International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 05, May-2015

Implementation of Sha-3 for Security and Error Detection and Correction Mechanism to Enhance Memory Reliabilty

Asha T R[1],
[1]Mtech, DE, Dept of ECE, SJBIT
Uttharahalli main road, Kengeri, Bangalore-60,
Karnataka, India

Hamsaveni N[2]
[2]Assoc Professor, Dept of ECE, SJBIT
Uttharahalli main road,
Kengeri, Bangalore-60,
Karnataka, India

***Abstract* - The secure hash algorithm (SHA)-3 that is keccak is selected in 2012 to provide security to any application that requires hashing, pseudo-random number generation, and integrity checking. Based on several benchmarks such as security, performance and complexity, this algorithm is selected. So in this paper SHA-3 algorithm is implemented to provide security to the message.Saved hash values in memory may suffer from multiple cell upsets(MCUs) when exposed to radiation environment. Several error correction codes(ECCs) are widely used to protect memory from causing data corruption but the problem was they were capable to detect and correct very few errors. In this paper, novel decimal matrix code (DMC) is used to improve the memory reliability. DMC is based on divide-symbol and it utilizes decimal algorithm to maximize error detection capability. To minimize area overhead error reuse technique (ERT) which uses encoder as part of decoder is utilized.**

Index Terms— Secure hash algorithm (SHA)-3, keccak, high performance, security, decimal algorithm, error correction codes (ECCs), memory, multiple cells upsets (MCUs) and error reuse technique(ERT).

INTRODUCTION

Widespread hash functionsbefore 2004 were MD4, MD5, RIPE-MD, RIPE-MD160, SHA0, SHA1 and SHA2. Several attacks on these hash functions such as collision attacks, second pre-image attacks and pre-image attacks broke them and gave rise to a need of new and highly secured hash function.[1]

In 2007 the National Institute of Standards and Technology (NIST) initiated a selection process to choose a new secure cryptographic hash algorithm, i.e., secure hash algorithm (SHA)-3 in order to increase security and performance of hash functions. After three successive rounds of assessments, on October 2, 2012, keccak was chosen as the winner algorithm for this standard[1].

Based on different aspects, among five finalists keccak was selected as the winner algorithm. Those different aspects includes software implementations [2], hardware application-specific integrated circuit (ASIC) implementations [3]–[5], and field programmable gate array(FPGA) implementations [5]–[7].

Keccak the winner of SHA-3 competition was expected to provide various security- constrained applications such as the ones used for generating digital signatures and message authentication codes. For example, it can be used in conjunction with the efficient implementations for the point multiplication in elliptic curve cryptography (ECC) or in integrity-checking for mobile *ad hoc* networks (MANETs) which lack physical layer security [8]. This secure hash algorithm was found to be utilized in integrity assurance of implantable and wearable medical devices, Internet of nano-things , and smart infrastructures, smart buildings, smart fabrics, networked control systems, and wireless sensor networks as well[9].

There are six functions in SHA-3 family. Among, four are cryptographic hash functions, called SHA3-224, SHA3-256, SHA3-384, and SHA3-512 and the rest two are extendable-output functions (XOFs), called SHAKE128 and SHAKE256. Extendable-output functions are different from hash functions. Among four cryptographic hash functions of SHA-3, SHA3-512 was found to be more secure with its higher output length of 512. [10]. In this paper to provide tight security to the message or the information, SHA3-512 (keccak f(1600)) is being used.

The soft error rate in memory cells is rapidly increasing due to ionizing effects of atmospheric neutron, alpha-particle, and cosmic rays [11] because of combination of memories with an increasing number of electronic systems as the technology is scaled down to Nano-scale. In order to have highly reliable memory, even though single bit upset is a major concern, in several applications [12], multiple cell upsets (MCUs) have become a serious reliability concern.

To protect memories against soft errors, several error correction codes (ECCs)[13]–[16] have been widely used for years in order to make memory cells as fault-tolerant as possible.]. For example, to deal with MCUs in memories, BCH codes [17], Reed–Solomon codes [18], and PDS codes [19] were used. But it is found that these codes require more area, power, and delay overheads because of more complexity in their encoding and decoding circuits. To assist with single-error correction and double-error detection, Built-in current sensors (BICS) were proposed in order to provide protection against MCUs [20], [21]. But they were successful to correct only two errors in a word.

More recently, in [22], to efficiently correct MCUs per word with a low decoding delay, 2-D matrix codes (MCs) are proposed in which one word is divided into multiple rows and columns logically. By adding parity code to each column, bits per row are protected by Hamming code. The vertical syndrome bits are activated when two errors are detected by Hamming. Then these two errors can be corrected. As a result, in all cases MC is capable of correcting only two errors. In [23], decimal algorithm is combined with Hamming code and applied at software level, wherein addition of integer values is used to detect and correct soft errors. It results in a lower delay overhead compared to other codes.

In this paper, to achieve enhanced memory reliability, novel decimal matrix code (DMC) based on divide-symbol is used. To detect errors it makes use of decimal algorithm (decimal integer addition and decimal integer subtraction). Using decimal algorithm, reliability of memory is enhanced by maximizing the error detection capability. Without disturbing the whole encoding and decoding processes, the encoder-reuse technique (ERT) is used in order to minimize area overhead of extra circuits (encoder and decoder). This is achieved by using DMC encoderitself be a part of the decoder.

PROPOSED METHODOLGY

A. Sha-3 (Keccak Algorithm)

Permutation f is the core of the Keccak algorithm. This is applied repeatedly to a fixed-length state of $b = r + c$ bits, where r and c are bit rate and capacity respectively. In which higher values of r represents higher speed whereas higher values of c corresponds to higher security level. The data or input information is first padded to get a length multiple of r. Both absorbing and squeezing phases are performed to obtain sufficient message length for all five steps and standard output length respectively [1].

Keccak family possess seven possible types among which keccak-f[r+c=1600] is chosen for the sake of brevity. In this 1600, is the width of permutation (in bits).For such a kind of keccak, the state consists of an array of 5×5 lanes, each of length $w = 64$ bits is considered. As per the standard, 24rounds [1] are recommended for keccak-f[1600].For each of 24rounds, there exists five internal steps shown in fig 1. For all these steps , $0 \le x, y \le 4$ and $0 \le z \le 63$.

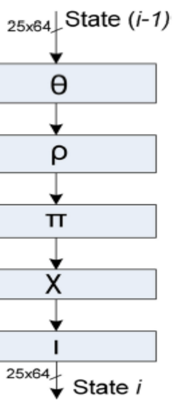

Figure1: Five internal steps involved in the Keccak algorithm.

The first step θ: It computes the parity of for length of 5w (320, when w = 64) at a time in a 5-bit column pattern using exclusive-or operation.Columns involved will be nearby in a regular pattern. Precisely, $a[x][y][z] \leftarrow a[x][y][z] \oplus \text{parity}(a[x][y-1][z]) \oplus \text{parity}(a[x][y+1][z-1])$.

Second step ρ: Here the bitwise rotation for each of the 25 words will be done. Precisely, a[0][0] is not rotated, and for all $0 \le t < 24$, $a[x][y][z] \leftarrow a[x][y][z-(t+1)(t+2)/2]$.

Third step π: A fixed pattern $a[y][2x+3y] \leftarrow a[x][y]$ will be permuted for every 25 words.

Fourth step χ: Combination along rows in a bitwise pattern will be done using $a \leftarrow a \oplus (\neg b \,\&\, c)$ where b and c are intermediate states. Precisely, $a[x][y][z] \leftarrow a[x][y][z] \oplus \neg a[x][y+1][z] \,\&\, a[x][y+2][z]$.

Final step ι: Here obtained values will be xored with RC $A[0, 0] \leftarrow A[0, 0] \oplus RC$, where RC is the round constant specific for each of the 24 rounds of Keccak-f[1600].

B. Error Detection And Correction Approach

1. *Schematic of Memory tolerating faults*

The schematic of memory, tolerating faults using DMC is depicted in Fig. 2. In the encoding (write) process, message bits D are fed to DMC encoder. Horizontal H and the vertical V redundancy bits are calculated for the message bits. After the completion of the encoding process, information bits and redundancy bits are stored in memory as shown in the figure 2. If multiple cells are upset, they are located and corrected in the decoding (read) process. Since the DMC uses decimal algorithm, fault tolerant capability is maximized with lower performance overhead and the ERT minimizes the total area coverage.

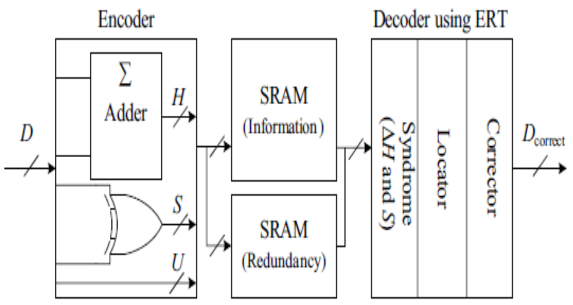

Figure 2: Schematic of memory tolerating faults using DMC.

2. *DMC Encoder*

The first operation that will be carried out in the encoder is the division of message into symbols and arrangement of them to a matrix form ie if N bit is the message, that will be divided into k symbols each of m bits ($N = k \times m$). Divided symbols are arranged in$k1 \times k2$ 2-D matrix form ($k = k1 \times k2$), where $k1$ represents number of rows and $k2$ number of columns respectively. The second operation performed in the encoder is calculation of horizontal (H) and vertical (V) redundancy bits. Horizontal redundancy bits are obtained by performing decimal integer addition on selected symbols for every row. Here, each symbol is treated as decimal integer. Vertical redundancy bits are calculating by performing binary xor operationto the bits in the same column. Since the symbol division and matrix formation is done logically but not physically, DMC enforce for the change of physical structure of the memory.

To explain the DMC scheme, 32-bits are taken as an example ie$D0$ to $D31$ are information bits. These 32-bits are divided into eight symbols each of 4-bits. $k1$ and $k2$ values are chosen as 2 and 4 respectively. $H0$–$H19$ are horizontal check or redundancy bits; $V0$ through $V15$ are vertical redundancy bits. However,maximum correction capability (i.e., the maximum size of MCUs can be corrected) and the usage of number of redundant bits varieswith the variation in the values of$k$ and m. So, adjusting the values of$k$ and m must be done carefully to maximize the correction capability and also to minimize the number of redundant bits.

For example, in the considered case of data bits 32, if$k = 2\times2$ and $m = 8$, only single error bit can be corrected and the usage of redundant bits will be 40. Whereas for$k = 4 \times 4$ and $m = 2$, 3-bits of error can be corrected with the reduced redundancy bits of 32. Instead, when $k = 2 \times 4$ and $m = 4$ is chosen, the correction capability can be maximized up to 16 bits with the redundancy of 36. In this paper, to prioritize the enhancement of the reliability of memory, the first consideration is taken onerror correction capability, so $k = 2 \times 4$ and $m = 4$ are utilized to construct DMC.

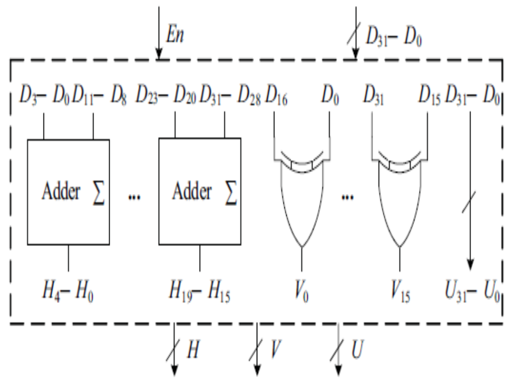

Figure 3:Structure of 32-bit DMC encoder that uses multibit adders and XOR gates

The encoder computing redundant bits using multibit adders and XOR gates is shown in Fig. 3. In this figure, $H19 - H0$ represents horizontal redundant bits, while $V15 - V0$ represents vertical redundant bits, and the remaining bits $U31 - U0$ are the information bits that are directly taken from $D31$ to $D0$.

3. *DMC Decoder*

The decoding process is required to correct the corrupted information. For an example, first the received horizontal redundant bits $H4H3H2H1H0'$ and vertical redundancy bits $V0' - V3'$ are calculated to theinformation bitsD' that are read to the decoder. Then, the horizontal syndrome bits $\Delta H4H3H2H1H0$ and the vertical syndrome bits $S3 - S0$ are calculated as follows:

$$\Delta H4H3H2H1H0 = H4H3H2H1H0' - H4H3H2H1H0$$

$$S0 = V0' \oplus V0$$

Where "–" represents decimal integer subtraction. In the same way complete horizontal and vertical syndrome bits are calculated for the remaining received message. If$\Delta H4H3H2H1H0$ and $S3 - S0$ are zero, the stored codeword has original information bits in symbol 0 ie no error has been occurred. Else if$\Delta H4H3H2H1H0$ and $S3 - S0$ are nonzero, presence of error in the corresponding symbol will be ensured. Then the induced errors in symbol 0 are detected and located, and finally errors can be corrected by

$$D0correct = D0 \oplus S0.$$

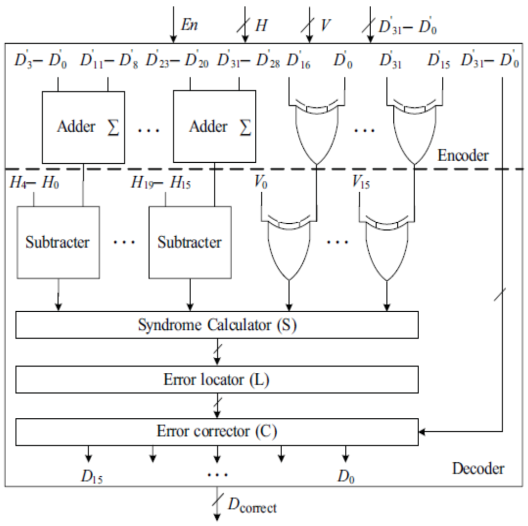

Fig 4:Structure of 32-bitDMC decoder using ERT.

The DMC decoder involving syndrome calculator, error locator, and error corrector as its sub-modules is represented in Fig. 4. Each sub-module performs a specific task in the decoding process.From the figure it can be observed that the redundancy bits must be recomputed from the received information bits *D'* then those redundancy bits are compared with the original redundancy bits in order to obtain syndrome ΔH and *S*. Non zero value of ΔHensures the presence of error in that corresponding symbol. This operation is done in the error locator. Then by performing xor operation to the located symbol with corresponding syndrome vertical bits, corrupted bits are corrected in the error corrector.

RESULTS

Using keccak algorithm that is SHA-3, security to the message is provided by generating hash values. 64bits taken as input information and hash size of 512 is generated as shown in figure 5. It is found that the recognition of input message from the hash value is highly impossible even after several random trials. This proves that the security provided to the information via SHA-3 is very high.

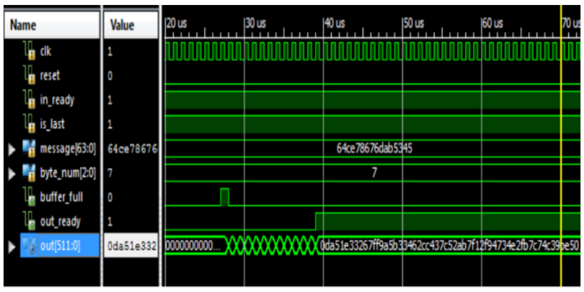
Name
Value
20 us
30 us
40 us
50 us
60 us
70 us
clk 1
reset 0
in_ready 1
is_last 1
message[63:0] 64ce78676 64ce78676dab5345
byte_num[2:0] 7 7
buffer_full 0
out_ready 1
out[511:0] 0da51e332 0000000000 0da51e33267ff9a5b33462cc437c52ab7f12f94734e2fb7c74c39e50

Fig 5: Hash output of size 512 for message signalof 64bits.

Error detection and correction approach that is implemented using decimal matrix code used decimal algorithm to efficiently detect and correct errors upto 16 bits in a data of 32bits as shown in figure 6. Number of redundancy bits used to locate and correct 16 bits of error are 36. Among, 20 were horizontal parity and 16 were vertical parity. Encoder reuse technique is used which reduced the area overhead by utilizing encoder as a part of decoder.

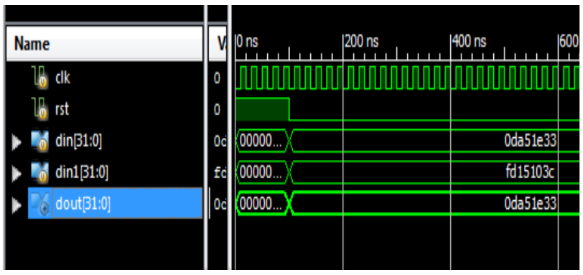
Name
0 ns
200 ns
400 ns
600
clk 0
rst 0
din[31:0] 0c 00000... 0da51e33
din1[31:0] fc 00000... fd15103c
dout[31:0] 0c 00000... 0da51e33

Fig 6: 16bits error correction in 32bit message.

CONCLUSION

Keccak was chosen as the best hashing algorithm, not just for its very strong overall security and performance, but because it offers exceptional performance in areas where other secure hash algorithms such as SHA=0 and SHA-1 are failed to perform as itcompletely relies on different architectural principles from those of SHA-0 and SHA-1 for its security. Keccak has a large security margin, good general performance, and a flexible design. And to overcome the MCUs in memory, novel per-word DMC is used. This protection code utilized decimal algorithm to detect errors, so that more errors were detected and corrected. The results obtained from the implementation have shown that the utilized scheme has a superior level of protection against large MCUs in memory.

The only drawback of the utilized DMC is that the number of redundant bits used is found to be larger in order to maintain higher reliability of memory. To overcome this, a reasonable combination of *k* and *m* must be chosen to maximize memory reliability. Therefore, the future work can be conducted for the reduction of the redundant bits and the maintenance of the reliability of the utilized technique.

REFERENCES

[1] *Keccak Hash Function*, NIST (National Institute of Standards and Technology), (2014, Mar.) [Online]. Available: http://csrc.nist.gov/groups/ ST/hash/sha-3

[2] D.-J. Bernstein and T. Lange. (2012). The new SHA-3 software shootout. *e-Print* [Online]. Available: http://eprint.iacr.org/2012/004.pdf

[3] S. Tillich, M. Feldhofer, M. Kirschbaum, T. Plos, J.-M. Schmidt, and A. Szekely, "Uniform evaluation of hardware implementations of the round-two SHA-3 candidates," in *Proc. Conf. SHA-3 Candidate*, pp. 1–16, 2010.

[4] X. Guo, S. Huang, L. Nazhandali, and P. Schaumont, "Fair and comprehensive performance evaluation of 14 second round SHA-3 ASIC implementations," in *Proc. Conf. SHA-3 Candidate*, pp. 1–13, Aug. 2010.

[5] M. Kneˆzevi´c *et al.*, "Fair and consistent hardware evaluation of fourteen round two SHA-3 candidates," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 20, no. 5, pp. 827–840, May 2012.

[6] E. Homsirikamol, M. Rogawski, and K. Gaj, "Throughput vs. area trade-offs in high-speed architectures of five round 3 SHA-3 candidates implemented using Xilinx and Altera FPGAs," in *Proc. Workshop Cryptograph. Hardw. Embedded Syst.*, 2011, pp. 491–506.

[7] K. Latif, M. Rao, A. Aziz, and A. Mahboob, "Efficient hardware implementations and hardware performance evaluation of SHA-3 finalists," in *Proc. Conf. SHA-3 Candidate*, pp. 1–14, Mar. 2012.

[8] E. M. Shakshuki, N. Kang, and T. R. Sheltami, "EAACK–A secure intrusion detection system for MANETs," *IEEE Trans. Ind. Electron.*, vol. 60, no. 3, pp. 1089–1098, Mar. 2013.

[9] M. Mozaffari-Kermani, M. Zhang, A. Raghunathan, and N. K. Jha, "Emerging frontiers in embedded security," in *Proc. Conf. VLSI Design*, Jan. 2013, pp. 203–208.

[10] Penny Pritzker, Patrick D. Gallagher andCharles H. Romine " Federal Information Processing Standards Publication 202 May 2014".

[11] D. Radaelli, H. Puchner, S. Wong, and S. Daniel, "Investigation of multi-bit upsets in a 150 nm technology SRAM device," *IEEE Trans.Nucl. Sci.*, vol. 52, no. 6, pp. 2433–2437, Dec. 2005.

[12] E. Ibe, H. Taniguchi, Y. Yahagi, K. Shimbo, and T. Toba, "Impact of scaling on neutron induced soft error in SRAMs from an 250 nm to a 22 nm design rule," *IEEE Trans. Electron Devices*, vol. 57, no. 7, pp. 1527–1538, Jul. 2010.

[13] C. Argyrides and D. K. Pradhan, "Improved decoding algorithm for high reliable reed muller coding," in *Proc. IEEE Int. Syst. On Chip Conf.*, Sep. 2007, pp. 95–98.

[14] A. Sanchez-Macian, P. Reviriego, and J. A. Maestro, "Hamming SEC-DAED and extended hamming SEC-DED-TAED codes through selective shortening and bit placement," *IEEE Trans. Device Mater. Rel.*, to be published.

[15] S. Liu, P. Reviriego, and J. A. Maestro, "Efficient majority logic fault detection with difference-set codes for memory applications," *IEEETrans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 148–156, Jan. 2012.

[16] M. Zhu, L. Y. Xiao, L. L. Song, Y. J. Zhang, and H. W. Luo, "New mix codes for multiple bit upsets mitigation in fault-secure memories," *Microelectron. J.*, vol. 42, no. 3, pp. 553–561, Mar. 2011.

[17] R. Naseer and J. Draper, "Parallel double error correcting code design to mitigate multi-bit upsets in SRAMs," in *Proc. 34th Eur. Solid-StateCircuits*, Sep. 2008, pp. 222–225.

[18] G. Neuberger, D. L. Kastensmidt, and R. Reis, "An automatic technique for optimizing Reed-Solomon codes to improve fault tolerance in memories," *IEEE Design Test Compute.*, vol. 22, no. 1, pp. 50–58, Jan.–Feb. 2005.

[19] P. Reviriego, M. Flanagan, and J. A. Maestro, "A (64,45) triple error correction code for memory applications," *IEEE Trans. Device Mater.Rel.*, vol. 12, no. 1, pp. 101–106, Mar. 2012.

[20] P. Reviriego and J. A. Maestro, "Efficient error detection codes for multiple-bit upset correction in SRAMs with BICS," *ACM Trans. DesignAutom. Electron. Syst.*, vol. 14, no. 1, pp. 18:1–18:10, Jan. 2009.

[21] C. Argyrides, R. Chipana, F. Vargas, and D. K. Pradhan, "Reliability analysis of H-tree random access memories implemented with built in current sensors and parity codes for multiple bit upset correction," *IEEE Trans. Rel.*, vol. 60, no. 3, pp. 528–537, Sep. 2011.

[22] C. Argyrides, D. K. Pradhan, and T. Kocak, "Matrix codes for reliable and cost efficient memory chips," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 19, no. 3, pp. 420–428, Mar. 2011.

[23] C. A. Argyrides, C. A. Lisboa, D. K. Pradhan, and L. Carro, "Single element correction in sorting algorithms with minimum delay overhead," in *Proc. IEEE Latin Amer. Test Workshop*, Mar. 2009, pp. 652–657.