Area Efficiency And High Speed Implementation Of Cryptography Using Hash Function And RSA Algorithm

Mr.S.Jayakumar, A.Sumathi, P.Santhosh Kumar, M.Suganthan, S.Syed Hanies

Assistant Professor, Department of Electronics and Instrumentation Engineering, Adhiyamaan College of Engineering, Hosur
Professor, Head of Department of Electronics and communication Engineering, Adhiyamaan College of Engineering, Hosur.

Students, Department of Electronics and Instrumentation Engineering, Adhiyamaan College of Engineering, Hosur.

Received 28 January 2017; Accepted 22 March 2017; Available online 28 April 2017

ABSTRACT
In modern term, evaluation of networking and wireless networks has come forward to allowance message anywhere at any time. Security of wireless networks and the process of cryptography plays an significant role to provide security to the wireless networks. The security of authenticity and integrity of information is necessary to achieve a secure communication between communicating parties. Cryptographic hash functions are used to protect information reliability and accuracy in a wide collection of applications. Hash functions form an important group of cryptography, which is widely used in a great number of protocols and security mechanisms. Here we use BLAKE hash function and The RSA ALGORITHM which implements a high speed implementation and high secure data communication. The tools used here is Xilinx ISF 14.5.

KEYWORDS: Encryption, decryption, BLAKE, RSA.

INTRODUCTION
Cryptography is the preparation and study of techniques for secure message while third parties are present. A communication in its original form is known as plain text or clear text. The twisted information is known as cipher text. The two process in cryptography are encryption and decryption. Encryption is the standard method for making message personal. The process used for producing cipher text from plaintext is known as encryption. Anyone who wants to send a private message to another user encrypts the message before transmitting it[2]. The reverse of encryption is called decryption. Only the projected recipient knows how to correctly decrypt the message. Anyone who was “eavesdropping” on the communication can only see the encrypted message[1]. In Secret key cryptography (symmetric key cryptography) both the dispatcher and the recipient know the same secret code, called the key. Public key cryptography (asymmetric encryption) which uses a pair of keys for encryption and decryption and Hash Function.

II. Literature Review:
The Blake hash function like many other hash functions was designed with the intent of making it capable of running at high speed. It has a relatively simple algorithm; its compression function is a modified “double round”
version of Bernstein’s stream cipher “chacha” which has been intensively analyzed and found to be of excellent performance and parallelizable [3]. Blake has been examined by researchers seeking ways of providing high speed operation. One of the techniques for speed optimization of Blake that is found in literature is parallelism[4]. Other speed optimization techniques that have been applied to Blake are pipelining (in an area of the algorithm where pipelining is feasible) [6] and the use of carry-save adders [5] in the compression function. These techniques focus on the main ‘core’ of the hash function.

III. Purpose of Cryptography:
A. Authentication:
Authentication mechanism help to establish facts of identities. This process ensures that the origin of the message is correctly recognized.

B. Confidentiality:
The opinion of confidentiality specifies that the dispatcher and the intended recipient should be able to process the contents of a message.

C. Availability:
The principle of availability states that resources should be available to authorized parties all the times.

D. Integrity:
The veracity mechanism confirms that the contents of the communication remain the same when it reaches the intended recipient as sent by the sender.

E. Access Control:
Access Control specifies and controls who can access the process

Problem Statement:
Hash functions are very useful in information security schemes. Apart from the applications (digital signatures, digital image watermarking and so on), hash functions are also utilized in generating pseudo random numbers which are in turn utilized in many cryptographic schemes. In most of these applications, particularly digital signatures, digital image watermarking and Message Authentication Codes, it is desirable to have the hash function operate as fast as possible especially when a huge traffic or load of messages are expected to be operated on. Here the high speed implementation of Hash functions is performed and analysed.

Types & Terminology Used In Cryptography:
Secret Key Cryptography:
Here both the sender and the receiver know the same secret code called the key.

Public Key Cryptography:
When two different keys are used, that is one key for encryption and another key for decryption.

Plain text:
The plain message which will be converted into encrypted message.

Cipher text:
A message is in encrypted form.

Key:
An main aspect of performing encryption and decryption is the key. That the key used for encryption and decryption that makes the process of cryptography very secure.

Hash Function:
A. MD5 – Message digest algorithm 5:
Takes random data as input and generate a fixed size hash value as a output. the input size are given, the algorithm generates a fixed size. The MD5 is almost broken as lot of collisions have been initiate.

B. SHA-0- Secured hash algorithm:
It has been quickly retained due to an undisclosed flaw. It was replaced by SHA-1
C. SHA-1:
A 160 bit hash function that is related to previous algorithm but more conservative. It is the most generally used SHA algorithm.

D. SHA-2:
The message block size is same as MD5. This algorithm design has no known security vulnerabilities because the construction has been successful against generic attacks[7].

E. SHA-3:
Due to security concerns of SHA-1 and recent advances in the cryptanalysis of hash algorithms, NIST held a public competition for a new hash algorithm standard SHA-3, which is meant to replace SHA-2[8]. Therefore, the new algorithm is expected to be more secure SHA-2.

F. BLAKE:
It is a modified version whose security has been intensively analysed and performance is excellent and presents an high speed application.

Proposed Method of Blake Hash Function:
The main input to the hash function is the message block input and the main output is the digest (or hash value). The message block input takes in a 512-bit message block which may represent text, image pixels or any kind of information. There are other inputs (salt and counter), one of which is optional. A top-level diagram of Blake-256 with its inputs and outputs is shown in figure 1 below (when we make reference to the Blake hash function, we normally refer to its core functionality alone as depicted in figure 1; that is the unit that hashes only individual message blocks). The main input to the hash function is the message block input and the main output is the digest (or hash value). The message block input takes in a 512-bit message block which may represent text, image pixels or any kind of information. There are other inputs (salt and counter), one of which is optional. The designers of Blake did not re-invent the wheel; rather they put together components which had been previously analyzed and found to be secure and effective to form Blake.

Fig. 1: Top level diagram of BLAKE 256

Round Rescheduling:
The core function of stream cipher is modified version of G function of BLAKE. The introduction of the addition with the message/constant (MC) -pair in the G function leads to an increment of the transmission delay. If in the core function (similar to G) the maximal delay is given by the total delay of four XORs and four modular adders (rotation is a simple re-routing of the word without effective propagation delay), the slightly modified G function inserts an addition with the MC-pair. The compression function can be decomposed into three main steps, described in II-A1a) to II-A1c). Fig 2 shows the block diagram of modified G function.
Fig. 2: Gi Function

Compression Function:
The compression function of BLAKE-256 takes as input four values are Chain value, Message block, Salt and Counter
- a chain value given an input value \( h=h_0,....h_7 \)
- a message block \( m=m_0,......m_{15} \)
- a salt values message \( s=s_0,...s_3 \)
- a counter values are fixed \( t=t_0,t_1 \)

The compression function of \( h, m, s, t \) is given as Compressed by \( (h ,m, s, t) \). The compression function performs three main operations as shown in fig 3

\( t = \) counter  \\
\( h = \) chaining values  \\
\( s = \) salt values message  \\
\( v = \) internal values  \\
\( m = \) inputs

Fig. 3: Overall flow diagram of BLAKE 32

a) Initialization:
The internal states of BLAKE will first be initialized using a set of initial value. Thus the first step of the compression function is initialization.
This state is represented as a 4×4 matrix:
The initial state value is defined as follows:

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

where $c_0,...,c_{15}$ are predefined word constants.

b) **RoundFunction:**
A round function is a transformation of the state $v$ that compute

$G_0(v_0,v_4,v_8,v_{12})$
$G_1(v_1,v_5,v_9,v_{13})$
$G_2(v_2,v_6,v_{10},v_{14})$
$G_3(v_3,v_7,v_{12},v_{15})$

parallelly because each updates a dissimilar column of the state. Are

$G_4(v_0,v_5,v_{10},v_{15})$
$G_5(v_1,v_6,v_{11},v_{12})$
$G_6(v_2,v_7,v_8,v_{13})$
$G_7(v_3,v_4,v_9,v_{14})$

**Fig. 4:** State update column
Fig 4: State update diagonals

c) Finalization:

The finalization stage is the last process in the computation of the hash value of a message block. In this stage, the hash value is extracted from the updated state. The salt (s) and initial value (h_0) which were used to initialize the state are again used in the extraction of the hash code, but the counter is not used. When the salt input is not used, its value is set to 0 and it simply functions as a constant. Essentially, a set of XOR operations are performed using the initial value, salt and 2 state variables in this stage. Thus its gives the initial values used for the first block of a message (referred to as the initialization vector) in both the state initialization and the finalization processes.

**RSA Algorithm:**

RSA algorithm (named after its founders, Ron Rivest, Adi Shamir, and Leonard Adleman) has become almost synonymous with public key cryptography. In RSA algorithm the interesting factor is that it can be allow most components used in encryption process and also can be used in decryption process. From this factor it can minimised a hardware area.

The RSA algorithm involves in both public key and private key. The public key which is known to everyone and this key can be used to encrypt the message. The messages which can be encrypted with the help of public key it can be decrypted only by using private key.

- The p and q are the two prime numbers which can be chosen. This integers p and q should be choose at random and similar bit length for security purpose
- For the both public and private key n is used as modules. While we can compute n=pq.
- Compute φ(n) = (p-1)(q-1), where φ is Euler’s totient function.
- Choose an integer e such that 1 < e < φ(n) and gcd(e,φ(n)) = 1, i.e. e and φ(n) are co prime. e is released as the public key exponent having a short bit-length and small Hamming weight results in more efficient encryption - most commonly 0x10001 = 65537. However, small values of e (such as 3) have been shown to be less secure in some settings.
- Determine d = e⁻¹ mod φ(n); i.e. d is the multiplicative inverse of e mod φ(n).

| Table 1: RSA method public and private key pairs |
|------------------|---|---|---|---|---|
| Prime | Prime | n=pq | m | E | calc. 'd' | Private n,d | Public n,e |
| 5 | 5 | 15 | 8 | 11 | 3 | 73,3 | 75,11 |
| 7 | 5 | 35 | 24 | 11 | 11 | 35,1 | 35,11 |
| 13 | 7 | 221 | 192 | 11 | 11 | 221, 35 | 221, 11 |
| 17 | 23 | 391 | 352 | 5 | 5 | 391, 141 | 391, 141 |

**RESULT AND DISCUSSION**

In this project we have used hash function BLAKE 32 to perform an high speed security mechanism. Table 3 shows the comparison between the previously proposed method and our method. We have reduced total delay, gate count(size) and memory which have been high in the previous method.

<table>
<thead>
<tr>
<th>Table 2: Synthesis result for BLAKE 32 using RSA algorithm.</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reduced Functions</strong></td>
<td><strong>BLAKE 32</strong></td>
<td><strong>Proposed method</strong></td>
<td></td>
</tr>
<tr>
<td>Delay</td>
<td>40.648ns</td>
<td>6.12ns</td>
<td></td>
</tr>
<tr>
<td>Gate count(size)</td>
<td>13,390</td>
<td>4,306</td>
<td></td>
</tr>
<tr>
<td>Memory usage</td>
<td>170916Kb</td>
<td>187860kb</td>
<td></td>
</tr>
</tbody>
</table>
Refer Table 2. As per the process although the trade off value of delay and area is reduced, the memory is increased. The speed is also increased.

Refer fig 5. The Process which Sender and receiver can Send their Messages with secured purpose. In indata is that Sender can give the input, inexp is the key for two parties, inmod is the multiple of two values and decypher is output which the receiver can receive from sender.

**Fig. 5:** output of cryptography

- indata - Input
- inexp - key for two parties
- inmod - multiple of two values
- decypher - output

**Conclusion:**

In this thesis, high speed FPGA implementation of HASH functions was explored using the Blake hash function, one of the SHA-3 candidates. A new design for the Blake hash function which incorporates the ability to recognize common message blocks, store the hash codes of the common message blocks in memory and skip the computation of the hash codes of these blocks when they are subsequently encountered was proposed. From the performance analysis of the proposed design, it is evident that the speed of the Blake hash function can be improved by the proposed design. The higher speed performance comes into play when common message blocks are encountered in the messages being hashed. When there are no common message blocks, the proposed design operates at the same speed as the original Blake design. The speed improvement becomes more significant when a large number of messages with common message blocks are hashed.

**ACKNOWLEDGEMENT**

I would like to thank our HOD Dr. S. Sujatha, for her valuable feedback. I would also like to thank Mr. S. Jayakumar our guide for his guidance and support for this research work.

**REFERENCES**