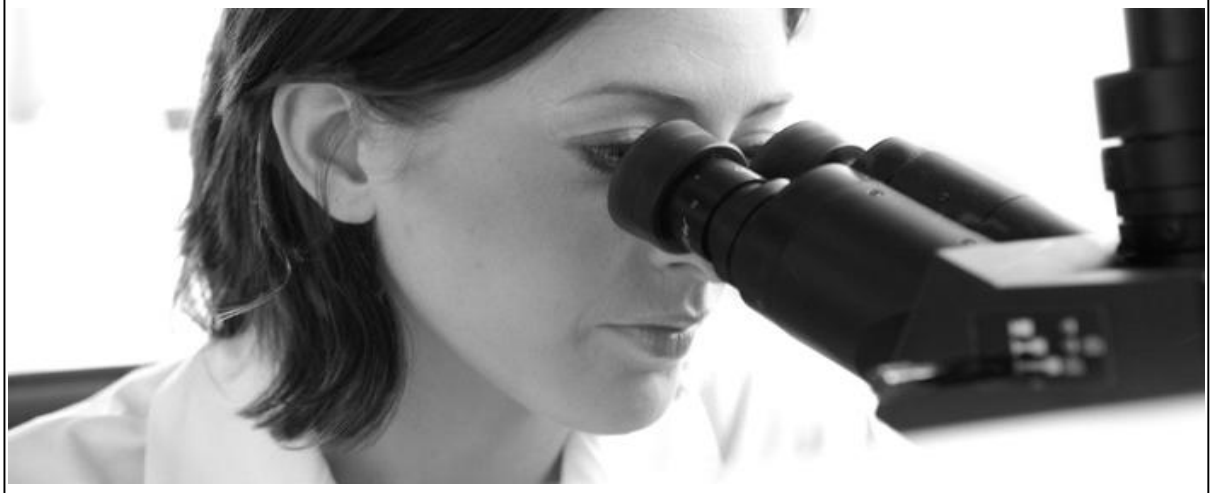
ISSN: 0976-3031

International Journal of Recent Scientific Research

Impact factor: 5.114

COMPACT IMPLEMENTATION OF SECURE CRYPTOGRAPHIC SHA-3 ALGORITHM



Christy Ann Luke

Volume: 6 Issue: 10

THE PUBLICATION OF
INTERNATIONAL JOURNAL OF RECENT SCIENTIFIC RESEARCH
http://www.recentscientific.com
E-mail: recentscientific@gmail.com

Available Online at http://www.recentscientific.com

ISSN: 0976-3031

International Journal of Recent Scientific Research
Vol. 6, Issue, 10, pp. 6749-6752, October, 2015

International Journal of Recent Scientific Research

RESEARCH ARTICLE

COMPACT IMPLEMENTATION OF SECURE CRYPTOGRAPHIC SHA-3 ALGORITHM

Christy Ann Luke

Department of ECE, Mangalam College of Engineering, Kerala, India

ARTICLE INFO

Article History:

Received 16th July, 2015
Received in revised form 24th August, 2015
Accepted 23rd September, 2015
Published online 28st October, 2015

Key words:

SHA-3, Cryptography, Security, Compact Implementation

ABSTRACT

Cryptographic hash functions have many security based applications in message authentication, digital signatures and data integrity. Secure Hash Algorithm-3(SHA-3) is a new cryptographic algorithm that was selected on Oct 2012 after five year public contest organized by National Institute of Standards and Technology (NIST), USA. This paper presents a compact and secured design of SHA-3 Algorithm on Xilinx Field Programmable Gate Array (FPGA) device Virtex-5. The design is logically optimized for area efficiency by merging Theta, Rho, and Pi steps of algorithm into single step. The security of the design is also increased by using a 64-bit LFSR. By logically merging these three steps, latency is reduced and maximum operating frequency of design is enhanced. Comparing the results with the previously reported FPGA implementations of SHA3-512, this design shows the best throughput per area (TPA) ratio of 47.

INTRODUCTION

There is no doubt about the fact that electronic communication has revolutionized this world. The world has progressed from communication with mainly letters written on paper and sent through the post office to instant communication via email, chat and social networking websites like Facebook and Google+. Many communication activities that were traditionally done via post are now done through electronic means. These activities include transferring documents, images, audio and video.

Communication needs to be secure to avoid fraudulent activities. Documents created by an institution such as transcripts can be digitally signed; images created by a camera can be digitally watermarked, all is an effort to ensure secure communication. These schemes, such as digital signatures and digital watermarking, utilize a number of cryptographic primitives. Cryptography is the art of secret writing. Cryptographic hash functions are primitives or building blocks utilized in the schemes that are used to provide information security. The cryptographic hash functions on their own do not typically provide full information security; however, they play a critical role in the schemes that do provide information security. Hence the security and speed of the cryptographic hash function can significantly impact the overall security and computational efficiency of an information security scheme.

Recent secure hash algorithms were found susceptible to attacks including MD5, SHA-0, SHA-1 and SHA-2. Secure hashing algorithms take a block of data (message), and return a fixed size bit string (hash value), such that any change on data leads to a change on the hash value (digest). This can be considered as a scenario that describes briefly the mechanism of the secure hashing algorithm. The long term security of these algorithms was uncertain, which led to requirement of new cryptographic hash function. Therefore National Institute of Standards and Technology (NIST) announced Keccak algorithm as new secure hash algorithm (SHA-3) in the year 2012 and announced as Federal Information Processing Standard Publication 202 in April, 2014[FIPS-202, 2014].

Secure Hash Algorithm -3 (Sha-3)

Keccak is recognized as a new Secure Hash Algorithm-3 i.e. SHA-3[FIPS-202, 2014] announced by NIST. The Keccak-f permutation is the basic component of Keccak Hash function and supports 224-bit, 256-bit, 384-bit and 512-bit hash variants. It consists of number of rounds (n_r=12+2l) and each round is the combination of logical operations and bit permutations. Keccak is generated from sponge function with Keccak [r, c] members. It is categorized by these additional functions i.e, bit rate (r) and capacity (c). The addition of r + c gives width of the Keccak function permutation and is it is further limited to values as indicated 25, 50, 100, 200, 400, 800, 1600. The value of r = 25*2^l .The Keccak team introduced

Corresponding author:* **Christy Ann Luke
Department of ECE, Mangalam College of Engineering, Kerala, India

the Keccak [1600] function for SHA-3 proposal with different values of 'r' and 'c. Keccak [1600] was selected because of its increased number of rounds in order to provide improved security margin. For 256-bit hash value r = 1088 and c = 512. For 512-bit hash output, the values of r and c are 576 and 1024 respectively [Alia Arshad, Dur-e-Shahwar kundi, Arshad Aziz, 2014].

Initially, the 1600-bit input of the compression function is stored in the 5x5 state matrix (A) and distributed into twenty five 64-bit words from A[0,0] to A[4,4] as shown below,

A[0][0]=l[63:0],A[0][1]=l[127:64],A[0][2]=l[191:128],……, A[4][4]=l[1599:1536]

Every single compression function of Keccak composed of 24 rounds and each round is sub-divided into five steps i.e. Theta (Θ), Rho (ρ) and Pi (π), Chi (χ), and Iota (i) as shown in eq.(1) to (6).

THETA (Θ) STEP

$$C[X] = A[X,0] \oplus A[X,1] \oplus A[X,2] \oplus A[X,3] \oplus A[X,4], \quad 0 \le X \le 4 \qquad (1)$$

$$D[X] = C[X-1] \oplus ROT(C[X+1, 1], \quad 0 \le X \le 4 \qquad (2)$$

$$A[X, Y] = A[X, Y] \oplus D[X], \quad 0 \le X, Y \le 4 \qquad (3)$$

Rho (P) And Pi (Π) Step

$$B[Y, 2X+3Y] = ROT(A[X,Y], r[X,Y]), \quad 0 \le X, Y \le 4 \qquad (4)$$

Chi (X) Step

$$A[X, Y] = B[X, Y] \oplus ((NOT\ B[X+1,Y])\ AND\ B[X+2,Y]), \quad 0 \le X, Y \le 4 \qquad (5)$$

Iota (I) Step

$$A[0, 0] = A[0, 0] \oplus RC \qquad (6)$$

The above five steps of compression function are the core of SHA-3. In the above equations all operations within indices are performed modulo 5. The complete permutation state array is denoted by 5x5 state matrix (A) and A[x, y] denotes a particular 64-bit word in that state. B[x, y], C[x] and D[x] are intermediate variables. Other operations include bitwise XOR, NOT and AND logical operations. In eq. (2) and (4) ROT is used to represent a bit-rotation operation. The constant r[x, y] provides the rotation bit scheme for the updated bits of A[x, y], and the RC is a 64-bit word that is unique for each round of the compression function.

The SHA-3 hash function operation consists of three phases that include: initialization, absorbing and squeezing. Initialization is simply the initialization of state matrix (A) with all zeros. In the absorbing phase each r-bit (bitrate) wide block of the message is XORed with the current matrix state and 24 rounds of the SHA-3 compression function are performed. After absorbing all blocks of the input message, there comes the squeezing phase. In this phase the resultant state matrix of the absorbing phase is simply truncated to the desired length of the output hash [Alia Arshad, Dur-e-Shahwar kundi, Arshad Aziz, 2014].

Table1 Cyclic Shift Offset for KECCAK

	X=3	**X=4**	**X=0**	**X=1**	**X=2**
Y=2	25	39	3	10	43
Y=1	55	20	36	44	6
Y=0	28	27	0	1	62
Y=4	56	14	18	2	61
Y=3	21	8	41	45	15

Table.2 RC values in Hexadecimal

RC[0] = 0x0000000000000001
RC[1] = 0x0000000000008082
RC[2] = 0x800000000000808A
RC[3] = 0x8000000080008000
RC[4] = 0x000000000000808B
RC[5] = 0x0000000080000001
RC[6] = 0x8000000080008081
RC[7] = 0x8000000000008009
RC[8] = 0x000000000000008A
RC[9] = 0x0000000000000088
RC[10] = 0x0000000080008009
RC[11] = 0x000000008000000A
RC[12] = 0x000000008000808B
RC[13] = 0x800000000000008B
RC[14] = 0x8000000000008089
RC[15] = 0x8000000000008003
RC[16] = 0x8000000000008002
RC[17] = 0x8000000000000080
RC[18] = 0x000000000000800A
RC[19] = 0x800000008000000A
RC[20] = 0x8000000080008081
RC[21] = 0x8000000000008080
RC[22] = 0x0000000080000001
RC[23] = 0x8000000080008008

Architecture of Sha -3

An iterative design of SHA-3 512-bit for compact implementation [Alia Arshad, Dur-e-Shahwar kundi, Arshad Aziz, 2014] is shown below. The architecture has 128-bit input data just to save extra input bits. The next block in proposed design is padder block which pads the required number of zeros with the input data in order to form 1600-bit state and then inversion is applied on each byte. The output from the padder block is forwarded to 2 x 1.Multiplexer (MUX) which drives the output data from padder to the compression-box of the architecture and selects the input data for the first round and feed-back data for other twenty three rounds of Keccak with the help of controlling signal (Ctrl 1).

When Ctrl 1 is low, MUX select the input data and at high, MUX will select the feedback data. First padded message is directly copied to Reg A which previously initialized with all zeroes and resulting bits are forward to Compression-Box (C Box).

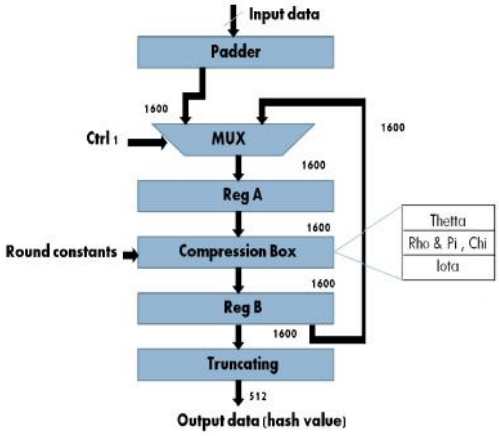


Figure 1 Existing 128 bit Keccak sequential architecture

It is basically the implementation of compression function in SHA-3 algorithm which comprises of theta (Θ),

rho (ρ), pi (π), chi (χ) and iota (i) step. For performance, the design is optimized by implementing rho (ρ), pi (π) and chi (χ) steps as a single step. After completing 24 iterations, final output is forwarded to Reg B for storage in order to synchronize the data-path. The last component in the architecture is Truncating component where inversion per byte is performed on the output bits and then truncated to the desired length of hash output.

Proposed Implementation of Sha -3

The Proposed architecture of SHA-3 Algorithm involves two phases.

1. Logically combining theta (Θ), rho (ρ) and pi (π) steps of SHA-3 compression function.
2. Increasing the security of SHA-3 algorithm by using Linear Feedback Shift Register.

Logically combining theta (Θ), rho (ρ) and pi (π) steps

In the existing method, Keccak algorithm consists of five steps. In these five steps, three steps, i.e., rho (ρ) and pi (π) steps are merged together and given to chi (γ) step to form a single step. In theta (Θ) step, two intermediate matrixes' like C[x] and D[x] are used. These intermediate matrixes's can be avoided and can form a direct equation for matrix A in Theta step. This equation of matrix A can be directly given to rho and pi step and these three steps can be written in single equation.

To combine the intermediate matrixes, C[x] is taken (i.e, eq.(1)). C[x-1] and C[x+1] is found using eq. (1). D[x] is obtained by combining eq. (7) and eq. (8)

$$C[X] = A[X,0] \oplus A[X,1] \oplus A[X,2] \oplus A[X,3] \oplus A[X,4] \quad \textbf{(1)}$$

$$C[X-1] = A[X-1,0] \oplus A[X-1,1] \oplus A[X-1,2] \oplus A[X-1,3] \oplus A[X-1,4] \quad \textbf{(7)}$$

$$C[X+1] = A[X+1,0] \oplus A[X+1,1] \oplus A[X+1,2] \oplus A[X+1,3] \oplus A[X+1,4] \quad \textbf{(8)}$$

$$D[X] = C[X-1] \oplus ROT(C[X+1,1]) \quad \textbf{(2)}$$

$$D[X] = (A[X-1,0] \oplus A[X-1,1] \oplus A[X-1,2] \oplus A[X-1,3] \oplus A[X-1,4]) \oplus (ROT\ (A[(X+1,0),1]) \oplus ROT\ (A[(X+1,1),1]) \oplus ROT\ (A[(X+1,2),1]) \oplus ROT\ (A[(X+1,3),1]) \oplus ROT\ (A[(X+1,4),1])) \quad \textbf{(9)}$$

Eq.(9) can be applied in eq.(3) to obtain A[x,y] and matrix C[x] and D[x] can be avoided

$$A[X, Y] = A[X, Y] \oplus D[X] \quad \textbf{(3)}$$

$$A[X, Y] = \{A[X, Y]\} \oplus \{(A[X-1,0] \oplus A[X-1,1] \oplus A[X-1,2] \oplus A[X-1,3] \oplus A[X-1,4]) \oplus (ROT\ (A[(X+1,0),1]) \oplus ROT\ (A[(X+1,1),1]) \oplus ROT\ (A[(X+1,2),1]) \oplus ROT\ (A[(X+1,3),1]) \oplus ROT(A[(X+1,4),1])) \quad \textbf{(10)}$$

Let's assume, m=x-1 and n=x+1

$$A[X, Y] = \{A[X, Y]\} \oplus \{(A[m,0] \oplus A[m,1] \oplus A[m,2] \oplus A[m,3] \oplus A[m,4]) \oplus (ROT\ (A[(n,0),1]) \oplus ROT\ (A[(n,1),1]) \oplus ROT\ (A[(n,2),1]) \oplus ROT\ (A[(n,3),1]) \oplus ROT\ (A[(n,4),1]))\} \quad \textbf{(11)}$$

A[x,y] obtained from theta step can be applied to rho and pi step to obtain B[x,y]

$$B[Y, 2X+3Y] = ROT\ (A[X,Y], r[X,Y]) \quad \textbf{(4)}$$

$$B[Y, 2X+3Y] = \{ROT\ (A[X,Y], r[X,Y])\} \oplus \{ROT\ (A[m,0], r[X,Y]) \oplus ROT\ (A[m,1], r[X,Y]) \oplus ROT\ (A[m,2], r[X,Y]) \oplus ROT\ (A[m,3], r[X,Y]) \oplus ROT\ (A[m,4], r[X,Y])\} \oplus (ROT\ (ROT\ (A[(n,0),1])) \oplus ROT\ (ROT\ (A[(n,1),1])) \oplus ROT\ (ROT\ (A[(n,2),1])) \oplus ROT\ (ROT\ (A[(n,3),1])) \oplus ROT\ ROT\ (A[(n,4),1]))\} \quad \textbf{(12)}$$

As the theta, rho and pi steps are merged to form B[x,y] by avoiding the intermediate matrixes. Thus area can be reduced and the speed of the system can be increased.

Linear Feedback Shift Register (LFSR)

In the existing method, the third step of the compression box, i.e., iota step, consist of many round constants. As these values are constant in all operations, to increase the security instead of using these values, a concept of password can be introduced. This can be done by using 64-bit LFSR.

A password which can be a four digit or five digit secret numbers can be used while transferring messages from one end to another. This four or five digit password can be converted to 64-bit data and is given to 64-bit LFSR. The value generated from the LFSR can be used instead of round constants in iota step for each round. Thus the hash value of the message can be obtained.

A linear feedback shift register (LFSR) is the heart of any digital system that relies on pseudorandom bit sequences (PRBS), with applications ranging from cryptography and bit-error-rate measurements, to wireless communication systems employing spread spectrum or CDMA techniques. The most

commonly used linear function of single bits is exclusive-or (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value. The maximum-length of an LFSR sequence is $2^n - 1$, where n is the degree of polynomial.
The polynomial equation for 64-bit LFSR is,

$$x^{64} + x^{63} + x^{61} + x^{60} + 1 \quad (13)$$

By using LFSR the security of the performance of the system can be increased.

Modified 128 bit Keccak sequential architecture

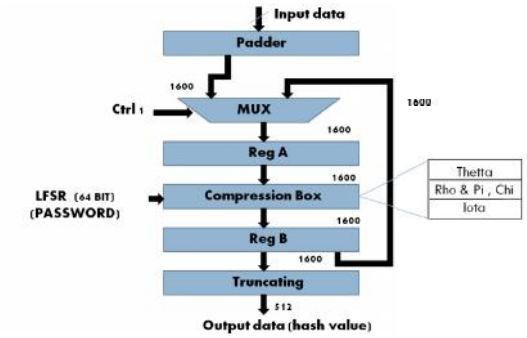


Figure 2 Modified 128 bit Keccak sequential architecture

IMPLEMENTATION RESULTS AND COMPARISON

The designs has been implemented and verified on Xilinx ISI Design Suite, System Edition 14.2 tool. The targeted device for the implementation was a Xilinx Virtex 5 (xc5vlx330t-2ff1738). The technique that was propose is unique and provides a throughput of 3.32 Gbps and TPA of 46.29 on Virtex-5 FPGA as compare to previously published results as given in table 3. The implementation aim was to get maximum throughput by tightening the timing constraints. The throughput (TP) of the given design can be calculated by eq. (14).

$$\text{Throughput} = \frac{Block\ size}{Latency} \times \text{Frequency} \quad (14)$$

where the block size of the message in bits is given by Block Size i.e. 576 for 512-bit variant and latency is the number of clock cycles required for a valid hash output.

Table 3 Obtained results

Implementation	Platform	Area	Frequency (mhz)	Throughput (gbps)	Tpa
This work	Virtex 5	81	508.182	3.75	46.29
[1]	Virtex 5	240	301.02	7.2	30.1
[7]	Virtex 6	188	285	0.08	0.425
[8]	Virtex 4	2024	143	6.07	2.99
[4]	Virtex 5	1229	238.4	1.08	0.879

CONCLUSION

Secure Hash Algorithm-3 (SHA-3) is a new cryptographic algorithm organized by National Institute of Standards and Technology (NIST). In this paper, a compact and secured design of SHA-3 Algorithm on Xilinx Field Programmable Gate Array (FPGA) device Virtex-5 is presented. The design is logically optimized for area efficiency by merging Theta, Rho, and Pi steps of algorithm into single step. The security of the design is also increased by using a 64-bit LFSR instead of round constants in the design. By logically merging these three steps, latency is reduced and maximum operating frequency of design is improved. Comparing the results with the previously reported FPGA implementations of SHA3-512, this design shows the best throughput per area (TPA) ratio of 46. This implementation can be used in giga-bit communication networks due to its improved throughput.

References

1. Alia Arshad, Dur-e-Shahwar kundi, Arshad Aziz, "Compact Implementation of SHA3-512 on FPGA" *Conference on Information Assurance and Cyber Security (CIACS), 2014*.
2. X. Wang,, D. Feng, X. Lai, and H. Yu, "Collisions for hash functions md4, md5, haval-128 and ripemd," *IACR, August 2004.*
3. Schneier, Bruce, "Cryptanalysis of MD5 and SHA: Time for a New Standard". *Computerworld, Retrieved 15 October 2014.*
4. K. Gaj, E. Homsirikamol, and M. Rogawski, "Comprehensive comparison of hardware performance of fourteen round 2 sha-3 candidates with 512-bit outputs using field programmable gate arrays," *2nd SHA-3 Candidate Conference, pp 23-24, August 2010.*
5. B. Baldwin, N. Hanley, M. Hamilton, L. Lu, A. Byrne, M. O Neill, and W.P. Marnane, "FPGA implementations of the round two sha-3 candidates," *The second SHA-3 Candidate Conference, 2010.*
6. FIPS-202, "Federal information processing standards publication fips-202, secure hash algorithm-3 (sha-3)," 2014.
7. S. Kerckhof, F. Durvaux, N.V. Charvillon, F. Regazzoni, G.M. de Dormale, and F.X. Standaert, "Compact fpga implementations of the five sha-3 finalists," *Springer Berlin Heidelberg, vol. 7079, pp. 217–233, 2011.*
8. A. Akin, A. Aysu, O.C. Ulusel, E. Savas, "Efficient hardware implementations of high throughput sha-3 candidates keccak, luffa and blue mid night wish for single- and multi-message hashing," *ACM, pp. 168–177, 2010.*

****

How to cite this article:

Christy Ann Luke.2015, Compact Implementation of Secure Cryptographic Sha-3 Algorithm. *Int J Recent Sci Res.* 6(10), pp. 6749-6752.

International Journal of Recent Scientific Research

ISSN 0976-3031