MAYHAM – A New Hash Function

Naila Shakeel, Ghulam Murtaza, and Nassar Ikram
(Corresponding author: Ghulam Murtaza)

National University of Sciences and Technology, Sector H-10, Islamabad, Pakistan
(Email: {nailashakeel, azarmurtaza}@hotmail.com, dr_nassar_ikram@yahoo.com)

(Received Mar. 15, 2011; revised and accepted Nov. 15, 2011)

Abstract

With the succumbing of various Hash functions to collision attacks, there have been serious research efforts to design new Hash functions which are robust against various contemporary attacks. MAYHAM is one such hash function that has been designed keeping in view the cryptographic properties needed yet resistant to all the publically known attacks. MAYHAM is designed as a simple and efficient hash function as compared to other dedicated hash functions. Main operations involved in MAYHAM's hash computation include S-box, MDS Matrix multiplication, data mixing, XOR, addition and interleaving. MAYHAM supports parallel processing in its compression function and can be efficiently implemented by using lookup tables.

Keywords: Data mixing, hash function, MDS matrix, nonlinearity, S-Box

1 Introduction

A hash function is a one way function that maps an input of arbitrary length into a fixed length output sequence of bits. An account of cryptographic hash functions can be found in Menezes et al. [4]; while a more recent survey is provided by Preneel [5]. A secure cryptographic hash function has to satisfy the requirements of preimage resistance, second preimage resistance and collision resistance [3].

The recent attacks on MD4 [9], MD5 [2, 6, 11], SHA-0 [12] and SHA-1 [10, 14] by Wang et al have given a major impetus to research in designing new cryptographic hash functions as well as cryptanalysis of the existing ones. The cryptographic community is now on the lookout for a new hash function aimed at replacing SHA-1. NIST has opened a new competition for SHA-3 pressing the need for more robust and secure hash function, similar initiative was earlier taken under NESSIE and CRYPTREC [8, 13, 15, 16]. This paper proposes a new hash function, MAYHAM that is resistant to various attacks which are presently known. It offers attributes of speed as well as simplicity in implementation on processors of different bits.

The paper is organized as follows: Section 2 gives design specification and details of MAYHAM, Section 3 explains design rationale, Section 4 treats the security analysis, Section 5 covers the performance analysis and Section 6 gives the conclusion.

2 Design Specifications

The algorithm is divided into two stages: preprocessing and hash computation. The former is akin to the structure used in MD4. Preprocessing involves the padding of the message (to make it divisible by 512), parsing the padded message into *m*-bit blocks of 512 bits each, and setting initialization values to be used in the second stage i.e., hash computation. In this stage, a compression function is used iteratively on each block of 512 bits to compress them all into an output of 256 bits. Main operations involved in MAYHAM's hash computation include S-box, MDS Matrix multiplication, data mixing, XOR, addition and interleaving. Lookup tables based implementations of S-box (8x16) and MDS Matrix (4x2) enable realization of competitive speed, a highly desired attribute for hash algorithms.

2.1 Parameters and Acronyms

The following parameters are used in MAYHAM:

$H_0^{(0)}, H_1^{(0)}, \ldots, H_{15}^{(0)}$	Initial Hash values
X	Message to be hashed
M	Message length in bits
$X^{(i)}$	Message block i, with a size of m bits.
$X_j^{(i)}$	The j^{th} word of the i^{th} message block, where $X_0^{(i)}$ is the left-most word of message block i
$A_0, A_1, \ldots, A_{15}$	Working variables that are 32-bit words used in the computation of the hash values
$a_{i,j}$	j^{th} byte of the i^{th} working variable
K_t	Constant value to be used for iteration t of the hash computation
$H_0^{(C)}, H_1^{(C)}, \ldots, H_7^{(C)}$	Current message block hash value
$H_0^{(F)}, H_1^{(F)}, \ldots, H_7^{(F)}$	Final hash value
M	Matrix
M^T	Transpose of the Matrix M
DM	Data Mixing

F_R	Round Function
F_O	Out Function
MDP	Maximum Differential Probability
DP	Differential Probability
LUT	Look Up Table
SPN	Substitution Permutation Network
IV	Initialization Vector

2.2 Notations

The following symbols are used in MAYHAM, and each operates on 32-bit words:

$\oplus$	bit-wise exclusive-or operation
$\boxplus$	Addition modulo 2^{32}
$\bowtie$	Interleaving of bits into 32 bits

2.3 Constants

MAYHAM uses a sequence of sixty-four constants of size 32-bit each i.e., K_0, K_1, …, K_{63} that are generated randomly. In hex, these constant words are (from left to right):

2678d3b9; a2100094; 1251db5d; e425eb7f; 6f23988f ; 2ca5adf6; ce079e3b; 27267f75; 7cc590c6; 565d7491; fb224be5; 3b3abd0c; 8343ccc9; 13a1d2ba; bb55ea81; cbdcea9a; 71c4174d; c0140686; b05a834d; d7b354f2; e7a0728b; 3bd428db; f3d50b01; 73f3c61f; 37b07377; 236f4b6e; 3c4619b8; 1e85e207; 00f8992a; 695f5f9c; 9b3c5aea; 090513bd; 94631527; 3fddce78; 55034ff5; bb70bc6d; f304007b; 599cb5d8; 50d7dbe6; 15606536; 325b23ba; 7484a4ba; 0a923572; 0c458ef1; 5f887fe8; d882d15e; a6276c1d; 8e45e65f; 8786eca0; 6a8a2132; 0754ea2b; 4cb946f4; 508daebd; 1f9f9707; 1ee8c64d; 18862cc4; abd6ea86; 59ed86d5; 664de184; 97a611ec; e078cd00; 972840e8; eee48181; c3d41980

2.4 Initial Hash Value ($H^{(0)}$)

The initial hash value, $H^{(0)}$ consists of the following sixteen 32-bit words, in hex that are generated randomly in preprocessing stage:

$H_0^{(0)}$ = cfc09a62; $H_1^{(0)}$ = 06d956d1; $H_2^{(0)}$ = bd126f78;
$H_3^{(0)}$ = d7da6ca0; $H_4^{(0)}$ = b92c2d82; $H_5^{(0)}$ = 96029774;
$H_6^{(0)}$ = 6b041292; $H_7^{(0)}$ = 34605d71; $H_8^{(0)}$ = 269b1c20;
$H_9^{(0)}$ = dd6af9bd; $H_{10}^{(0)}$ = d16e2adb; $H_{11}^{(0)}$ = 7409aae8;
$H_{12}^{(0)}$= 6779c7fb; $H_{13}^{(0)}$ = f8fddfcd; $H_{14}^{(0)}$ =2826f702;
$H_{15}^{(0)}$ = c6c49d5d

2.5 The Algorithm MAYHAM

The algorithm is divided into two stages which, in order, are: preprocessing and hash computation. Preprocessing is done in three steps: padding the message, parsing the padded message into message blocks of size 512 bits each, and setting the initialization values mentioned in Para 2.3 and 2.4. After preprocessing, each message block $X^{(i)}$ of 512 bit is divided into sixteen 32 bit words i.e., $X_0^{(i)}$, $X_1^{(i)}$,…, $X_{15}^{(i)}$ and successively processed through MAYHAM function comprising of the following (refer to Figure 1):

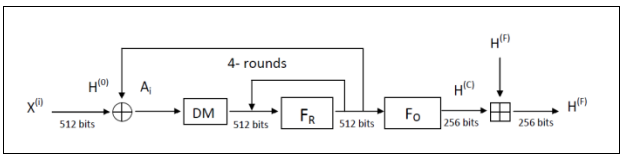

Figure 1: MAYHAM hash algorithm

XOR Operation. Message words $\mathbf{X^{(i)}}$'s are XORed with the initial values $H_j^{(0)}$ to get the working variables A_j where $0 \le j \le 15$.

$$H_j^{(0)} \oplus X_j^{(i)} = A_j.$$

where $X_j^{(i)}$ is the j^{th} word of the i^{th} message block and $0 \le j \le 15$.

Data Mixing (DM) Function. Data mixing function is shown in the following figure.

After data mixing the following transformations of the working variables are obtained:

$A_0 = 8A_0+4A_1+2A_2+4A_3+A_4+A_5+A_6+A_7+A_8+A_9+2A_{10}+2A_{11}+2A_{12}+A_{13}+A_{14}+2A_{15}.$

$A_1 = 2A_0+2A_1+2A_2+2A_3+4A_4+2A_5+A_6+2A_7+2A_8+A_9+A_{10}+2A_{11}+A_{12}+A_{13}+2A_{14}+2A_{15}.$

$A_2 = 2A_0+2A_1+4A_2+4A_3+2A_4+A_5+A_6+2A_7+4A_8+2A_9+A_{10}+2A_{11}+A_{12}+A_{13}+A_{14}+A_{15}.$

$A_3 = 4A_0+2A_1+2A_2+4A_3+A_4+A_5+2A_6+2A_7+A_8+A_9+A_{10}+A_{11}+4A_{12}+2A_{13}+A_{14}+2A_{15}.$

$A_4 = A_0+A_1+A_2+A_3+4A_4+2A_5+A_6+2A_7+4A_8+2A_9+2A_{10}+4A_{11}+A_{12}+A_{13}+2A_{14}+2A_{15}.$

$A_5 = 4A_0+2A_1+A_2+2A_3+A_4+A_5+A_6+A_7+2A_8+2A_9+4A_{10}+4A_{11}+2A_{12}+A_{13}+A_{14}+2A_{15}.$

$A_6 = 2A_0+A_1+A_2+2A_3+A_4+A_5+2A_6+2A_7+2A_8+2A_9+2A_{10}+2A_{11}+4A_{12}+2A_{13}+A_{14}+2A_{15}.$

$A_7 = A_0+A_1+2A_2+2A_3+2A_4+A_5+A_6+2A_7+8A_8+4A_9+2A_{10}+4A_{11}+A_{12}+A_{13}+A_{14}+A_{15}.$

$A_8 = A_0+A_1+2A_2+2A_3+2A_4+A_5+A_6+2A_7+16A_8+8A_9+4A_{10}+8A_{11}+2A_{12}+2A_{13}+2A_{14}+2A_{15}.$

$A_9 = 2A_0+A_1+A_2+2A_3+A_4+A_5+2A_6+2A_7+4A_8+4A_9+4A_{10}+4A_{11}+8A_{12}+4A_{13}+2A_{14}+4A_{15}.$

$A_{10} = 4A_0+2A_1+A_2+2A_3+A_4+A_5+A_6+A_7+4A_8+4A_9+8A_{10}+8A_{11}+4A_{12}+2A_{13}+2A_{14}+4A_{15}.$

$A_{11} = A_0+A_1+A_2+A_3+4A_4+2A_5+A_6+2A_7+8A_8+4A_9+4A_{10}+8A_{11}+2A_{12}+2A_{13}+4A_{14}+4A_{15}.$

$A_{12} = 4A_0+2A_1+2A_2+4A_3+A_4+A_5+2A_6+2A_7+2A_8+2A_9+2A_{10}+2A_{11}+8A_{12}+4A_{13}+2A_{14}+4A_{15}.$

$A_{13} = 2A_0+2A_1+4A_2+4A_3+2A_4+A_5+A_6+2A_7+8A_8+4A_9+2A_{10}+4A_{11}+2A_{12}+2A_{13}+2A_{14}+2A_{15}.$

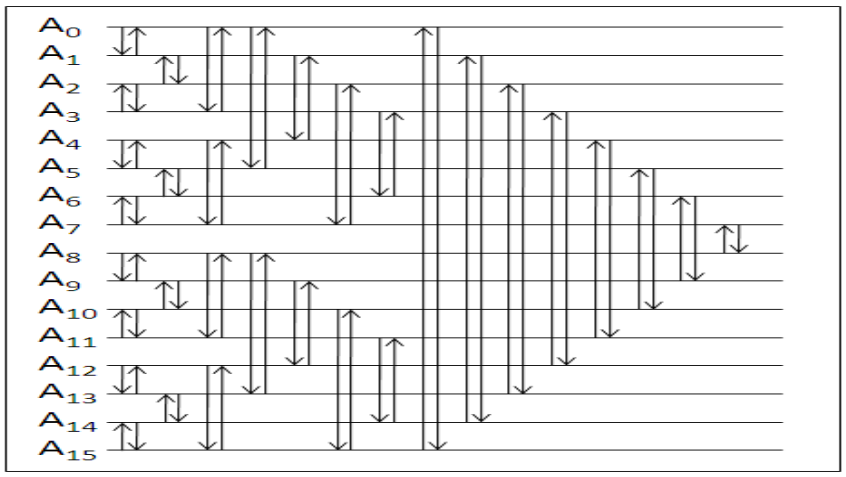

Figure 2: Data Mixing (DM) function

$$A_{14}=2A_0+2A_1+2A_2+2A_3+4A_4+2A_5+A_6+2A_7+4A_8+2A_9+2A_{10}+4A_{11}+2A_{12}+2A_{13}+4A_{14}+4A_{15}.$$
$$A_{15}=8A_0+4A_1+2A_2+4A_3+A_4+A_5+A_6+A_7+2A_8+2A_9+4A_{10}+4A_{11}+4A_{12}+2A_{13}+2A_{14}+4A_{15}.$$

Round Function (F_R). Round function F_R operates 4 times for every message block of size 512 bits. After every iteration, the resultant value of the round function F_R will become the next input for the round function F_R. The Round Function F_R is illustrated in Figure 3 and comprises of sub processes involving Matrix Transposition, *g*-function, MDS Matrix multiplication and *S*-function before giving its output. Each of these processes has been explained hereunder.

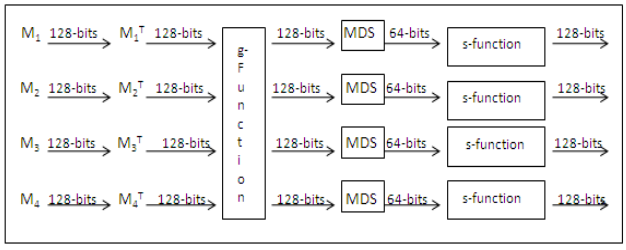

Figure 3: Round function *(F_R)*

Where

$$M_1 = \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}_{(4x4)}.$$

$$M_2 = \begin{bmatrix} A_4 \\ A_5 \\ A_6 \\ A_7 \end{bmatrix} = \begin{bmatrix} a_{4,0} & a_{4,1} & a_{4,2} & a_{4,3} \\ a_{5,0} & a_{5,1} & a_{5,2} & a_{5,3} \\ a_{6,0} & a_{6,1} & a_{6,2} & a_{6,3} \\ a_{7,0} & a_{7,1} & a_{7,2} & a_{7,3} \end{bmatrix}_{(4x4)}.$$

$$M_3 = \begin{bmatrix} A_8 \\ A_9 \\ A_{10} \\ A_{11} \end{bmatrix} = \begin{bmatrix} a_{8,0} & a_{8,1} & a_{8,2} & a_{8,3} \\ a_{9,0} & a_{9,1} & a_{9,2} & a_{9,3} \\ a_{10,0} & a_{10,1} & a_{10,2} & a_{10,3} \\ a_{11,0} & a_{11,1} & a_{11,2} & a_{11,3} \end{bmatrix}_{(4x4)}.$$

$$M_4 = \begin{bmatrix} A_{12} \\ A_{13} \\ A_{14} \\ A_{15} \end{bmatrix} = \begin{bmatrix} a_{12,0} & a_{12,1} & a_{12,2} & a_{12,3} \\ a_{13,0} & a_{13,1} & a_{13,2} & a_{13,3} \\ a_{14,0} & a_{14,1} & a_{14,2} & a_{14,3} \\ a_{15,0} & a_{15,1} & a_{15,2} & a_{15,3} \end{bmatrix}_{(4x4)}.$$

Transpose Matrix. M_1^T is the transpose of matrices M_1, M_2^T is the transpose of matrices M_2 and so on. After taking transpose of each matrix, these matrices are now passed through the *g*-Function.

g-Function. Following equations define the g-Function

$$A_4 = A_4+A_0.$$

$A_9 = A_9 + A_5$.

$A_{14} = A_{14} + A_{10}$.

$A_3 = A_3 + A_{15}$.

MDS Matrix. MDS Matrix of order (4x2) defined over irreducible polynomial 283 (in decimal) is used in MAYHAM. The working variables A_i's, previously of size 32-bits each, get reduced to 16-bits each (Appendix A) after passing through MDS Matrix.

s-function. It consists of S-box of size 8 x 16, sixty four 32-bits round constants and interleaving of two consecutive output bits from the S-box into 32 bits. Interleaving is performed in a usual way in which output bits of S-Box when accessed the first time are placed in odd position while output bits of S-Box when accessed the second time are placed in even positions. The s-function operates the following way:

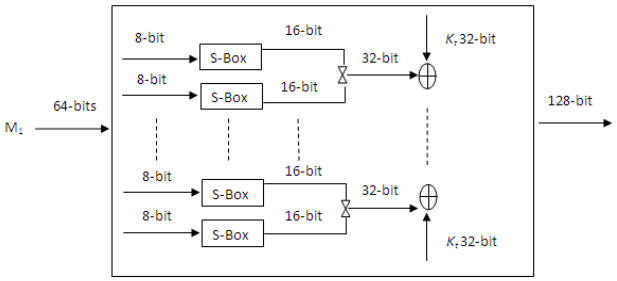

Figure 4: s-function

IV-Assignment. After 4 iterations of round function F_R, the initial hash values are replaced with working variable values, i.e.

$$H_j^{(0)} = A_j, \qquad \text{where } 0 \le j \le 15.$$

Out Function - F_O. The F_O is described by the following equation:

$$H_j^{(C)} = A_{2j} + A_{2j+1}, \text{ where } 0 \le j \le 7.$$

Final Hash Value.

$$H_j^{(F)} = H_j^{(F)} + H_j^{(C)}, \quad \text{where } 0 \le j \le 7.$$

Test vectors for MAYHAM hash algorithm are given in Appendix B.

3 Design Rationale

Each component of MAYHAM algorithm individually complies with its respective security criteria, thus ensuring that the algorithm is secure and collision free.. Rationale for selecting different sub-functions within each stage i.e. Preprocessing and Hash Computation either from the perspective of security or efficient implementation, is given below:

1. Padding in preprocessing stage pads the original input data to thwart attacks based on fixed points [7]. In such attacks, the attacker tries to produce second preimages or collisions, by inserting extra blocks into the input.
2. The iteration mode used in Hash Computation stage is based on the well known Merkle-Damgard construction [1], which enables compression function to be collision resistant.
3. In data mixing function of Hash Computation stage, parallel processing is done to mix the data in 4 steps and so is done for round function F_R, for efficient implementation. Other operations used in MAYHAM include addition 2^{32}, interleaving, XOR and LUTs ensure efficient realization.
4. Different round constants, though generated in Preprocessing stage but used in Hash computation, are employed for each round. This improves the security of the overall hash function.
5. In order to get the best possible diffusion, MDS Matrix is used which has the maximum branch number. The MDS is efficiently implemented with two LUTs.
6. g-Function is used to add global diffusion for round function F_R.
7. s-function is used to add confusion and non linearity to the data. S-box used in the s-function is balanced having nonlinearity of 84. Interleaving is used to mix the output bits of two consecutive S-boxes.
8. Round function F_R is processed 4 times, considered most appropriate to get complete data diffusion for g-function.
9. F_O is irreversible function in order to prevent preimage.

4 Security Analysis

Following security attributes are offered by MAYHAM:

1. Nonlinearity.
   The nonlinearity comes mostly from S-box of size 8 x 16 and interleaving of 32 bits. This is better than merely combining additions and XORs (i.e., using the carry bits) and it affects all the output bits, not just the neighboring bits.
2. Resistance against Differential Analysis.
   The round function of MAYHAM is of SPN type structure and uses S-box having MDP of 2-6. In first round the structure ensures 8 active S-boxes which increases to 32 active S-boxes each in the following three rounds. For all four rounds there are 104 active S-boxes which together give a DP of 2-624 which is much lesser than 2-256. In addition, g-function and interleaving of 32 bits destroys differential patterns and adds complexity for differential analysis.
3. Protection against Slide Attack.
   Different round constants are used in each round to avoid the slide attack.
4. Avalanche Effect.
   There is a strong avalanche; each message bit affects all the output bits due to SPN type structure.

5 Performance Analysis

Software implementations of MAYHAM were tested on system with Intel based CPUs Dual Core 3.34GHz with 1 GB RAM. The comparison is given in the following table for various hash functions tested on 0.8 Mb data file. It shows that MAYHAM has the third fastest output after RIPEMD-128 and RIPEMD-160.

Table 1: Comparison of software implementation of MAYHAM with MD5, SHA-256, SHA-224, SHA-512, RIPEMD-128 and RIPEMD-160

Algorithm	Time(ms)
MAYHAM	282
MD5	312
SHA-256	437
SHA-224	453
SHA512	422
RIPEMD-128	250
RIPEMD-160	250

6 Conclusion

In this paper we proposed a new 256-bit hash function which is designed not only to be secure but also faster and simpler than other known hash functions. MAYHAM possesses the property of being resistant to collision. However, unlike other dedicated hash functions, MAYHAM uses S-box, MDS Matrix and Data mixing function, together these functions offer security attributes distinct from others in withstanding various cryptanalytic attacks. MAYHAM is processed in parallel and is fast in software using Look up tables. Its hardware implementation is also envisaged to be efficiently realized.

References

[1] I. Damgard, *A Design Principle for Hash Functions*, http://saluc.engr.uconn.edu/refs/algorithms/hashalg/damgard89adesign.pdf

[2] V. Klima, "Finding MD5 collisions on a notebook PC using multi-message modifications", Cryptology ePrint Archive, 2005. http://eprint.iacr.org/2005/102.pdf

[3] L. R. Knudsen, "Small size hashes with enhanced security", *International Journal of Network Security*, vol. 2, no. 1, pp. 41-42, 2006.

[4] A. J. Menezes, P. C. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Washington, D.C., 1997.

[5] B. Preneel, *Hash functions-Present State of Art*, ECrypt Conference on Hash Functions, 2005. http://www.ecrypt.eu.org/stvl/hfw/

[6] R. Rivest, *The MD5 Message-Digest Algorithm. Request for Comments (RFC) 1321*, Internet Activities Board: Internet Privacy Task Force, 1992.
http://tools.ietf.org/html/rfc1321

[7] B. V. Rompay, *Analysis and Design of Cryptographic Hash Functions, MAC Algorithms and Block Ciphers*, Thesis, Katholieke University Leuven, 2004.

[8] R. Tirtea, "Cryptographic hash functions, trends and challenges," *Journal of Computer and System Sciences*, vol. 2, pp. 62-65, 2009.

[9] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, "Cryptanalysis of the hash functions MD4 and RIPEMD," in *Eurocrypt 2005*, LNCS 3494, pp. 1-18, Springer-Verlag, Heidelberg, 2005.

[10] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full SHA-1," in *Crypto 2005*, LNCS 3621, pp. 17-36, Springer-verlag, Heidelberg, 2005.

[11] X. Wang and H. Yu, "How to break MD5 and other hash functions," in *Eurocrypt 2005*, LNCS 3494, pp. 19-35, Springer-Verlag, Heidelberg, 2005.

[12] X. Wang, H. Yu, and Y. L. Yin, "Efficient collision search attacks on SHA-0," in *Crypto 2005*, LNCS 3621, pp. 1-16, Springer, Heidelberg, 2005.

[13] CRYPTography Research and Evaluation Committees (CRYPTEC).
http://www.ipa.go.jp/ security/enc/CRYPTREC/index-e.html

[14] National Institute of Standards and Technology. http://csrc.nist.gov/publications/fips/fips180-**2**/fips180-**2**.pdf

[15] National Institute of Standards and Technology (NIST), http://csrc.nist.gov/groups/ST/hash/index.html

[16] New European Schemes for Signature, Integrity, and Encryption (NESSIE). https://www.cosic.esat.kuleuven.ac.be/nessie/deliverables/press_release_feb27.pdf

Naila Shakeel received her M.Phil. degree in Statistics from Government College University, Lahore, Pakistan in 2004. She is currently a PhD Student/Research Associate in National University of Sciences and Technology (NUST), Pakistan. Her current research interest includes Digital Signal Processing and Cryptography.

Ghulam Murtaza received his M.Sc. degree in Mathematics from Government College University, Lahore, Pakistan in 2000. He worked as a software engineer in public/private sector organizations for 5 years. He did M.S. Information Security from Sichuan University, Chengdu, China in 2007. He is currently a Research Associate in National University of Sciences and Technology (NUST), Pakistan. He also works in symmetric key cryptology at Horizon, Islamabad, Pakistan. His current research interest includes Block Cipher Design & Analysis and Multiple Layered Ciphers.

Nassar Ikram graduated in Electrical Engineering from NED, Karachi, Pakistan in 1987. He received his M.Sc. in Military Electronics and Systems Engineering from Royal Military College of Sciences, Shrivenham, Cranfield University, UK in 1995 and PhD in Information Security from Bradford University, UK in 1999. He is currently Professor at National University of Sciences and Technology, Islamabad, Pakistan. His current research interests include multimedia system security and

Cryptography.

Appendix: A

S-Box(in hexadecimal):

4F6F	5B8E	888B	9E79	97A9	60E4	6B3E	C6AC
B61D	AD00	D349	4018	65C3	F5AC	1E7F	7539
2347	4D9F	2B10	CD1D	78E2	BFA3	385A	E48D
75C5	122E	F648	A086	C281	F7B0	A0AB	3C55
8C72	ED77	E57A	6855	E11C	40DF	5AA	3EE2
B67D	CCA4	A89C	CAD3	EC14	60F9	8758	5B59
A48A	B998	AF6C	C72	2921	F640	BAD6	8FD3
D36A	42DF	6B19	8BF8	2FA0	501A	EF2	E93
3096	758D	992F	256F	111B	F4B4	2F37	25E6
283E	DFFD	7303	DD43	BCCC	5A46	D79C	6312
7ACC	8622	DB12	8F36	F698	102F	D229	1E81
E4FF	F165	95CA	FD23	A1E9	36D6	530F	5F3A
84CF	5A3C	5D93	236B	BB4B	779B	CB11	A32D
CF14	4813	1EE6	B5C	F802	3C7A	679E	3F44
2F0F	4635	F	DE67	67B2	7E40	9415	EF50
2CC7	A284	97CA	AEE8	BDF0	82C5	F86D	D2A8
16F	3EBD	F5AD	61A5	FE7	E26C	D5B7	1693
60E2	5911	C176	6820	40F9	8113	149	7DB7
11CC	43DF	41E6	5FE7	18E3	DA60	C1E7	E34
3FDD	27D	BA28	CC44	9BF0	274E	B05A	CC36
6475	5427	559	5F12	26D8	C6CF	D6C4	8FAC
1C3F	FDED	C49E	6AF0	2288	9637	32C1	9A86
2BB1	8AFB	E8BC	9F82	12C2	27DE	AACC	B378
F4	57FD	9894	5369	9A81	7D91	5D4E	14BA
20CC	8823	B3F8	C078	9A13	91B1	D7B6	F0BF
9F48	E9C2	211E	4102	B184	5C5F	D902	E9C5
3819	3845	C71D	74B9	8C48	7BB1	DFF0	9428
F9E2	D53B	99C5	4165	C23	F346	94F9	164A
E2B0	900	45AC	A901	AFF5	3285	91C6	2006
7EEA	6FB5	E2DF	6BB6	FF76	8873	DC73	7001
2B3A	EE89	3A76	85BD	78D1	322F	C14A	5087
EC7E	9371	6C2B	1F6F	279B	9ED4	6EB3	2549

$$MDS = \begin{bmatrix} 1 & 1 \\ 1 & 51 \\ 51 & e1 \\ e1 & 1 \end{bmatrix}$$

Appendix: B

Test Vectors

All results are given in Hex format.

Test Vectors (Single block message)

Message: M, be the 24-bit ASCII string "abc", which is equivalent to the following binary string:

01100001 01100010 01100011

The initial hash value, $H^{(0)}$, is

$H_0^{(0)}$= cfc09a62 $H_1^{(0)}$=06d956d1 $H_2^{(0)}$=bd126f78 $H_3^{(0)}$=d7da6ca0
$H_4^{(0)}$=b92c2d82 $H_5^{(0)}$=96029774 $H_6^{(0)}$=6b041292 $H_7^{(0)}$=34605d71
$H_8^{(0)}$=269b1c20 $H_9^{(0)}$=dd6af9bd $H_{10}^{(0)}$ = d16e2adb $H_{11}^{(0)}$=7409aae8
$H_{12}^{(0)}$=6779c7fb $H_{13}^{(0)}$=f8fddfcd $H_{14}^{(0)}$ = 2826f702 $H_{15}^{(0)}$=c6c49d5d

The words of the padded message block are assigned to X_0, $X_1,\ldots,X_{15}$. These variables are the input for the MAYHAM hash Algorithm and are shown below:

X_0 = 61626380 X_1 = 00000000 X_2 = 00000000 X_3 = 00000000
X_4 = 00000000 X_5 = 00000000 X_6 = 00000000 X_7 = 00000000
X_8 = 00000000 X_9 = 00000000 X_{10}= 00000000 X_{11}= 00000000
X_{12}= 00000000 X_{13}= 00000000 X_{14}= 00000000 X_{15}= 00000018

Message blocks XORed with the initial hash values to get the working variables A_0, $A_1,\ldots,A_{15}$.

aea2f9e2	06d956d1	bd126f78	d7da6ca0
b92c2d82	96029774	6b041292	34605d71
269b1c20	dd6af9bd	d16e2adb	7409aae8
6779c7fb	f8fddfcd	2826f702	c6c49d45

Data Mixing

6536536f	9bbe8ef8	f8f2f266	bed5c848
357814d5	bf2636f0	376d209b	2860ce28
954bd381	0ffc08ed	5ab35b05	3c0c4a7b
4de6c4fa	571a131a	be3011f6	71cdb621

Intermediate results for the first round of F_R
After transposing the data

659bf8be	36bef2d5	538ef2c8	6ff86648
35bf3728	78266d60	143620ce	d5f09b28
950f5a3c	4bfcb30c	d3085b4a	81ed057b
4d57be71	e61a30cd	c41311b6	fa1af621

g-Function

659bf8be	36bef2d5	538ef2c8	6a135c69
9b5b2fe6	78266d60	143620ce	d5f09b28
950f5a3c	c423206c	d3085b4a	81ed057b
4d57be71	e61a30cd	971b6d00	fa1af621

MDS, S-Box, Interleaving and XORing with round constants

d3cb954a	54befb63	78b60437	947f59c5
25f6e9a8	aee21c8e	8ceb4b7c	dcb77a38
894d7799	2e4d9dcd	b0c8a857	223f1df9
14729458	f05f29aa	d0a99b91	676dd073

Intermediate results for the second round of F_R
After transposing the data

d3547894	cbbeb67f	95fb0459	4a6337c5
25ae8cdc	f6e2ebb7	e91c4b7a	a88e7c38
892eb022	4d4dc83f	779da81d	99cd57f9
14f0d067	725fa96d	94299bd0	58aa9173

g-Function

d3547894	cbbeb67f	95fb0459	a30dc938
f9030570	f6e2ebb7	e91c4b7a	a88e7c38
892eb022	4430b3f6	779da81d	99cd57f9
14f0d067	725fa96d	0bc743ed	58aa9173

MDS, S-Box, Interleaving and XORing with round constants

6a1554ca	69c2581a	caa3a21c	465cb6ac
a95dced3	e882b6cc	2c63b141	dcd54aea
e14dc057	a9e56bb1	c85f8cfe	ea64ed6b
d468c475	924c32ab	5c16c32c	258a057a

Intermediate results for the third round of F_R
After transposing the data

6a69ca46	15c2a35c	5458a2b6	ca1a1cac
a9e82cdc	5d8263d5	ceb6b14a	d3cc41ea
e1a9c8ea	4de55f64	c06b8ced	57b1fe6b
d4925c25	684c168a	c432c305	75ab2c7a

g-Function

6a69ca46	15c2a35c	5458a2b6	3fc54926
1451f722	5d8263d5	ceb6b14a	d3cc41ea
e1a9c8ea	ab67c339	c06b8ced	57b1fe6b
d4925c25	684c168a	849e4ff2	75ab2c7a

MDS, S-Box, Interleaving and XORing with round constants

b934cc54	eaee008b	ae3f9d65	512f9951
eb82592c	7fc9b245	cd2b6366	888f8a15
85b1d819	6774bf45	d7042570	534d6ba6
c3a8aa21	28edd2fd	2e6eec3d	229069ea

Intermediate results for the Fourth round of F_R
After transposing the data

b9eaae51	34ee3f2f	cc009d99	548b6551
eb7fcd88	82c92b8f	59b2638a	2c456615
8567d753	b174044d	d8bf256b	194570a6
c3282e22	a8ed6e90	aad2ec69	21fd3dea

g-Function

b9eaae51	34ee3f2f	cc009d99	7688a33b
a56a7bd9	82c92b8f	59b2638a	2c456615
8567d753	343d2fdc	d8bf256b	194570a6
c3282e22	a8ed6e90	839211d4	21fd3dea

MDS, S-Box, Interleaving and XORing with round constants

74e9678c	d8bcae09	91bbcbea	bba62376
6e4d02b5	b561e82b	3dec94c1	bdfae135
f53c8ff6	7cad353b	67b1bfb0	122259f7
e02ee0ba	3b8e2a82	71a4f208	bc29a41c

Hash:

4da61595	4d61ef60	23aeeae0	fbe775f6
71e9c531	79d419a7	1bbd0b3c	2dce9624

Test Vectors (Multiple Block Message)

Let the message, M, be the 448-bit, ASCII string "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq".

Thus, the final padded message consists of two blocks.

The initial hash value, $H^{(0)}$, is

$H_0^{(0)}$ = cfc09a62	$H_1^{(0)}$= 06d956d1	$H_2^{(0)}$ = bd126f78	$H_3^{(0)}$ = d7da6ca0
$H_4^{(0)}$ = b92c2d82	$H_5^{(0)}$= 96029774	$H_6^{(0)}$= 6b041292	$H_7^{(0)}$ = 34605d71
$H_8^{(0)}$ = 269b1c20	$H_9^{(0)}$ = dd6af9bd	$H_{10}^{(0)}$=d16e2adb	$H_{11}^{(0)}$ = 7409aae8
$H_{12}^{(0)}$ = 6779c7fb	$H_{13}^{(0)}$ = f8fddfcd	$H_{14}^{(0)}$=2826f702	$H_{15}^{(0)}$ = c6c49d5d

The words of the padded message block are assigned to X_0, $X_1,\ldots,X_{15}$. These variables are the input for the MAYHAM hash Algorithm and are shown below:

X_0 = 61626364	X_1 = 62636465	X_2 = 63646566	X_3 = 64656667
X_4 = 65666768	X_5 = 66676869	X_6 = 6768696a	X_7 = 68696a6b
X_8 = 696a6b6c	X_9 = 6a6b6c6d	X_{10}= 6b6c6d6e	X_{11} = 6c6d6e6f
X_{12} = 6d6e6f70	X_{13} = 6e6f7071	X_{14}= 80000000	X_{15}= 00000000

Message blocks XORED with the initial hash values to get the working variables A_0, $A_1,\ldots,A_{15}$.

aea2f906	64ba32b4	de760a1e	b3bf0ac7
dc4a4aea	f065ff1d	0c6c7bf8	5c09371a
4ff1774c	b70195d0	ba0247b5	1864c487
0a17a88b	9692afbc	a826f702	c6c49d5d

Data Mixing

55422793	21ca6b7c	8532f05e	121285bb
bde88008	370ee3fb	e3785aa7	e600746d
26c01439	2160c0b7	6f0361b1	bfcaab63
76a0d273	2d5db697	e1fc41ac	e1757fb5

Intermediate results for the first round of F_R
After transposing the data

55218512	42ca3212	276bf085	07140b70
125968f8	e80e7800	80e35a74	08fba76d
26216fbf	a86e7bca	14c061ab	39b7b163
762de1e1	a05dfc75	e776a32a	7397acb5

g-Function

55218512	42ca3212	276bf085	07140b70
125968f8	e80e7800	80e35a74	08fba76d
26216fbf	a86e7bca	14c061ab	39b7b163
762de1e1	a05dfc75	e776a32a	7397acb5

MDS, S-Box, Interleaving and XORing with round constants

db3ad11d	70aa6140	a310f0dc	6473bc71
ade12da8	ebc4ee94	c15b4a1d	b0bf3d2e
061f1fd0	d8565f45	11a36242	854299c9
255cbd1f	842c52bd	b4d18818	21993570

Intermediate results for the second round of F_R
After transposing the data

db70a364	3aaa1073	d161f0bc	1d40dc71
adebc1b0	e1c45bbf	2dee4a3d	a8941d2e
06d81185	1f56a342	1f5f6299	d04542c9
2584b421	5c2cd199	bd528835	1fbd1870

g-Function

db70a364	3aaa1073	d161f0bc	3cfdf4e1
895c6514	e1c45bbf	2dee4a3d	a8941d2e
06d81185	011aff01	1f5f6299	d04542c9
2584b421	5c2cd199	dcb1eace	1fbd1870

MDS, S-Box, Interleaving and XORing with round constants

701a61f9	3018c685	d9c1d18b	50f97fb1
1c46ce29	180abd26	d914a903	a276f851
aeb7cfb4	194f94ad	63b32624	d458bb82
c636637a	25d51d88	6b2bf391	0ba54c07

The words of the padded message block are assigned to X_0, $X_1,\ldots,X_{15}$. These variables are the input for the MAYHAM hash Algorithm and are shown below:

X_0 = 00000000	X_1 = 00000000	X_2 = 00000000	X_3 = 00000000
X_4 = 00000000	X_5 = 00000000	X_6 = 00000000	X_7 = 00000000
X_8 = 00000000	X_9 = 00000000	X_{10} = 00000000	X_{11} = 00000000
X_{12} = 00000000	X_{13} = 00000000	X_{14} = 00000000	X_{15} = 000001c0

Intermediate results for the third round of F_R
After transposing the data

7030d950	1a18c1f9	61c6d17f	f9858bb1
1c18d9a2	460a1476	cebda9f8	29260351
ae1963d4	b74fb358	cf9426bb	b4ad2482
c6256b0b	36d52ba5	631df34c	7a889107

Message blocks XORED with the initial hash values to get the working variables A_0 , $A_1,\ldots,A_{15}$.

603847f1	1e98746f	c59ca864	5afee300
a4956eee	5f777d4a	655e145f	cb03baa0
e70de886	671f0092	7b580061	582e0828
9de98151	b58352a1	684863e1	28b1c760

g-Function

7030d950	1a18c1f9	61c6d17f	740e1cb8
8c49b2f2	460a1476	cebda9f8	29260351
ae1963d4	fd59c7ce	cf9426bb	b4ad2482
c6256b0b	36d52ba5	32b21a07	7a889107

Data Mixing

4802eb51	61d2dc36	9ea83e35	d822fe68
235576c1	038f6e75	9f327d54	74ca1efa
85848607	7ef0fdbd	9903aaad	5a9465d3
962e8d30	1938f155	3822e8f9	e83e2d5f

MDS, S-Box, Interleaving and XORing with round constants

59aaf4c7	6276d043	df2a5902	728e475f
9e04bbbd	8facfd09	b861f5c8	ec9cf1a1
241387b2	cd639fd1	07522e9e	9e6c84df
ee8ad686	f0191f32	f4df8025	d1447066

Intermediate results for the first round of F_R
After transposing the data

48619ed8	02d2a822	ebdc3efe	51363568
23039f74	558f32ca	766e7d1e	c17554fa
857e995a	84f00394	86fdaa65	07bdadd3
961938e8	2e38223e	8df1e82d	3055f95f

Intermediate results for the Fourth round of F_R
After transposing the data

5962df72	aa762a8e	f4d05947	c743025f
9e8fb8ec	04ac619c	bbfdf5f1	bd09c8a1
24cd079e	1363526c	879f2e84	b2d19edf
eef0f4d1	8a19df44	d61f8070	86322566

g-Function

48619ed8	02d2a822	ebdc3efe	818c2ec7
6b653e4c	558f32ca	766e7d1e	c17554fa
857e995a	da7f365e	86fdaa65	07bdadd3
961938e8	2e38223e	14ef9292	3055f95f

g-Function

5962df72	aa762a8e	f4d05947	4d7527c5
f7f2985e	04ac619c	bbfdf5f1	bd09c8a1
24cd079e	180fb408	879f2e84	b2d19edf
eef0f4d1	8a19df44	5dbeaef4	86322566

MDS, S-Box, Interleaving and XORing with round constants

0d4e435f	63ee5bac	6317cf55	c78b1351
f8496906	d9f09262	e24271b8	342bdf28
c2f8f542	eb4a3774	b34dbc0d	ec24874e
2d86cdc1	b140f8b0	68db04c1	7791c76e

MDS, S-Box, Interleaving and XORing with round constants

603847f1	1e98746f	c59ca864	5afee300
a4956eee	5f777d4a	655e145f	cb03baa0
e70de886	671f0092	7b580061	582e0828
9de98151	b58352a1	684863e1	28b1c6a0

Intermediate results for the second round of F_R
After transposing the data

0d6363c7	4eee178b	435bcf13	5fac5551
f8d9e234	49f0422b	699271df	0662b828
c2ebb3ec	f84a4d24	f537bc87	42740d4e
2db16877	8640db91	cdf804c7	c1b0c16e

The hash value for the first message block, $H^{(1)}$

7ed0bc60	209b8b64	040cec38	3061ceff
4e2ce918	d3860889	536cd3f2	90fa2a81

g-Function

0d6363c7	4eee178b	435bcf13	215d16bf
063d45fb	49f0422b	699271df	0662b828
c2ebb3ec	423a8f4f	f537bc87	42740d4e
2db16877	8640db91	c32fc14e	c1b0c16e

The initial hash value, $H^{(0)}$, is

$H_0^{(0)}$ = 603847f1	$H_1^{(0)}$ =1e98746f	$H_2^{(0)}$ = c59ca864	$H_3^{(0)}$ =5afee300
$H_4^{(0)}$ =a4956eee	$H_5^{(0)}$ =5f777d4a	$H_6^{(0)}$ =655e145f	$H_7^{(0)}$ = cb03baa0
$H_8^{(0)}$ = e70de886	$H_9^{(0)}$ = 671f0092	$H_{10}^{(0)}$=7b580061	$H_{11}^{(0)}$ =582e0828
$H_{12}^{(0)}$ =9de98151	$H_{13}^{(0)}$=b58352a1	$H_{14}^{(0)}$=684863e1	$H_{15}^{(0)}$ = 28b1c6a0

MDS, S-Box, Interleaving and XORing with round constants

9f959de7	1307a3eb	78a5efbd	894975c2
1d1f59f2	8faa6f7c	2c61eae1	2efcf621
e63eddb6	c00814e4	67db7684	db54bf0d
a148544e	95a39f5f	7fc34c9a	05521140

Intermediate results for the third round of F_R
After transposing the data

9f137889	9507a549	9da3ef75	e7ebbdc2
1d8f2c2e	1faa61fc	596feaf6	f27ce121
e6c067db	3e08db54	dd1476bf	b6e4840d
a1957f05	48a3c352	549f4c11	4e5f9a40

g-Function

9f137889	9507a549	9da3ef75	364b5802
bca2a4b7	1faa61fc	596feaf6	f27ce121
e6c067db	5db33d50	dd1476bf	b6e4840d
a1957f05	48a3c352	31b3c2d0	4e5f9a40

MDS, S-Box, Interleaving and XORing with round constants

89f24f9e	1e827887	2d9d9588	c681b0e2
6437f0e2	7b6b63c7	a1665683	1a48860a
637d4578	add43197	693d8859	4e0d935c
4098b881	cea20fd2	a3014aca	ec8f26b2

Intermediate results for the Fourth round of F_R
After transposing the data

891e2dc6	f2829d81	4f7895b0	9e8788e2
647ba11a	376b6648	f0635686	e2c7830a
63ad694e	7dd43d0d	45318893	7897595c
40cea3ec	98a2018f	b80f4a26	81d2cab2

g-Function

891e2dc6	f2829d81	4f7895b0	205a5394
ed99cee0	376b6648	f0635686	e2c7830a
63ad694e	b53fa355	45318893	7897595c
40cea3ec	98a2018f	fd40d2b9	81d2cab2

MDS, S-Box, Interleaving and XORing with round constants

0e8668dd	fc46f41a	fc299ffa	07f4b3f6
2aeb5f9a	08b2b78a	8b2c1697	9c6927ec
c8188810	6577996c	149ff396	2c5d2b90
b3c3a0bf	aa7307f5	1f439362	80c4103b

The hash value for the second message block is the final hash value, $H^{(F)}$

899e1957	24b9df54	37ab035c	57f70d82
7bbd0a94	148327af	b1a37ca6	3101ce1e