ISSN: 2456-043X

International Journal of Engineering Science and Generic Research (IJESAR)
Available Online at www.ijesar.in
Volume 2; Issue 5; September-October; 2016; Page No. 114-119

Implementation of SHA-2(256) & SHA-3(512) Algorithms for Information Security

M.Mounika [1], T.Thammi Reddy [2]

[1] PG Scholar, Department of ECE, G. Pulla Reddy Engineering College(Autonomous), Kurnool, AP, India

Email:madapuri18@gmail.com.

[2] Assoc Prof, Department of ECE, G. Pulla Reddy Engineering College(Autonomous), Kurnool, AP, India

Email:thammireddy@gmail.com

Abstract

The speedy tendencies in the wireless communications and personal communication methods has triggered the generation of new cryptographic algorithms. SHA-2 hash family is a new standard in the hash function category.SHA-256 is the one of hash functions of SHA-2 Family. Though SHA-2(256) is a secured Hash Function but the fear of collision attacks possible in SHA-2(256) made NIST announce a new hash function called SHA-3 or Keccak hash function.SHA-512 is one of the hash functions of the SHA-3 Family.

In this paper, both SHA-2(256) and SHA-3(512) are implemented in VHDL. The simulation and synthesis are performed in Xilinx ISE 13.2.The area consumed by both the hash algorithms is compared. Message Preparation, Message Expansion, Round logic and Round Addition are major steps performed in SHA-2(256). The basic SHA-3 presented here converts 128 bits input into 512 bits within the intermediate stage using one C-box. The main steps performed by C-Box are Theta, Rho, Pi, Chi and Iota. SHA-3(512) is more secure hash function but it utilizes more area compared to SHA-2(256).

Keywords: Cryptography, Hash Function, Theta, Rho, Pi, Chi, Iota.

Introduction

In the last years, communications growth has increased dramatically the amount of the transmitted data. In addition, to the raised quantity of information is the increased quality demand for the protection of the transmission channel with high level security strength[1]. In order, these special needs for security to be satisfied sufficiently, new cryptographic algorithms and security schemes have been developed. Lately, a new family of secure hash functions SHA-2 [2] have been published.

A hash function is a computationally efficient function, which maps binary strings of arbitrary length to binary strings of some fixed length, called hash-values. The main scope of the hash function is to ensure the data integrity in the transmission channel. Hash functions are also used for the implementation of digital signature algorithms [4, 5], keyed-hash message authentication codes [6].The Secure Hash Algorithm-1 (SHA-1) [7], is the world's most popular hash function. Unfortunately, the security level of this standard is limited and there were attacks reported on SHA-1.So NIST have announced new hash function SHA-2 and later SHA-3.
The remaining paper is organized as follows. Section II overviews the SHA-2(256) algorithm hash function. Section III describes the SHA-2(256) Implementation.
Section IV gives brief overview on SHA-3.Section V describes implementation of SHA-3.In section VI we give implementation results of SHA-2(256) and SHA-3(512).At the end, we provide conclusions related to this work.

II. SHA-2(256) ALGORITHM

The SHA-2 family was published in 2002 by the National Institute of Standards and Technology (NIST).SHA-256 is an algorithm specified in the SHA-2 family, It computes the digest of an arbitrary length message in the following way. The input message m is padded with one '1' and leading '0's until the message length (in bits) becomes a multiple of 512. The last 64 bits in the padded message are used to store the length of the original message as a 64-bit number. After the padding, the resulting message is divided into blocks of length 512-bits $D^{(1)}, D^{(2)}, \ldots\ldots, D^{(N)}$. Each block of data $D^{(i)}$ is processed sequentially by a main function during 64 rounds. A partial 256-bit hash value Hi is obtained as the current $D^{(i)}$ data block is totally processed. After computing the last data block $D^{(N)}$, the final hash HN is computed and delivered. Algorithm 1 lists the general functioning of SHA-2(256) algorithm.

It uses six logical functions, where each function operates on 32-bit words, which are represented as x, y, and z. The result of each function gives a new 32-bit word.

$$Ch(x, y, z) = (x \wedge y) \oplus (x \wedge z) \qquad (2.1)$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (2.2)$$

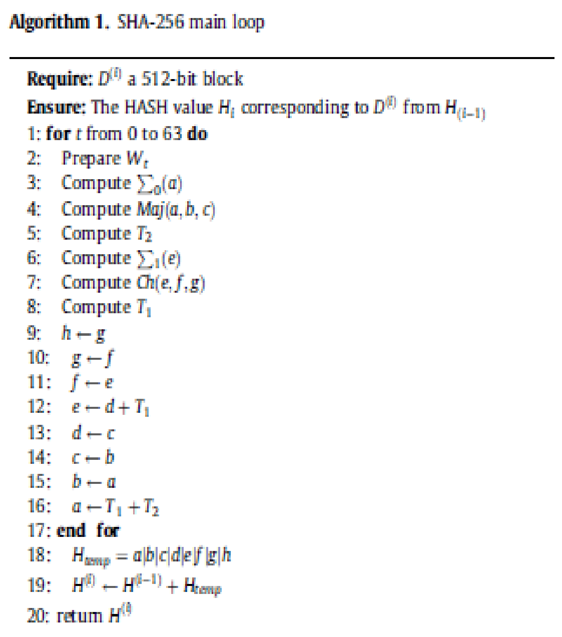
$$\sum\nolimits_{0}^{256}(x) = ROTR^{2}(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \qquad (2.3)$$

$$\sum_{1}^{256}(x) = ROTR^{6}(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \qquad (2.4)$$

$$\sigma_{0}^{256} = ROTR^{7}(x) \oplus ROTR^{18}(x) \oplus SHR^{3}(x) \qquad (2.5)$$

$$\sigma_{1}^{256}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \qquad (2.6)$$

Algorithm 1 lists the general functioning of SHA-2(256).



Algorithm 1. SHA-256 main loop

```
Require: $D^{(i)}$ a 512-bit block
Ensure: The HASH value $H_i$ corresponding to $D^{(i)}$ from $H_{(i-1)}$
1: for $t$ from 0 to 63 do
2:    Prepare $W_t$
3:    Compute $\sum_0(a)$
4:    Compute $Maj(a,b,c)$
5:    Compute $T_2$
6:    Compute $\sum_1(e)$
7:    Compute $Ch(e,f,g)$
8:    Compute $T_1$
9:    $h \leftarrow g$
10:   $g \leftarrow f$
11:   $f \leftarrow e$
12:   $e \leftarrow d + T_1$
13:   $d \leftarrow c$
14:   $c \leftarrow b$
15:   $b \leftarrow a$
16:   $a \leftarrow T_1 + T_2$
17: end for
18:   $H_{temp} = a|b|c|d|e|f|g|h$
19:   $H^{(i)} \leftarrow H^{(i-1)} + H_{temp}$
20: return $H^{(i)}$
```

III. IMPLEMENTATION OF SHA-2(256)

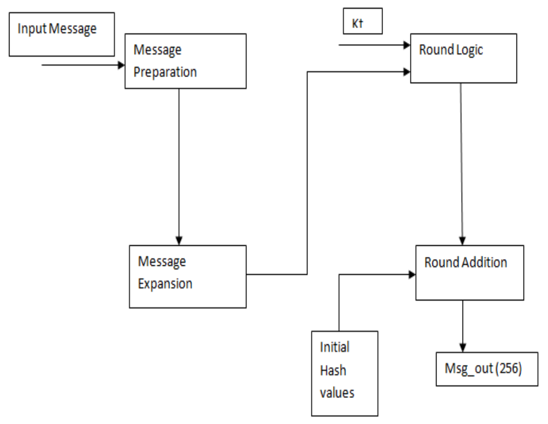


Fig: 1 SHA-2(256) block diagram

A. MESSAGE PREPARATION

- It can prepare the variable length message to 512 bits by adding extra padding bits.
- In message Preparation contain mainly three blocks as shown in the figure 2.

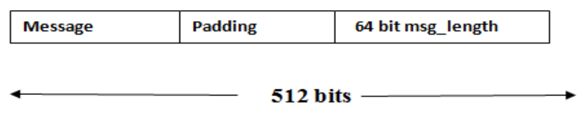


Fig: 2 Message Preparation

B. MESSAGE EXPANSION

Parsing the Padded Message

After a message has been padded, it must be parsed into *N m*-bit blocks before the hash computation can begin.

For SHA-2(256), the padded message is parsed into *N* 512-bit blocks, *M*(1), *M*(2),…, *M*(*N*). Since the 512 bits of the input block may be expressed as sixteen 32-bit words, the first 32 bits words W0, W1, W2, W3…….W15.

From 16 to 63 steps we calculate the Wt values using below equation

For i = 16 to 63

$$S_0 = ROTR^{7}(W_{i-15}) \oplus ROTR^{18}(W_{i-15}) \oplus SHR^{3}(W_{i-15})$$

$$S_1 = ROTR^{7}(W_{i-2}) \oplus ROTR^{19}(W_{i-2}) \oplus SHR^{10}(W_{i-2})$$

$$W_i = W_{i-16} + S_0 + W_{i-7} + S_1$$

Initialize the hash value, A = H0, B = H1, C = H2, D= H3, E = H4, F = H5, G = H6,

H = H7.

C. ROUND LOGIC

By using below equation we can perform round logic operation.
For i = 0 to 63

$$S_0 = ROTR^2(A) \oplus ROTR^{13}(A) \oplus ROTR^{22}(A)$$
$$maj = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$

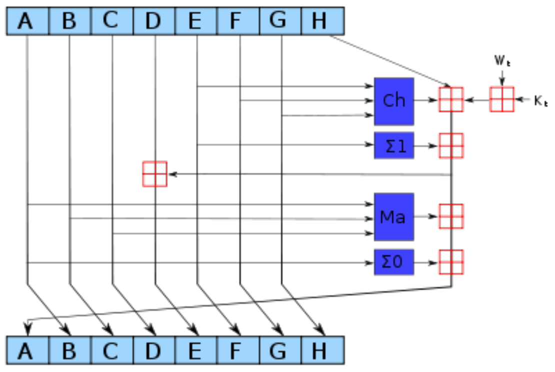


Fig 3 Block Diagram of Round Logic

The red ⊞ is an addition modulo 2^{32}.

$$S_1 = ROTR^6(E) \oplus ROTR^{11}(E) \oplus ROTR^{25}(E)$$
$$Ch = (E \wedge F) \oplus (\neg E \wedge G)$$
$$T_1 = H + S_1 + Ch + K_t + W_t$$
$$T_2 = S_0 + maj$$
$$H = G, G = F, F = E, E = D + T_1, D = C, C = B, B = A,$$
$$A = T_1 + T_2$$

IV. KECCAK (SHA-3)

Recent secure hash algorithms were found Susceptible to attacks including MD5, RIPEMD, SHA-0, SHA-1 and SHA-2 [8]. 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 [9] and announced as Federal Information Processing Standard Publication (FIPS PUB) 202 in April, 2014 [10].

Keccak is recognized as a new Secure Hash Algorithm-3 i.e. SHA-3 announced by NIST. Gilles Van Assche, Guido Bertoni, Michael Peeters and Joan Daemen designed and proposed the construction of Keccak Hash function. 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. Keccak is generated from sponge function with Keccak [r, c] members. It is categorized by these additional functions i.e bitrate (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 Keccak team introduced the Keccak [1600] function for SHA3 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. The 1600-bit state matrix of Keccak composed of 5x5 matrixes of 64-bit words. Initially, the message block should undergo the inversion procedure so that last byte should come first and first byte should become last. Every single compression function of Keccak composed of 24 rounds and each round is sub-divided into five steps.

- Theta (Θ),
- Rho (ρ)
- Pi (π),
- Chi (χ),
- Iota (i).

V.IMPLEMENTATION OF SHA-3(512)

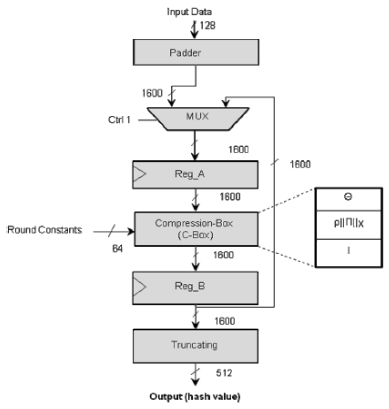


Fig: 4 128-bit Keccak Architecture

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

feedback 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 (CBox).It is basically the implementation of compression function in SHA-3 algorithm which comprises of thetha (Θ), rho (ρ), pi (π), chi (χ) and iota (i) step. For performance, we logically optimized our design 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.

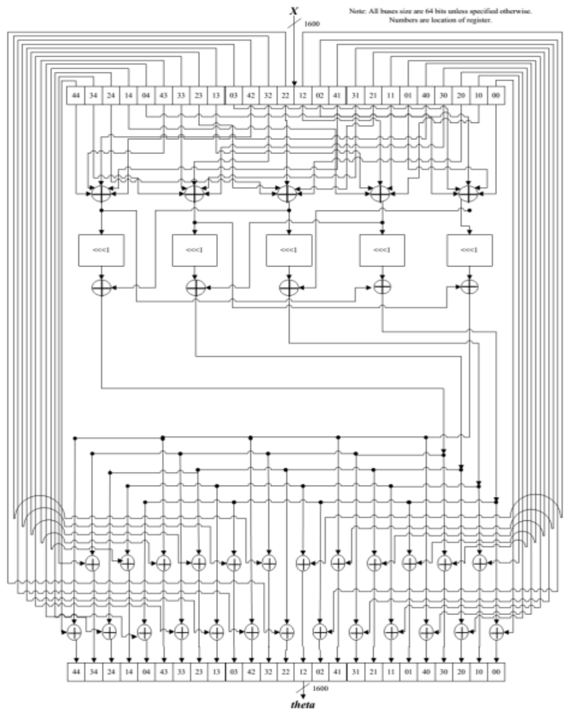
Implementation of C-Box

The details of the implementation for each step are given as follows.

A) Theta Step (Θ):

Theta step consists of three main steps in terms of equations that mainly require bitwise XOR operation. Equation (1) involves bitwise XOR operation between the 64-bit lanes of each row where every lane of each row is independent of each other so parallel operations can be applied on these lanes. We have used conventional 64-bit XOR operator in parallel to perform XORing between the five lanes in each row of the state array 'A' and results are stored in intermediate registers. Second step (2) of theta involves one bit left circular rotation which is accompanied by simple rewiring or replacing the bit pattern of each row, then XORed with the previous output lanes. The results are stored in an intermediate registers in the form of five lanes. These lanes are again XORed with input state matrix A[x, y] to form new 5 x 5 state matrix A'[x,y]. All the operations are done on modulo 5.

$$C[X] = A[X,0] \oplus A[X,1] \oplus A[X,2] \oplus A[X,3] \oplus A[X,4]$$
$$0 \le X \le 4 \qquad (1)$$
$$D[X] = C[X-1] \oplus ROT(C[X+1,1]$$
$$0 \le X \le 4 \qquad (2)$$
$$A[X,Y] = A[X,Y] \oplus D[X]$$
$$0 \le X,Y \le 4 \qquad (3)$$



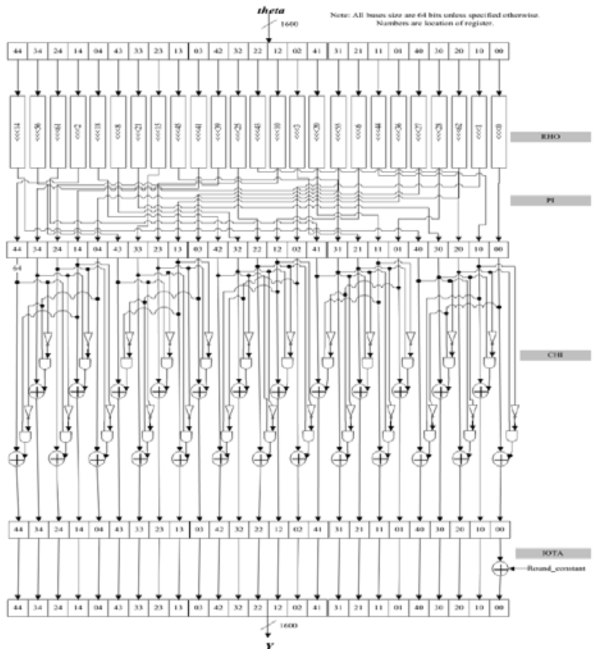
X
1600
Note: All buses size are 64 bits unless specified otherwise.
Numbers are location of register.
44 34 24 14 04 43 33 23 13 03 42 32 22 12 02 41 31 21 11 01 40 30 20 10 00
<<<1
theta

Fig: 5 Theta step

B) Integrated Rho (ρ), Pi (π) and Chi (χ) Step:

The two steps Rho (ρ) and Pi (π) can be expressed jointly by (4) that compute an auxiliary 5 x 5 array B from the state array 'A'.

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



theta
1600
Note: All buses size are 64 bits unless specified otherwise.
Numbers are location of register.
RHO
PI
CHI
IOTA
Round_constant
1600

Fig 6 Integrated (ρ), (π), (χ)

The rho and pi are basically permutations and each lane require cyclic shift by some fixed numbers according to the value of cyclic shift offset given in SHA-3 FIPS PUB 202.

Table 1: The Cyclic Shift Offsets "R(X,Y)" 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	10	18	2	61
Y=3	21	8	41	45	15

Following equation illustrates the function Chi (χ).

$$A[X,Y] = B[X,Y] \oplus ((NOT\,B[X+1,Y])\,AND\,B[X+2,Y])$$

$$0 \le X, Y \le 4 \qquad (5)$$

we have logically optimized the design by merging the rho and pi step into the chi step. We have performed all the calculations required in rho and pi step manually and applied cyclic shift on each lane of state matrix obtained at the output of theta step and relocated them after calculating its new position according to (4).

C) Iota (i):

The Iota step is the simplest step of Keccak algorithm.

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

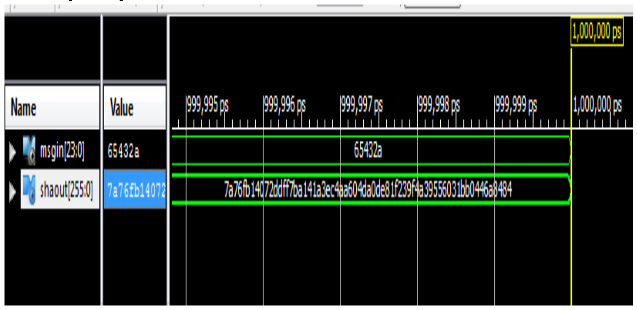
In this step, we have used conventional 64 bit XOR operator to perform XORing between the least significant 64-bits of state array and round constant RC. The values of round constant are fixed and different for every round. These round constants are stored in registers.

VI.RESULTS

The simulation and synthesis is performed in Xilinx ISE 13.2 (Virtex 7).

A.XILINX Simulation Results

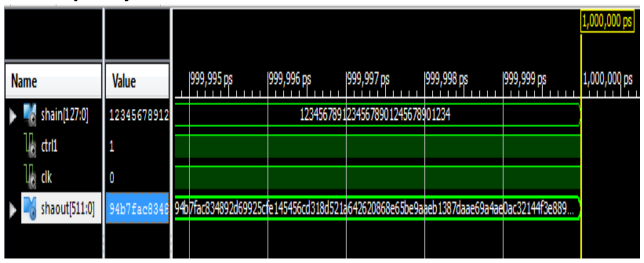
SHA-2(256)



```
msgin[23:0]    65432a       65432a
shaout[255:0]  7a76fb14072  7a76fb14072ddff7ba141a3ec4aa604da0de81f239f4a39556031bb0446a8484
```

Fig 7: Simulation waveform for SHA-2(256)

SHA-3(512)



```
shain[127:0]   12345678912  123456789123456789012456789012 34
ctrl1          1
clk            0
shaout[511:0]  94b7fac8348  94b7fac834892d69925cfe145456cd318d521a642620868e65be9aaeb1387daae69a4ae0ac32144f3e889...
```

Fig. 8 Simulation waveform or SHA-3(512)

B.Xilinx Synthesis Results

Parameter	SHA-2(256)	SHA-3(512)
No. of Slice LUTs	27127/612000	99245/612000
No. Of IOBs	280/720	642/720

VI.CONCLUSION

SHA-2(256) is more secure authentication protocol for which any cryptanalysis is not yet reported. SHA-3(512) is more secured than SHA-2(256).In this paper both the hash algorithms are implemented and the area consumed by both hash functions is compared. SHA-3(512) utilizes almost 12% more area consumed by SHA-2(256). Both the hash functions have equal importance based on the applications they are being used. Other Cryptographic hash functions such as Extendable Output Functions (XOF) functions can also be implemented in a similar fashion.

REFERENCES:

1. U. Maurer. Cryptography 2000±10. Informatics—10Years Back, 10Years Ahead, Lecture Notes in Computer Science, Springer-Verlag, vol. I, pp. 63–85, 2001.

2. SHA-2 Standard, National Institute of Standards and Technology (NIST), Secure Hash Standard, FIPS PUB180-2, www.itl.nist.gov/fipspubs/fip180-2.htm, 2002.
3. S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk. Cryptographic hash functions: A survey. Technical Report 95-09, Department of Computer Science, University of Wollongong, July 1995.
4. National Institute of Standards and Technology (NIST). Digital Signature Standard,FIPS PUB 186-2,http://csrc.nist.gov/publications/fips/fips186-2.htm, 2002.
5. R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems.*Communications of the ACM Proc.*, 21(2):120–126, 1978.
6. SHA-1 Standard, National Institute of Standards and Technology (NIST). Secure Hash Standard, FIPS PUB180-1, www.itl.nist.gov/fipspubs/fip180-1.htm, 2002.
7. X. Wang, D. Feng, X. Lai, and H. Yu, “Collisions for hash functions md4, md5, haval-128 and ripemd,” IACR, August 2004.
8. National Institute of Standards and Technology (nist), “Cryptographic hash algorithm competition,” 2007.
9. FIPS-202, “Federal information processing standards publication fips-202, secure hash algorithm-3 (sha-3),” 2014.