Cryptanalytic JH and BLAKE Hash Function for Authentication and Proposed Work Over BLAKE-256 Using C

Mr. Somil Jain[1], Jagannath University, Jaipur(Raj.), INDIA,

Mr. Vaibhav Doshi[2], Amity University Rajasthan, Jaipur(Raj.), INDIA,

Mr. Tarun Goyal[3]**,** SIET Rajasthan, Sikar(Raj.), INDIA,

Abstract: Hash functions form an important category of cryptography, which is widely used in a great number of protocols and security mechanisms. Hash functions are a fundamental primitive category of security science. It is defined as computationally efficient function, which maps binary strings of arbitrary length to binary strings of fixed length. The last ones are the outputs of a hash computation and they are called hash values. Hash functions are applied to support digital signatures, data integrity, random number generators, authentication schemes, and data integrity mechanisms. National Institute of Standard and technology (NIST) has selected the 14 Second Round Candidates of the SHA-3 Competition. It focus on the new SHA-3 competition, started by the NIST, which searches for a new hash function in response to authentication concerns regarding the previous hash functions SHA-1 and the SHA-2 family.

This work is based on comparative study of two SHA-3 cryptographic hash function candidates JH and BLAKE (out of fourteen). It compares the common features of both candidates which is widely used in above mentioned applications and improvement over SHA-256 with proposed work BLAKE-256 hash function.SHA-256 is used 64 rounds to calculate 256 bit final hash value with 256 bit initial hash value but BLAKE-256 is used 14 rounds to calculate final hash value with same input length and same output length.

Keyword: SHA-3, JH ,BLAKE , Hash, Compression Function.

INTRODUCTION

We proposed a new compression function structure to construct a compression function of JH from a block cipher with constant key. JH compression function is constructed from a bijective function (a block cipher with constant key) [4]. The block size of the block cipher is 2m bits,2m-bit hash value $H^{(i-1)}$ and the m-bit message block $M^{(i)}$ are compressed into the 2m-bit $H^{(i)}$. The BLAKE family contains four hash functions: BLAKE-28, BLAKE-32, BLAKE-48, BLAKE-64, with the bit lengths of their digests being 224,256,384 and 512, respectively. The former two operate on 32-bit words, while the latter two work with 64-bit words [3]. BLAKE comes in two variants: one that uses 32-bit words, used for computing hashes up to 256 bits long, and one that uses 64-bit words, used for computing hashes up to 512 bits long.

Compression Function is the main core of the hash functions data transformations, for a certain number of rounds for each hash function.

The compression function of BLAKE-256 takes as input four values [3]:

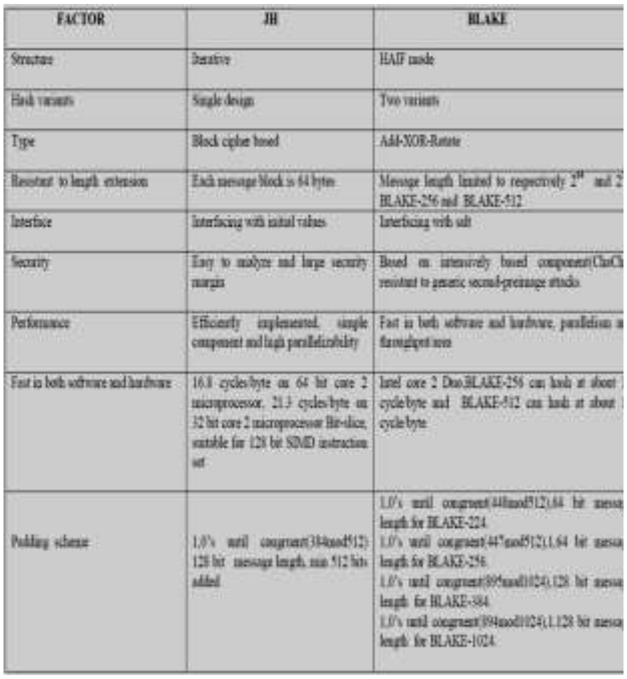
- Chain value $h = h_0, . . . , h_7$
- Message block $m = m_0, . . . ,m_{15}$
- Salt $s = s_0, . . . , s_3$
- Counter $t = t_0, t_1$

These four inputs represent 30 words in total (i.e., 120 bytes = 960 bits).

The output of the function is a new chain value $h' = h'_{0,\ldots,} h'_7$ of eight words (i.e., 32 bytes = 256 bits). We write the compression of h, m, s, t to h' as

$$\mathbf{h' = compress(h, m, s, t)}$$

COMPARATIVE STUDY OF JH AND BLAKE



FACTOR	JH	BLAKE
Structure	Iterative	HAIF mode
Hash variants	Single design	Two variants
Type	Block cipher based	Add-XOR-Rotate
Resistant to length extension	Each message block is 64 bytes	Message length limited to respectively 2^{64} and 2 BLAKE-256 and BLAKE-512
Interface	Interfacing with initial values	Interfacing with salt
Security	Easy to analyze and large security margin	Based on intensively based component(ChaCh resistant to generic second-preimage attacks
Performance	Efficiently implemented, simple component and high parallelizability	Fast in both software and hardware, parallelism a throughput area
Fast in both software and hardware	16.8 cycles/byte on 64 bit core 2 microprocessor, 21.3 cycles/byte on 32 bit core 2 microprocessor Bit-slice, suitable for 128 bit SIMD instruction set	Intel core 2 Duo.BLAKE-256 can hash at about cycle/byte and BLAKE-512 can hash at about cycle/byte
Padding scheme	1,0's until congruent(384mod512) 128 bit message length, min 512 bits added	1,0's until congruent(448mod512),64 bit messa length for BLAKE-224 1,0's until congruent(447mod512),1,64 bit messa length for BLAKE-256 1,0's until congruent(895mod1024),128 bit messa length for BLAKE-384 1,0's until congruent(894mod1024),1,128 bit messa length for BLAKE-1024

Table 1. Comparative Results of JH and BLAKE

SIMULATION WORK

It gives us C code computing the compression function of BLAKE-256.

- **Initialization**

  It calculates initialization step of BLAKE-256 with the input of initial chain value (hxdi), salt (sxdi) and counter (txdi) and generate state value (vxdo).

- **Round Function**

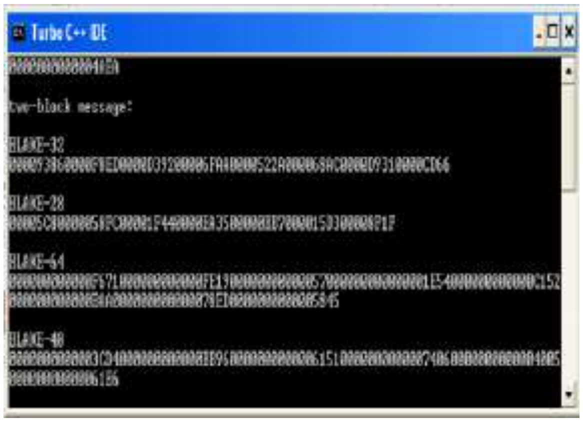
  It calculates round function step of BLAKE-256 with the input of state values (axdi, bxdi, cxdi, dxdi), message word (mxdi) and constant word (kxdi) which is generated by initialization step of BLAKE-256 , and then it generate new state values(axdo, bxdo, cxdo, dxdo).

- **Finalization**

  It calculates finalization step of BLAKE-256 with the input of final state values vxdi (which is generated after 14 round), salt (sxdi), initial chain value (hxdi) and then it generate final chain or hash value (hxdo).

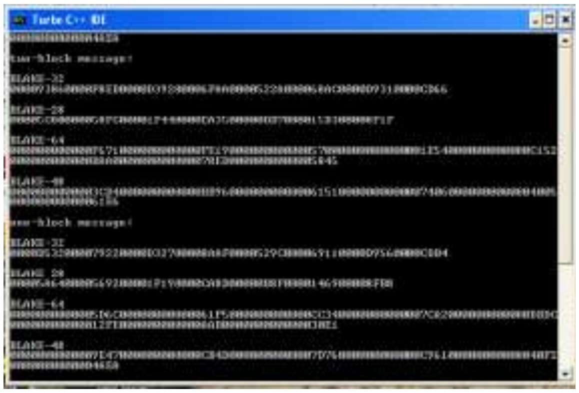
It gives us C code computing the compression function of BLAKE-256 with Header File BLAKEh.h and Turbo file Turbo.c.

Firstly, It calculates $G_0(v_0, v_4, v_8, v_{12})$



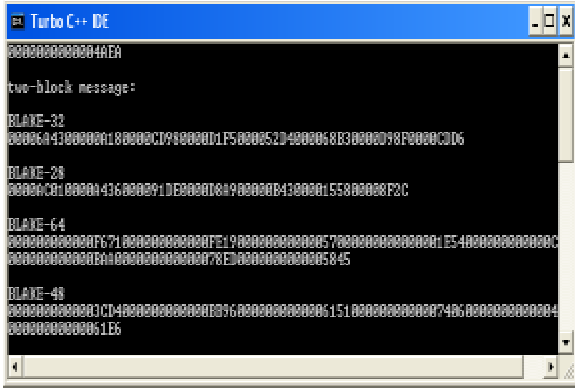
```
[illegible]

two-block message:

BLAKE-32
00009386000F9ED00000392000006FA40000522A000068AC00000D9310000CD66

BLAKE-28
00005C8000005FC80001F4400008EA3500000E70000153B0000AF1F

BLAKE-64
[illegible]
[illegible]

BLAKE-48
[illegible]
[illegible]
```

Figure1: Hash values Obtained after Two Block Message



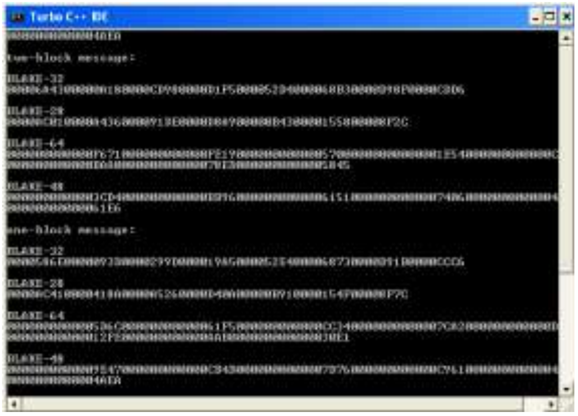
```
000000000000441A

two-block message:

BLAKE-32
000073B60000F8ED0000D39100006F9A0000522800006BAC0000D7310000C866

BLAKE-28
00005C8000005BFC00001F440000EA350000B3700015D3000000F1F

BLAKE-64
000000000000F671000000000000FE190000000000005700000000000001E54000000000000C152
000000000000EBA4000000000000078ED000000000000005845

BLAKE-48
0000000000003CD4000000000000E896000000000000615100000000000074060000000000004005
00000000000061E6

one-block message:

BLAKE-32
000053200000792200000D3270000AAF00000529C000069110000D7560000C8D4

BLAKE-28
0000A6A400005692000017190000CA830000818F000014650000EFB8

BLAKE-64
0000000000005D6C00000000000061F5000000000000CC340000000000007C82000000000000EDDC
000000000000127F000000000000A6AD0000000000003BE1

BLAKE-48
0000000000007E47000000000000C8430000000000007D76000000000000C7610000000000004BF1
0000000000004658
```

Figure2: Hash Value Obtained After one-block message



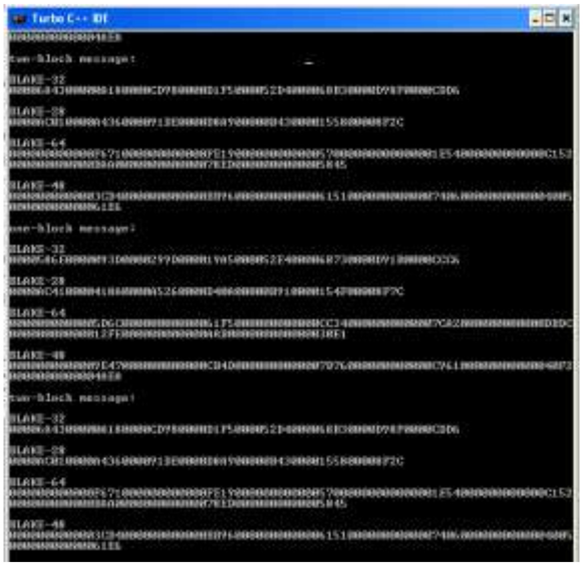
```
0000000000004AEA

two-block message:

BLAKE-32
00006A4300000A180000CD9F0000D1F5000052D4000068B30000D98F0000CDD6

BLAKE-28
0000AC810000A436000091DE0000D8490000B4300001558000008F2C

BLAKE-64
000000000000F671000000000000FE190000000000005700000000000001E5400000000000000C
0000000000000BA400000000000078ED0000000000005845

BLAKE-48
0000000000003CD4000000000000E89600000000000061510000000000007406000000000000004
00000000000061E6
```

Figure3 : Changed Hash Value After Two Block-Message



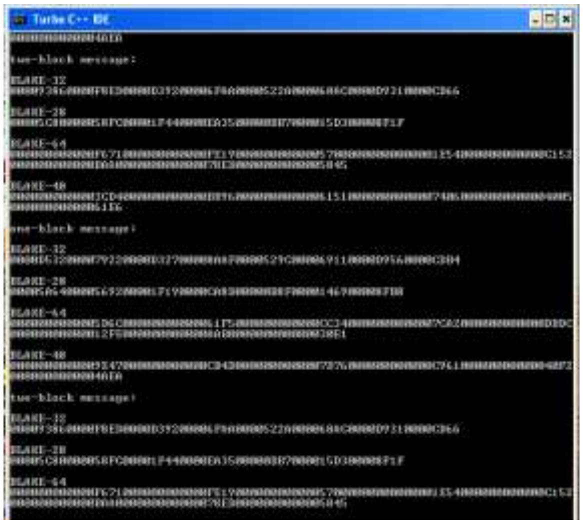
```
0000000000004AEA

two-block message:

BLAKE-32
00006A4300000A180000CD9F0000D1F5000052D4000068B30000D98F0000CDD6

BLAKE-28
0000AC810000A436000091DE0000D8490000B4300001558000008F2C

BLAKE-64
000000000000F671000000000000FE190000000000005700000000000001E5400000000000000C
0000000000000BA400000000000078ED0000000000005845

BLAKE-48
0000000000003CD4000000000000E896000000000000615100000000000074060000000000000004
00000000000061E6

one-block message:

BLAKE-32
00005A6E00009330000029900000A19A000052D4000068730000D9110000CCC6

BLAKE-28
0000C4180000411A000052600000D48600009100000154700000F7C

BLAKE-64
0000000000005D6C00000000000061F5000000000000CC340000000000007C82000000000000EDDC
000000000000127F000000000000A6AD0000000000003BE1

BLAKE-48
0000000000007E47000000000000C8430000000000007D76000000000000C76100000000000004
0000000000004AEA
```

Figure4: Changed Hash Value After One-Block Message



```
0000000000004658

two-block message:

BLAKE-32
00006A4300000A180000CD9F0000D1F5000052D4000068B30000D98F0000CDD6

BLAKE-28
0000AC810000A436000091DE0000D8490000B4300001558000008F2C

BLAKE-64
000000000000F671000000000000FE190000000000005700000000000001E54000000000000C152
0000000000000BA400000000000078ED0000000000005845

BLAKE-48
0000000000003CD4000000000000E8960000000000006151000000000000740600000000000004005
00000000000061E6

one-block message:

BLAKE-32
00005A6E00009330000029900000A19A000052D4000068730000D9110000CCC6

BLAKE-28
0000C4180000411A000052600000D48600009100000154700000F7C

BLAKE-64
0000000000005D6C00000000000061F5000000000000CC340000000000007C82000000000000EDDC
000000000000127F000000000000A6AD0000000000003BE1

BLAKE-48
0000000000007E47000000000000C8430000000000007D76000000000000C7610000000000004BF1
0000000000004AEA

two-block message:

BLAKE-32
00006A4300000A180000CD9F0000D1F5000052D4000068B30000D98F0000CDD6

BLAKE-28
0000AC810000A436000091DE0000D8490000B4300001558000008F2C

BLAKE-64
000000000000F671000000000000FE190000000000005700000000000001E54000000000000C152
0000000000000BA400000000000078ED0000000000005845

BLAKE-48
0000000000003CD4000000000000E8960000000000006151000000000000740600000000000004005
00000000000061E6
```

Figure5: Combined Hash Values Of One & Two Block- Message



```
0000000000004AEA

two-block message:

BLAKE-32
000073B60000F8ED0000D39100006F9A0000522800006BAC0000D7310000C866

BLAKE-28
00005C8000005BFC00001F440000EA350000B3700015D3000000F1F

BLAKE-64
000000000000F671000000000000FE190000000000005700000000000001E54000000000000C152
0000000000000BA400000000000078ED0000000000005845

BLAKE-48
0000000000003CD4000000000000E8960000000000006151000000000000740600000000000004005
00000000000061E6

one-block message:

BLAKE-32
000053200000792200000D3270000AAF00000529C000069110000D7560000C8D4

BLAKE-28
0000A6A400005692000017190000CA830000818F000014650000EFB8

BLAKE-64
0000000000005D6C00000000000061F5000000000000CC340000000000007C82000000000000EDDC
000000000000127F000000000000A6AD0000000000003BE1

BLAKE-48
0000000000007E47000000000000C8430000000000007D76000000000000C7610000000000004BF1
0000000000004658

two-block message:

BLAKE-32
000073B60000F8ED0000D39100006F9A0000522800006BAC0000D7310000C866

BLAKE-28
00005C8000005BFC00001F440000EA350000B3700015D3000000F1F

BLAKE-64
000000000000F671000000000000FE190000000000005700000000000001E54000000000000C152
0000000000000BA400000000000078ED0000000000005845
```

Figure6: Changed Combined Hash Value Of One & Two- Block Message

CONCLUSION

A cryptographic hash function is a hash function, that is, an algorithm that takes an arbitrary block of data and returns a fixed-size bit string, the (cryptographic) hash value, such that an (accidental or intentional) change to the data will (with very high probability) change the hash value. In this work the comparison of two SHA-3 cryptography hash function JH and BLAKE, less time consuming process BLAKE-256 with 14 rounds instead of 64 rounds of SHA-256, more reliable compare to SHA family. This work gives the comparative hash algorithms of both candidates.

FUTURE WORK

In future the working of the function (BLAKE-256) discussed here should be more less time consuming. It can be merge all three function of BLAKE-256 into single unit in VHDL then give one time input (256 bit) and get 256 bit final hash value.

REFERENCES

[1] FPGA Based Area And Throughput Implementation of JH And BLAKE Hash Function http://www.ijcttjournal.org/volume-3/issue-2/IJCTT-V3I2P112.pdf

[2] Comparative Study of Two SHA-3 Cryptographic Hash Function JH and BLAKE Vaibhav Doshi ,Richa Arya,Birbal Saran. April 21-23,2012. ICRTCTA 2012. Amity School of Engineering and Technology

[3] J. P. Aumasson , L. Henzen , W. Meier, and R.C.W. Phan, "SHA-3 Proposal BLAKE", online: http://www.131002.net/blake 2010.

[4] Wu, H., SHA-3 proposal JH, Website: http://icsd.i2r.astar. edu.sg/staff/hongjun/jh/.

[5] National Institute of Standard and Technology (NIST):Cryptographic Hash Algorithm Competition Website: http://csrc.nist.gov/groups/ST/hash/sha-3/.

[6] J.P. Aumasson, L. Henzen, W. Meier, R.C.W. Phan, "SHA-3 proposal BLAKE". Submission to the SHA-3 Competition, 2008.

[7] E. Biham, O. Dunkelman, "A framework for iterative hash functions- HAIFA".Cryptology ePrint Archive, Report 2007/278.

[8] Mao Ming,He Qiang ,Shaokun Zeng Xidian University Xi'an , Shanxi, China BLAKE- 32 based on differential properties, 2010 International Conference on Computational and Information Sciences.

[9] George Provelengios,Nikolaos S. Voros,Paris Kitsos Greece, 2011 14th Euromicro Conference on Digital System Design

[10] SHA-1 Standard, National Institute of Standards and Technology (NIST), Secure Hash Standard, FIPS PUB 180-1, 1995, available on line at www.itl.nist.gov/fipspubs/ fip180 1.htm

[11] Secure Hash Standard (SHS), National Institute of Standards and Technology(NIST),FIPS P UB 180-3,2008,available online at http://csrc.nist.gov/publications/fips/fips1803/fips180-3_final.pdf

[12] IEEE Annual Symposium Nicolas Sklavos 2010,ZIP 27100,GREECE,Paris Kitsos, Comp uter Science Hellenic Open University, GREEC.

[13] J.-L. Beuchat, E. Okamoto, and T. Yamazaki, "Compact implementations of BLAKE- 32 and BLAKE-64 on FPGA," in FPT, 2010, pp. 170–177.

[14] H. Wu. SHA-3 proposal JH, version January 15, 2009. JH

[15] Bernhard Jungk and J¨urgen Apfelbeck, Hochschule RheinMain, Wiesbaden, Germany 2011 International Conference on Reconfigurable Computing

[16] D. J. Bernstein. ChaCha, a variant of Salsa20. Available online at http://cr.yp. to/chacha/chacha-20080128.pdf, January 2008.

[17] E. Biham and O. Dunkelman. A Framework for Iterative Hash Functions - HAIFA. In Second NIST Cryptographic Hash Workshop, Santa Barbara, California, USA, August 24-25, 2006, August 2006.

[18] Cadence Design Systems. The Cadence Design Systems Website. http://www.cadence.com/.

[19] C. D. Canni_ere, H. Sato, and D. Watanabe. Hash Function Lufia, Specification Ver. 2.0.Available online at http://www.sdl.hitachi. co.jp/crypto/luffa/ Luffa_v2_Specificati on _20090915.pdf, September 2009.

[20] SHA-512/256 Shay Gueron 1, 2, Simon Johnson 3, Jesse Walker4 Department of Mathematics, University of Haifa, Israel Mobility Group, Intel Corporation, Israel Development Center, Haifa, Israel Intel Architecture Group, Intel Corporation, USA Security Research Lab, Intel Labs, Intel Corporation, USA

[21] H. Namin and M. A. Hasan, Waterloo, Ontario N2L 3G1 Canada. Compression Function for Selected SHA-3 Candidates.

[22] Iterative Difierentials, Symmetries, and Message Modification in BLAKE-256 Or Dunkelman and Dmitry Khovratovich University of Haifa, Israel Weizmann University, Israel Microsoft Research Redmond, USA

[23] Hongjun Wu, “The Hash Function JH”, The First SHA-3 Candidate Conference, 2009, available online at http://ehash.iaik.tugraz.at/wiki/JH