VOLUME-6 | ISSUE-1 | JANUARY-2017 • ISSN No 2277 - 8179 | IF : 3.508 | IC Value : 78.46

Original Research Paper

Enactment of ChaCha Stream Cipher based BLAKE Function using SHA

Engineering

KEYWORDS: SHA, BLAKE, ChaCha, cryptography, hash function, Encryption, Decryption.

Sudheesh S.R Asst. Prof in Dept. of ECE Mount Zion College of Engineering

ABSTRACT

Cryptographic hash functions are used to protect information integrity and authenticity in a wide range of applications. After the discovery of weaknesses in the current deployed standards, the U.S. Institute of Standards and Technology started a public competition to develop the future standard SHA-3, which will be implemented in a multitude of environments, after its selection in 2012. In this paper the VLSI implementation of one of the 14 "second-round" candidates BLAKE for 64 bit and the round rescheduling technique design are proposed by using modulo 2n adder and adiabatic multiplexer for high throughput when compared to SHA 2.

1. INTRODUCTION

A cryptographic hash function is a hash function, that is, any algorithm that takes an arbitrary block of data and returns a fixed-size bit string a hash function is generated by a function H of the form h=H(N) where N is a variable-length message and H(N) is the fixed length hash value. Hash functions are used in a multitude of protocols, be it for digital signatures within high-end servers or for authentication of embedded.

All hash function operate using bit-by-bit exclusive-OR (XOR) function

$$Ci= b_{i1} \oplus b_{i2} \oplus ..... \oplus b_{im}$$

Md5, SHA-2, and their variants are the most popular hash algorithms [1]. They follow the Merkle - Damgard model and use logic operations such as AND, OR, and XOR in their compression functions. Recently, collision pairs have been found for MD5, SHA-0 and SHA-1 making these algorithms vulnerable to attacks because they do not have the collision resistance property [2]. Due to security concerns of SHA-1 and recent advances in the cryptanalysis of hash algorithms a new hash algorithm standard, SHA-3, which is meant to replace SHA-2. SHA-3 is expected to have at least the security of SHA-2, and to achieve this with significantly improved efficiency besides a sufficient security level, SHA 3 should be implementable on a wide range of environments for security. BLAKE is a second round candidate in the NIST Hash Competition. BLAKE is one of the simplest designs to implement.

The BLAKE hash functions were designed to meet all NIST criteria for SHA-3, including:

message digests of 224, 256, 384, and 512 bits
same parameter sizes as SHA-2
one-pass streaming mode
maximum message length of at least 264 – 1 bits
In addition, we imposed BLAKE to:
explicitly handle hashing with a salt
be parallelizable
allow performance trade-offs
be suitable for lightweight environments.

2. EXPECTED STRENGTH

For all BLAKE hash functions, there should be no attack significantly more efficient than standard brute force methods for

- finding collisions, with same or distinct salt
- finding (second) pre images, with arbitrary salt

BLAKE should also be secure for randomized hashing, with respect to the experiment described by NIST. It should be impossible to distinguish a BLAKE instance with an unknown salt (that is, uniformly chosen at random) from a PRF, given black box access to the function; more precisely, it shouldn't cost significantly less than 2|s| queries to the box, where |s| is the bit length of the salt. BLAKE should have no property that makes its use significantly less secure than an ideal function for any concrete application. (These claims concern the proposed functions with the recommended number of rounds, not reduced or modified versions.). The inner state of the compression function is represented as a 4×4 matrix of words. A round of BLAKE-256 is a modified "double-round" of the stream cipher ChaCha: first, all four columns are updated independently, and thereafter four disjoint diagonals. In the update of each column or diagonal, two message words are input according to a round-dependent permutation. Each round is parameterized by distinct constants to minimize self-similarity. After the sequence of rounds, the state is reduced to half its length with feed forward of the initial value and the salt. An implementation of BLAKE requires low resources, and is fast in both software and hardware environments. In 180 nm ASIC, BLAKE-256 can be implemented with about 13 500 gates, and can reach a throughput of more than 4Gbps; BLAKE-512 can be implemented with about X Y gates, and can reach a throughput of more than 6Gbps.

3. COMPRESSION FUNCTION

Henceforth we shall use the following notations: if is m a message (a bit string), m^i denotes its i^{th} 16-word block, and m_j^i is the j^{th} word of the i^{th} block of m. Indices start from zero, for example a N -block message m is decomposed as $m=m^0 m^1 .....m^{N-1}$, and the block m^0 is composed of words $m_0^0.m_2^0.m_3^0.....m_{15}^0$ .We use similar notations for other bit strings.

a chain value h = h0, . . . , h7
a message block m = m0, . . . ,m15
a salt s = s0, . . . , s3.
a counter t = t0, t1

These four inputs represent 30 words in total (i.e., 120 bytes = 960 bits). The output of the function is a new chain value $h^0 = h_0^0, \ldots, h_0^7$ of eight words (i.e., 32 bytes = 256 bits). We write the compression of h,m, s, t to h0 as

$$h^0 = \text{compress}(h,m, s, t)$$

The compression function of BLAKE-64 makes 14 rounds instead of ten, and that Gi (a, b, c, d) uses rotation distances 32, 25, 16, and 11, respectively. After ten rounds, the round function uses the permutations 0, . . . , 4 for the last four rounds. The compressed function can be decomposed into initialization, round function and Finalization.

3.1 INITIALIZATION

At the initialization stage, constants and redundancy of the impose a nonzero initial state [5].The disposition of inputs implies that after the first column step the initial value h is directly mixed with the salt s and the counter. It consists of 16 word internal states (v_0,v_1.....v_{15}) is initialized such different input produce different state. Initialization takes the input as chaining value and produces the output as internal states. It represent as 4×4 matrix

VOLUME-6 | ISSUE-1 | JANUARY-2017 • ISSN No 2277 - 8179 | IF : 3.508 | IC Value : 78.46

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix}.$$

The initial state is defined as follows:

$$\begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

3.2 ROUND FUNCTION

Once the state v is initialized, the compression function iterates a series of 14 rounds.

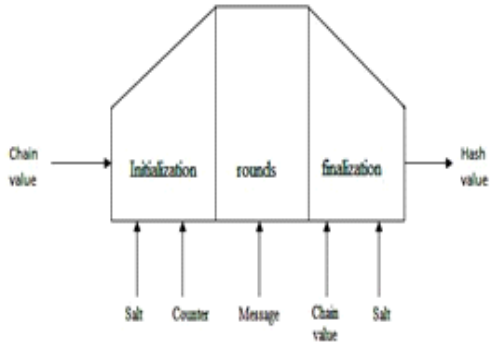

Figure 1: Block diagram of compression function

A round is a transformation of the state v that computes

G0(v0 , v4 , v8 , v12) G1(v1 , v5 , v9 , v13)

G2(v2 , v6 , v10, v14) G3(v3 , v7 , v11, v15) and then

G4(v0 , v5 , v10, v15) G5(v1 , v6 , v11, v12)

G6(v2 , v7 , v8 , v13) G7(v3 , v4 , v9 , v14)

The figure 1 shows the diagram is a compression function for BLAKE 64 with input hash h, salt s, counter t and message m. The round function takes the input as message, internal states v_0 to v_{15} and output is fin_v_{0to} fin_v_{15}.The sequence G0 to G3 is called a column step. Similarly, the last four calls G4

to G7 in update distinct diagonals and are called a diagonal step. The input message block is padded into 1024 bits. The unary operator >>> denotes rotation of words towards least significant bits.

3.3 PADDING

First the message is extended so that its length is congruent to 447 modulo 512. Length extension is performed by appending a bit 1 followed by a sufficient number of 0 bits. At least one bit and at most 512 are appended. Then a bit 1 is added, followed by a 64-bit unsigned big-endian representation of L. This procedure guarantees that the bit length of the padded message is a multiple of 512.

3.4 FINALIZATION

After the rounds sequence, the new chain value h00 to h07 is extracted from the state v0, . . . , v15.With input of the initial chain value h0, . . . , h7 and the salt s0, . . . , s3:

$HH_0 <= h0 \oplus s0 \oplus v0 \oplus v8;$

$HH_1 <= h1 \oplus s1 \oplus v1 \oplus v9;$

$HH_2 <= h2 \oplus s2 \oplus v2 \oplus v10;$

$HH3 <= h3 \oplus s3 \oplus v3 \oplus v11;$

$HH_4 <= h4 \oplus s0 \oplus v4 \oplus v12;$

$HH_5 <= h5 \oplus s1 \oplus v5 \oplus v13;$

$HH_6 <= h6 \oplus s2 \oplus v6 \oplus v14;$

$HH_7 <= h7 \oplus s3 \oplus v7 \oplus v15;$

4. HASHING A MESSAGE

When hashing a message, the function starts from an initial value and the iterated hash process computes intermediate hash values that are called chaining values. Before being processed, a message is first padded so that its length is a multiple of the block size (512 bits). It is then processed block per block by the compression function, as described in the following:

$h^0 := \mathrm{IV}$
for $i = 0, \ldots, N-1$
$\quad h^{i+1} := \mathrm{compress}(h^i, m^i, s, t^i)$
return h^N.

5. HASHING A SALT

The BLAKE hash functions take as input a message and a salt. The aim of hashing with distinct salts is to hash with different functions but using the same algorithm. Depending on the application, the salt can be chosen randomly (thus reusing a same salt twice can occur, though with small probability), or derived from a counter (nonce). For applications in which no salt is required, it is set to the null value (s = 0). In this case the initialization of the state v simplifies to

6. IMPLEMENTATION OF BLAKE

BLAKE, with an iterative decomposition of the round process .Different architectures are made possible by varying the number of integrated G modules. Modern high-speed communication systems where the space is not a fierce constraint can take advantage of architectures with eight G modules or even with a complete round-unrolled circuit. At the opposite, by scaling the number of G modules the design becomes slower but decreases in size. Besides the round computation, BLAKE requires some circuitry to perform initialization and finalization; for instance, -bit XORs are required to compute , where for BLAKE-32 and for BLAKE-64. Furthermore, the complete execution of initialization and finalization can be performed in the same clock cycle, when the new message block is given. Like most hash functions, BLAKE uses some constant values,

which are as follows:

- the initial value (eight -bit words);
- the 16 round constants ;
- the ten permutations (in total of 640 bits).

The figure 2 shows the architecture of Blake 64 with 8G BLAKE requires some circuitry to perform initialization and finalization for instance, w = 64 for BLAKE-64 the complete execution of initialization and finalization can be performed in the same clock cycle.

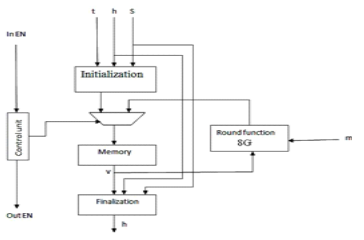

Figure 2: architecture of Blake 64 with 8G core

7. ROUND RESCHEDULING

The G function of BLAKE is a modified version of the core function of the stream cipher ChaCha proposed by Bernstein in the context of the e STREAM Project. Speed limits for plain designs implementing several architectures of ChaCha have been reported. The introduction of the addition with the message/constant (MC)-pair in the G function leads to an increment of the propagation delay. If in the core function (similar to) the maximum delay is given by the total delay of four XORs and four modular adders (rotation is a simple rerouting of the word without effective propagation delay), the slightly modified G function inserts an addition with the MC-pair. Accordingly, the maximum frequency values of analogous BLAKE architectures are slightly lower than those obtained for the stream cipher ChaCha. However, with a rescheduling of the G computation, it is possible to recover the original maximum path of ChaCha (four XORs and four adders), hence decreasing the overall propagation delay of the core function. Observing the flow dependencies is clear that the addition with the MC-pair is independent (message word and constant are unrelated to the state) and can be computed in parallel to the other computations. If in a single call of , similarly to the core function of ChaCha, each update of the state has been conceived to operate sequentially, the MC-pair addition can be shifted within the computations. It is thus possible to anticipate it, reducing the critical path of G . Since BLAKE iterates a series of rounds over an internal state, additional sequential components are required to store the following 44 values:

- the 8-word chaining value ;
- the 16-word internal state ;
- the 4-word of the salt value ;
- the 16-word message block .

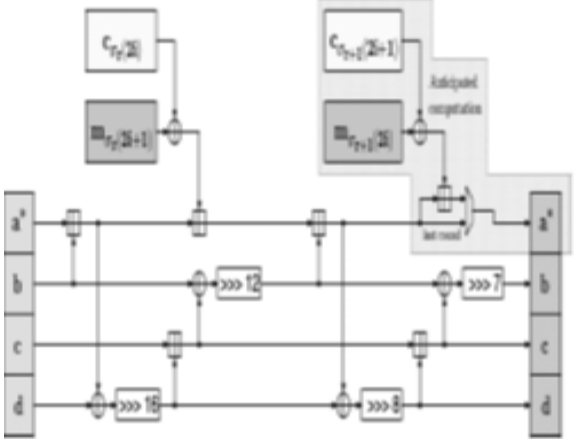

Figure 3: Block diagram of the rescheduled G function

To evaluate the speedup provided by the G rescheduling, we coded the 8G and 4G architectures in VHDL and we synthesized them for BLAKE-32 and BLAKE-64 with the Synopsys Compiler. Our results refer to fully-autonomous designs, which take as input salt, counter, and message blocks and generate the final hash value. Moreover, to obtain an exhaustive analysis of the BLAKE hash cores, the designs have been synthesized in three different UMC technologies: 0.18 m, 0.13 m, and 90 nm. Note that our designs for BLAKE-32 support salted hashing. Although very similar, the design presented in inserts a pipeline register at the output of the permutation.

8. MEMORY ARCHITECTURE

The VLSI implementation of BLAKE-32 needs memory to store 16 words of internal state and eight words of chaining value, plus additional registers to store the salt (four words), the counter (two words), and the message block (16 words), i.e., in total 1472 bits of memory. The counter is used during four clock cycles and needs thus to be stored. Compared to the minimum circuit needed to implement the compression function (initialization, rounds, and finalization), the memory units is the main contribution in terms of area and energy consumption. It is thus of primary interest to design special-purpose register elements, to decrease the global resource requirements of the hash core. We introduced in the compact architecture of BLAKE-32 semi-custom memories based on clock-gated latch arrays, able to store at most one new word per cycle. In general, depending on the word number of the target value to be stored, these memories replace the standard flip-flop cells by latch cells. The latches are organized in 32-bit banks, so that each bank stores a single word and is triggered by a dedicated gated clock.

9. CONCULSION

In the Blake 64 bit architecture, the design of Blake 32 bit architecture is modified by using MOD 2 adder and adiabatic multiplexer .compare to Blake 32 bit architecture this modified design provide high throughput and low area, low power in rescheduling G function. By the method of initialization, round function and finalization in the network security application the Blake 64 bit generate hash value .In future enhancement by using carry save adder to increase the throughput and reduce the propagation delay.

10. REFERENCES

[1] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender
[2] NIST, "Announcing the secure hash standard," FIPS 180-2, Technical report, 2002
[3] R. Lien, T. Grem bowski, and K. Gaj, "A 1 Gbit/s partially unrolledarchitecture of hash functions SHA-1 and SHA-512," in Topics in Cryptology - CT-RSA 2004, ser. Lecture Notes in Computer Science , vol. 2964.Springer Berlin / Heidelberg.
[4] Wang and H. Yu, "How to break MD5 and other hash functions, "in Advances in Cryptology - EUROCRYPT 2005, ser. Lecture Notes inComputer Science, vol. 3494. Springer Berlin / Heidelberg, 2005, pp.19–35
[5] C. D. Cannière and C. Rechberger, "Finding SHA-1 characteristics: General results and applications," in Advances in Cryptology – ASIA CRYPT2006, ser. Lecture Notes in Computer Science, vol. 4284. SpringerBerlin / Heidelberg, 2006, pp. 1–20
[6] J.-P. Aumasson, L. Henzen, W. Meier, and R. C.-W. Phan, "SHA-3 proposal BLAKE," Submission to NIST, 2008,
[7] Luca Henzen, Student Member, IEEE, Jean-Philippe Aumasson, Willi Meier, andRaphael C.-W. Phan, Member, IEEE "VLSI Characterization of the Cryptographic Hash Function BLAKE" Oct. 2011
[8] L.Rernstein and T. Lange, "eBACS: ECRYPT Benchmarking of Cryptographic Systems," Aug. 2010. [Online]. Available:http://bench.cr.yp.to/
[9] S. Tillich, M. Feldhofer, M. Kirschbaum, T. Plos, J.-M. Schmidt, and A. Szekely, "High-speed hardware implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein," Cryptology e Print Archive, Rep. 2009/510, 2009.
[10] S. Tillich, M. Feldhofer, W. Issovits, T. Kern, H. Kureck, M. Mühlberghuber, G. Neubauer, A. Reiter, A. Köfler, and M. Mayrhofer, "Compact hardware implementations of the SHA-3 candidates ARIRANG, BLAKE, Grøstl, and Skein," Cryptology ePrint Archive: ,Rep. 2009/349, 2009.