect to grlex or grevlex ordering. Furthermore, no leading monomial of* p_i *divides the leading monomials of* p_j *for* $i \neq j.$ *Thus* $\{p_1, p_2, \cdots, p_{32}\}$ *forms a minimal Groebner basis with respect to grlex or grevlex ordering.*

(ii) Let $G_1=\{p_1, p_2, \cdots, p_8\}, G_2=\{p_9, p_{10}, \cdots, p_{16}\}, G_3=\{p_{17}, p_{18}, \cdots, p_{24}\} \,\&\, G_4=\{p_{25}, p_{26}, \cdots, p_{32}\}$ *and* $V_1=\{x_1, x_2, \cdots, x_{16}\}, V_2=\{x_{17}, x_{18}, \cdots, x_{32}\}, V_3=\{x_{33}, x_{34}, \cdots, x_{48}\} \,\&\, V_4=\{x_{49}, x_{50}, \cdots, x_{64}\}.$ *Therefore,* $G=G_1 \cup G_2 \cup G_3 \cup G_4$ *and* $G_i \cap G_j = \phi$ *for* $i \neq j.$ *We have chosen the sets* $G_i's$ *in such a way that the linear terms of* G_i *consist of the variables from the set* V_i *for* $1 \leq i \leq j.$ *And the non-linear terms of the form* $x_j x_k$ *are chosen in such a way that* $x_j \in V_i$ *and* $x_k \in \{x_{j+1}, x_{j+2}, \cdots, x_{64}\}$ *for* $1 \leq j \leq 16$ *if* $i=1,$ *for* $17 \leq j \leq 32$ *if* $i=2,$ *for* $33 \leq j \leq 48$ *if* $i=3$ *and for* $49 \leq j \leq 63$ *if* $i=4.$

The sets G_i *are selected in a manner that no monomial will appear more than 7 times. Now suppose that the monomial* $x_a x_b$ *appears 7 times in some* G_i*. Then if we consider the variable order* $x_a > x_b > \cdots,$ *there exists 7 polynomials whose leading terms will be* $x_a x_b$*. If we compute Groebner basis of these 7 polynomials, there will be only one polynomial in the Groebner basis. Since each* G_i *contains 8 polynomials, so one has to take at least 2 elements from each* G_i *to form a Groebner basis with respect to any monomial order.*

Thus if G' is a non-empty subset of G and if G' is a Groebner basis for the ideal I with respect to some monomial order, then $\#G'$ is at least 8.

Since the set of polynomials used in designing the compression function of GB-hash can not be reduced to a set consisting of < 8 polynomials such that they form a Groebner basis with respect to any monomial ordering, so by [KPG], we can say that the system of polynomials equations taken from the compression function of GB-hash cannot be solved in polynomial time.

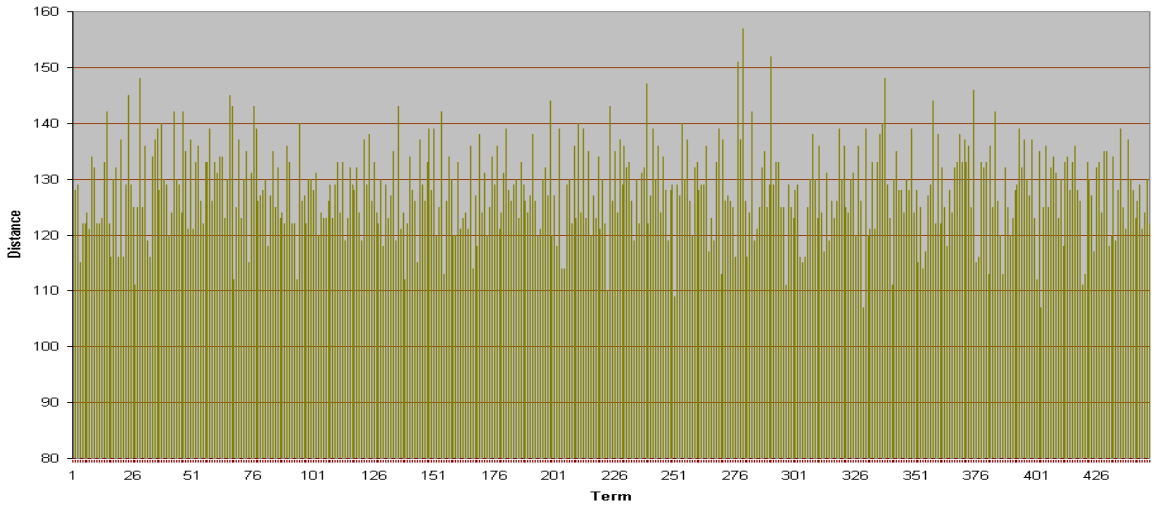
Now, in order to solve this system of equations using the [2]Algorithm A given by Courtois et. al. in [CGMT], at least 2^{27} operations are required to solve for one round of GB-hash. Since GB-hash has 64 rounds one has to compute $2^{27\times 64}$ operations to get back the value of *x*, for a given GB-hash(*x*). This is far beyond the today's computation power.

Thus, for any given GB-hash(*x*), it is difficult to find the input *x*.

iii. **Randomness:** We have taken an input file M consisting of 448 bits and computed GB-hash(M). By changing the i^{th} bit of M, the files M_i have been generated, for $1 \le i \le 448$. We then computed GB-hash(M_i) of all the 448 files M_i, computed the Hamming distances d_i between GBhash(*M*) and GB-hash(M_i), for $1 \le i \le 448$ 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 of the above distances.

Changes	W_1	W_2	W_3	W_4	W_5	W_6	W_7	W_8	**GB-hash**	HF-hash
Max	24	23	25	24	24	23	24	24	**157**	149
Min	8	7	8	7	6	7	8	8	**107**	103
Mode	16	16	15	15	15	16	15	16	**129**	132
Mean	16	16	16	16	16	16	16	16	**128**	128

Ideally d_i should be 128 for $1 \le i \le 448$. But we have found that $d_i's$ were lying between 107 and 157 for the above files. The following table and the figure show the distribution of the 448 files with respect to their distances.



[2] which is the best algorithm for solving our system of equations among Algorithms A, B & C

Range of Distance	**No. of Files**	**Percentage GB-hash**	Percentage HF-hash
128 ± 5	241	**53.79**	47.99
128 ± 10	366	**81.70**	80.80
128 ± 15	425	**94.87**	93.97
128 ± 20	444	**98.88**	98.88

iv. The Bit-Variance Test: The bit variance test consists of measuring the impact of changing 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 occur 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 $i = 1, 2, \cdots, n,$ where *n* is the digest length, then, the one-way hash function under consideration has attained maximum performance in terms of the bit variance test [KZ]. Therefore, 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 449 files and found the following results:

Number of digests = 449
Mean frequency of 1s (expected) = 224.50
Mean frequency of 1s (calculated) = 223.72

The above analysis shows that GB-hash exhibits a reasonably good avalanche effect. Thus it can be used for cryptographic applications.

3.2 Efficiency of GB-hash

The following table gives a comparative study in the efficiency of GB-hash and HF-hash in HP Pentium - D with 3 GHz processor and 512 MB RAM.

File Size (in MB)	**GB-hash (in Sec)**	**HF-hash (in Sec)**
1.4	**18.76**	20.02
4.84	**65.45**	67.72
7.48	**105.28**	109.73
12.94	**174.87**	181.01

The efficiency of GB-hash can be improved by choosing the reduced Groebner basis instead of a minimal Groebner basis.

3.3 Security Analysis

In this paper we have applied a new method for expanding a 512-bit message block into 2048-bit block. For this purpose we have to change the padding rule and the procedure of parsing a padded message. In case of MD-5, SHA-1 & SHA-256, the padded message is divided into 512-bit blocks whereas in case of GB-hash, the padded message is divided into 448-bit blocks. Then two 32-bit words are added to construct a 512-bit block as the input for each iteration, where these two words depend on the previous internal hash updates or chaining variables. So, in each iteration, the 512-bit blocks are not independent from the previous message blocks as in the case of MD-5, SHA-1 or SHA-256. Message expansion algorithm of GB-hash is dependent on

the first and last word of the previous hash. Now if small change is occurred in the inputs, the intermediate hash values will be different. Thus we will get the differences in first and last words of intermediate hash values. These differences along with the rotation in the message expansion formula make impossible to find corrective pattern described in [CJ]. Thus, differential attack by Chabaud and Joux is not applicable to our hash function because one does not have any control over two 32-bit words coming from the previous internal hash updates.

Moreover, a 1-bit difference in any one of 14 initial 32-bit words propagates itself to at least 162 bits of the expanded message since we have taken the 64 round operations. Less than 70 bit difference in expanded message and input message is obtained by changing 1-bit input when 32 or 48 round operations are performed. That is why we have taken 64 round operations for GB-hash function. This makes it impossible to find corrective patterns used by Chabaud and Joux in [CJ], due to the reason that differences propagate to other positions.

The idea of Wang et. al. for finding collision in SHA-0 [WYY1] and SHA-1 [WYY] is to find out the disturbance vectors with low Hamming weight first and then to construct a differential path. To construct a valid differential path, it is important to control the difference propagation in each chaining variable. After identifying the wanted and unwanted differences one can apply the Boolean functions (mainly IF) and the carry effect to cancel out these differences. In particular, when an input difference is 1, the output difference can be 1, -1 or 0. Hence, the function can preserve, flip or absorb an input difference. This gives a good flexibility to construct a differential path. The key of these attacks was the Boolean functions used in compression function which in combination with carry effect facilitate the differential attack.

We have replaced the Boolean functions with 32 polynomials having 64 variables, which form a Groebner basis for the ideal they generate. Now if we change 1 bit in the inputs of GB-hash, the outputs will be the same after one round of operation of the compression function. Because, this input difference will not effect since in our case $W_0 = H_0$. But this input difference will appear in W_1. Thus, the output differences will be found after two rounds of computing compression function. We have computed the difference propagation of chaining variables for several files having 1 bit input difference and the result is given in the following table.

Round	HF-hash		GB-hash	
	Minimum	Maximum	Minimum	Maximum
2	35	53	34	54
3	63	134	63	132
4	88	144	89	143
5	104	145	99	147

This shows that it is impossible to control the difference propagation of chaining variable after round two as in the case of GB-hash. Therefore, these attacks also are not applicable to GB-hash hash function. Although the cross dependence equation described by Sanadhya and Sarkar in [SS] can be formed in case of GB-hash, the procedure of message expansion as well as the compression function of GB-hash being different from SHA-2 family, this procedure for finding collision cannot be applied in our hash function. Thus, this hash function is also collision resistance against the method described by Sanadhya and Sarkar.

Thus the compression function of GB-hash is collision-resistant against existing attacks. Since *IV* of GB-hash is fixed and the padding procedure of GB-hash includes the length of the message, therefore by Merkle-Damgard theorem [Dam] [Mer] we can say that GB-hash is collision-resistant against existing attacks.

4 Conclusion

In this paper a dedicated hash function GB-hash has been presented. A system of multivariate polynomials which form a minimal Groebner basis with respect to grlex or grevlex ordering is applied for designing the compression function of our proposed hash function. Analysis of this hash function viz. randomness as well as security proof are also described here. GB-hash differs from the MD family and the SHA family mainly in the procedure of message expansion and the compression function. The advantages of our proposed hash function over the most commonly used hash functions, are that the differential attack applied by Chabaud and Joux in SHA-0 as well as collision search for SHA-1 by Wang et. al. and collision search method applied by Sarkar et. al. for SHA-2 family are not applicable. Further work is going on regarding the improvement of the efficiency as well as the security of GB-hash.

References

[CGMT] N. Courtois, L. Goubin, W. Meier & J. Tacier, "*Solving Under-defined Systems of Multivariate Quadratic Equation*", PKC 002, LNCS 2274, pages 211 - 227, Springer-Verlag, 2002.

[CJ] F. Chabaud & A. Joux, "*Differential Collisions in SHA-0*", Advances in Cryptology - CRYPTO 098, LNCS 1462, pages 56 - 71, Springer-Verlag, 1998.

[CKPS] N. Courtois, A. Klimov, J. Patarin & A. Shamir, "*Efficient Algorithms for Solving Over-defined Systems of Multivariate Polynomial Equations*", EUROCRYPT 2000, LNCS 1807, pages 392 - 407, Springer-Verlag, 2000.

[Dam] I. B. Damgard, "*A Design Principle for Hash Functions*", in Advances in Cryptology Crypto 089 pages 416 - 427, Springer-Verlag, 1990.

[DMS] D. Dey, P. R. Mishra & I. Sengupta, "*HF-hash: Hash Functions Using Restricted HFE Challenge-1*", in International Journal of Advanced Science and Technology, Vol. 37, pp 129-140, 2011. Available online at http://www.sersc.org/journals/IJAST/Vol37/11.pdf

[GH] H. Gilbert & H. Handschuh, "*Security Analysis of SHA-256 and Sisters*", Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Revised Papers, LNCS 3006, Springer, 2003.

[IMPR] 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

[KPG] A. Kipnis, J. Patarin & L. Goubin, "*Unbalanced Oil and Vinegar Signature Schemes*", Advances in Cryptology EUROCRYPT99, LNCS, 1592, pages 206 - 222 Springer Verlag, 1999.

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

[Mer] R. C. Merkle, "*One-way hash functions and DES*", in Advances in Cryptology Crypto 089, pages 428 - 446, Springer-Verlag, 1990.

[MPRR] F. Mendel, N. Pramstaller, C. Rechberger & V. Rijmen, "*Analysis of Step-Reduced SHA-256*", Fast Software Encryption, FSE 2006, LNCS 4047, pages 126 - 143, Springer, 2006.

[MPRR1] 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

[NB] I. Nikolic & A. Biryukov, "*Collisions for Step-Reduced SHA-256*", Fast Software Encryption, FSE 2008, pages 1 - 16. Springer, 2008.

[NIST] National Institute of Technology, Secure Hash Standard, "*FIPS Publication-180-2*", 2002. Available online at http://www.itl.nist.gov/_pspubs/.

[NIST1] National Institute of Standards and Technology, "*Cryptographic Hash Project*". Available online at http://csrc.nist.gov/groups/ST/hash/index.html

[SS] S. Sanadhya & P. Sarkar, "*Non-Linear Reduced Round Attacks Against SHA-2 Hash Family*", Information Security and Privacy -ACISP 2008, LNCS, Springer, 2008.

[SS1] S. Sanadhya & P. Sarkar, "*Attacking Step Reduced SHA-2 Family in a Unified Framework*", Available online at http://eprint.iacr.org/2008/271.pdf

[WY] X. Wang & H. Yu, "*How to Break MD5 and Other Hash Functions*", Advances in Cryptology - EUROCRYPT 2005, LNCS 3494, pages 19 - 35, Springer, 2005.

[WYY] X. Wang, Y. Yin & H. Yu, "*Finding Collisions in the Full SHA-1*", in Advances in Cryptology CRYPTO 005, LNCS 3621, pages 17 - 36, Springer, 2005.

[WYY1] X. Wang, H. Yu & Y. Yin, "*Efficient Collision Search Attacks on SHA-0*", in Advances in Cryptology CRYPTO 005, LNCS 3621, pages 1 - 16, Springer, 2005.

[YC] P. Yang & J. Chen, "*All in the XL Family: Theory and Practice*", Preprint.