Basic Operations of Modern Hashing Algorithms

Viktor A. Melnyk[1], Andriy Y. Kit[2]

[1]Department of Information Technologies Security,
Lviv Polytechnic National University, UKRAINE, Lviv,
S. Bandery street 12, E-mail: vamelnyk@lp.edu.ua

[2]Department of Information Technologies Security,
Lviv Polytechnic National University, UKRAINE, Lviv,
S. Bandery street 12, E-mail: andrew.kit.91@gmail.com

Abstract – In this paper we performed a comparative analysis of hashing algorithms SHA-1, SHA-2, MD-5, RIPEMD, Keccak respecting a set of their basic operations. As an outcome of this analysis a qualitative and quantitative composition of elements of the multifunctional high-performance hashing processor for execution any of these algorithms is defined.

Key words – hash function, SHA-1, SHA-2, MD-5, RIPEMD, Keccak, basic operations.

I. Introduction

The hashing is a process of transformation of the input data set of arbitrary length to the output bit string of fixed length. Such transformations are called hash functions. In general, a hash functions may be divided into hash functions of general purpose and cryptographic ones, which are widely used in cryptography and satisfy a number of requirements that are imposed by cryptographic applications. Any cryptographic hash function should be resistant to collisions. That is excluding the possibility, or rather, to make it as low as possible, of the existance of two posts that would produce the same hash. In this paper we focus on the cryptographic hash functions that are built "from scratch". We consider all the basic operations used in these algorithms and what should be executed by each separate digital element of the processor. Such an approach in the future will enable us to design a device that performs hash for any of hash algorithms from the set of algorithms.

Hardware implementation of hashing, as well as encryption, has a number of advantages over the software one. First of all, it is speed of transformation execution, because only with dedicated device designed to solve a specific problem, one can reach a top speed. The second is security. If the device for cryptographic transformation is implemented as a separate module, it is possible to implement a number of additional measures for protection. In particular, the device can be made so that any intrusion into its internal structure would violate its operation. Besides that the hardware devices are easy to integrate with unite computer system and that is easier than writing relevant software applications [1].

The hashing algorithms belong to computationally complex algorithms and intended to process large volumes of data and wide bit words. Implementation of these algorithms in general purpose processors is generally inefficient, thus specialized hashing processors are often being developed. However, the functionality of these processors is narrow and is often limited to execution of the single algorithm. In order to implement multifunctional hashing processor the task arises to analyse the basic operations of the hashing algorithms and to define elements required for their fast execution. This is the subject that is being addressed in this paper.

II. Modern hash functions

We consider two commonly used methods of hash functions construction. First one that uses a custom encryption algorithm like AES [2], while the second one is building "from scratch".

In the first type of the hash functions the symmetric block encryption algorithm is being used as a function of compression. In this case, in order to achieve a higher level of security, as a key the data block is being used, that is designed to calculate a hash in a given iteration, and a result of previously applicated compressing functions is being used as an input data. The resulting hash function is formed after the last iteration of the algorithm. In this case, the reliability of the hash function is determined by the reliability of used symmetric block encryption algorithm. The N-hash algorithm (developed by Nippon Telephone and Telegraph in 1990) is an example of this type of hashing algorithms [3].

The idea of using a block encryption algorithm, which reliability is known, to get a robust hashing schemes, looks natural. However, some of these hash functions have difficulties, namely the need of use the key that is applied for encryption and that should be kept in secret.

Another weakness of the above hashing schemes is that the size of the hash code matches with the size of a block encryption algorithm. Often, the size of block is insufficient to make the scheme stable against attacks based on "birthday paradox " [4]. In addition, a significant drawback of hash functions designed on the base of symmetric block encryption algorithms are relatively low speed of their operation.

In hashing algorithms built "from scratch", the function of compression (it`s the main element of the Merkle-Damharda scheme, which stability determines the stability of the algorithm [5]) is built from scratch and must meet the basic requirements of reliability. Execution speed of this type of hash functions is by an order higher than that of hash functions based on the symmetric block encryption algorithm. So when the high speed is a demand then a hash function built "from scratch" should be used.

III. The life cycle of cryptographic hash functions

In Fig. 1 [6] we show the years when each of the algorithms was created, depressed and broken. Despite the fact that the algorithm MD5 [7] was broken in 2004, it still remains relevant as continues to be widely used along with the algorithm SHA-1 [8]. In 2012 the Keccak [9] algorithm was won SHA-3 competition [10], but its replacement instead of the SHA-2 has not yet been scheduled as no significant attacks on SHA-2 algorithm were yet demonstrated [11].

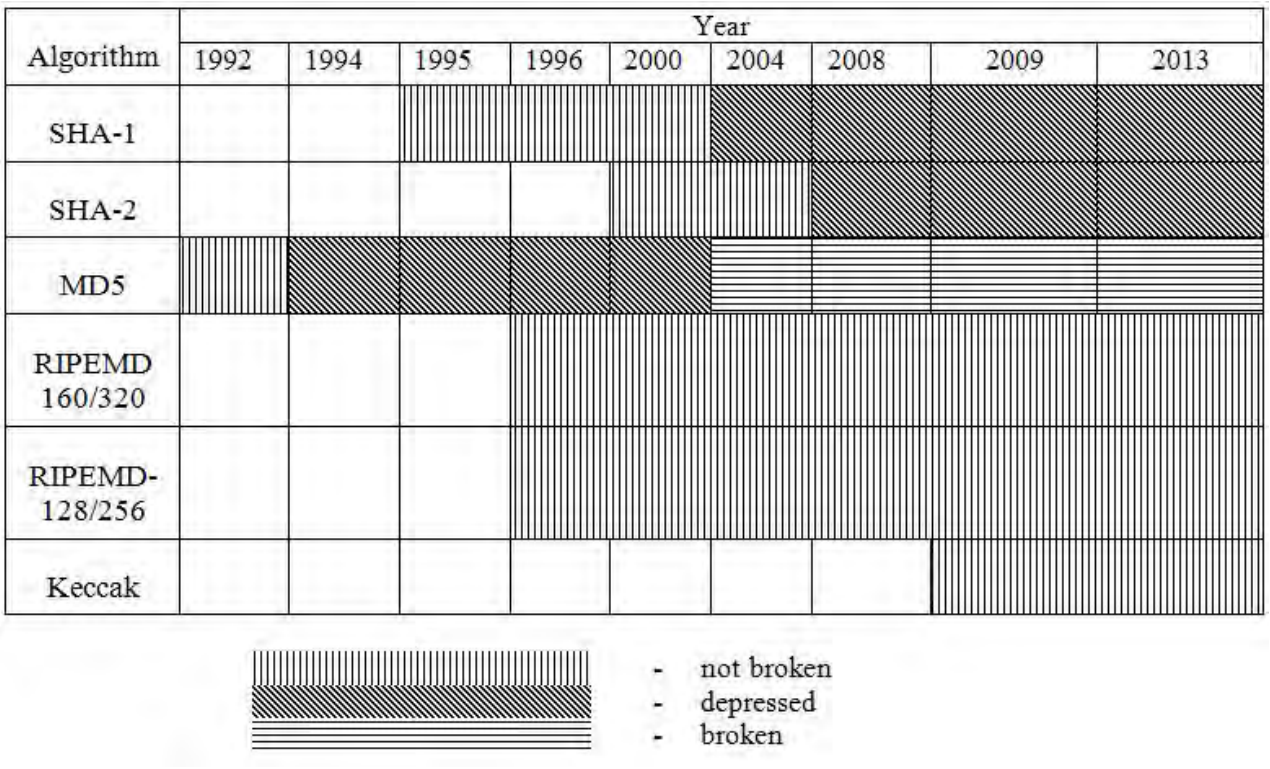

Fig. 1. The life cycle of cryptographic hash functions

IV. Basic operations of hashing algorithms SHA-1, SHA-2, MD-5, RIPEMD and Keecak

Let's analyze discussed above hashing algorithms and identify which basic operations are being used in there.

The basic operations of the algorithm SHA-1 are cyclic shift left, logical shift right, logical addition (OR), addition modulo 2 (XOR), arithmetic addition, logical multiplication (AND), and inversion (NOT).

The basic operations of the algorithm SHA-2 are cyclic shift right, logical shift right, addition modulo 2, arithmetic addition, logical multiplication, and inversion.

The basic operations of MD5 algorithm are cyclic shift left, multiplication, logical addition, addition modulo 2, arithmetic addition, logical multiplication, and inversion.

The basic operations of the algorithm RIPEMD [12] are cyclic shift left, logical addition, addition modulo 2, arithmetic addition, logical multiplication, and inversion.

The basic operations of the algorithm Keccak are cyclic shift right, addition modulo 2, logical multiplication, and inversion.

While designing of the hashing processor the following considerations should be taken into account:

1. During execution of each of the considered hashing algorithms a certain number of basic operations of each type are being executed. However, this does not mean that in implementation of appropriate hashing processor the number of a arithmecic/logic elements must be the same as the number of times one or other relevant operation is being executed.
2. In order to achieve high performance the computations need to be split into parallels. However, not all parts of the algorithm can be executed in parallel. For instance, in SHA-256 algorithm after message separation by 16 words of 32 bits each, one needs to generate another 48 words. This part of algorithm is represented on the C # below:

```
for (int j = 16; j < 64; j++)
{
   uint s0 = Mathem.Rotr(w[j - 15], 7) ^
             Mathem.Rotr(w[j - 15], 18) ^
                        (w[j - 15] >> 3);
   uint s1 = Mathem.Rotr(w[j - 2], 17) ^
             Mathem.Rotr(w[j - 2], 19) ^
                        (w[j - 2] >> 10);
   w[j] = w[j - 16] + s0 + w[j - 7] + s1;
},
```

where w[16…63] – another 48 words, Mathem.Rotr() – function which performs a cyclic shift right, ">>" – logical shift right, '^' – logical adding, '+' – arithmetic adding.

In this part one can parallelly compute s0 and s1, since they are independent of each other. So for fast hardware implementation of s0 and s1 are needed two elements that will implement a cyclic shift right. For hardware implementation of entire program code above – four elements that will operate in parallel. However, for example, in following (eqations (1)-(5)) nonlinear bitwise functions of RIPEMD-160 algorithm parallel execution of all functions is not possible.

$$f(j; x; y; z) = x \oplus y \oplus z \quad (0<=j<=15) \quad (1)$$

$$f(j; x; y; z) = (x \wedge y) \vee (\neg x \wedge z) \quad (16<=j<=31) \quad (2)$$

$$f(j; x; y; z) = (x \vee \neg y) \oplus z \quad (32<=j<=47) \quad (3)$$
$$f(j; x; y; z) = (x \wedge z) \vee (y \wedge \neg z) \quad (48<=j<=63) \quad (4)$$
$$f(j; x; y; z) = x \oplus (y \vee \neg z) \quad (64<=j<=79) \quad (5)$$

So only two elements of «NOT», for example, can be done. Thus during the first 16 iterations of the main loop of the algorithm RIPEMD-160 functions (1) and (5) can be simultaneously executed. There equations (1) and (5) contain three operations «XOR», one operation «NOT» and one «OR». Because two operations «XOR» in equation (1) can be executed by one logical element, it is advisable to talk about a need to execute «XOR» operations only twice at this stage of the algorithm.

Types of basic operations and a number of arithmetic/logic elements for their execution for each of the algorithms under consideration are shown in Table 1.

Based on information from Table 1 lets construct a diagram (Fig. 2), which concludes the basic operations and number of elements to implement them in the processor, which would perform each of the considered hashing algorithms.

As we can see from Fig. 2, for implementation of the hashing processor for SHA-2 algorithm it is needed to instantiate 13 adders, 10 cyclic shifters, 6 XOR and 5 AND logic elements, and one inverter. For enhancement of device functionality and its enforcement with other algorithms – SHA-1, MD5 and RYPEMD, Keccak we must apply additionally 4 leftwise cyclic shifters, 2 OR logic elements, one inverter and one XOR. Multiplication in the MD5 algorithm can be successfully replaced with addition, as not all adders that are available in the device will be involved.

TABLE 1

TYPES OF BASIC OPERATIONS OF HASHING ALGORITHMS AND A NUMBER OF ARITHMETIC/LOGIC ELEMENTS FOR THEIR EXECUTION

HASHING ALGORITHMS	BASIC OPERATION								
	cyclic shift left	cyclic shift right	multiplication	logical shift right	OR	XOR	Addition	AND	NOT
SHA-1	3	0	0	2	2	1	6	3	1
SHA-2	0	10	0	2	0	6	13	5	1
MD5	1	0	1	0	1	1	7	2	1
RIPEMD 160/320	4	0	0	0	2	2	9	4	2
RIPEMD-256	2	0	0	0	2	2	10	2	1
Keccak	0	2	0	0	0	7	0	1	1

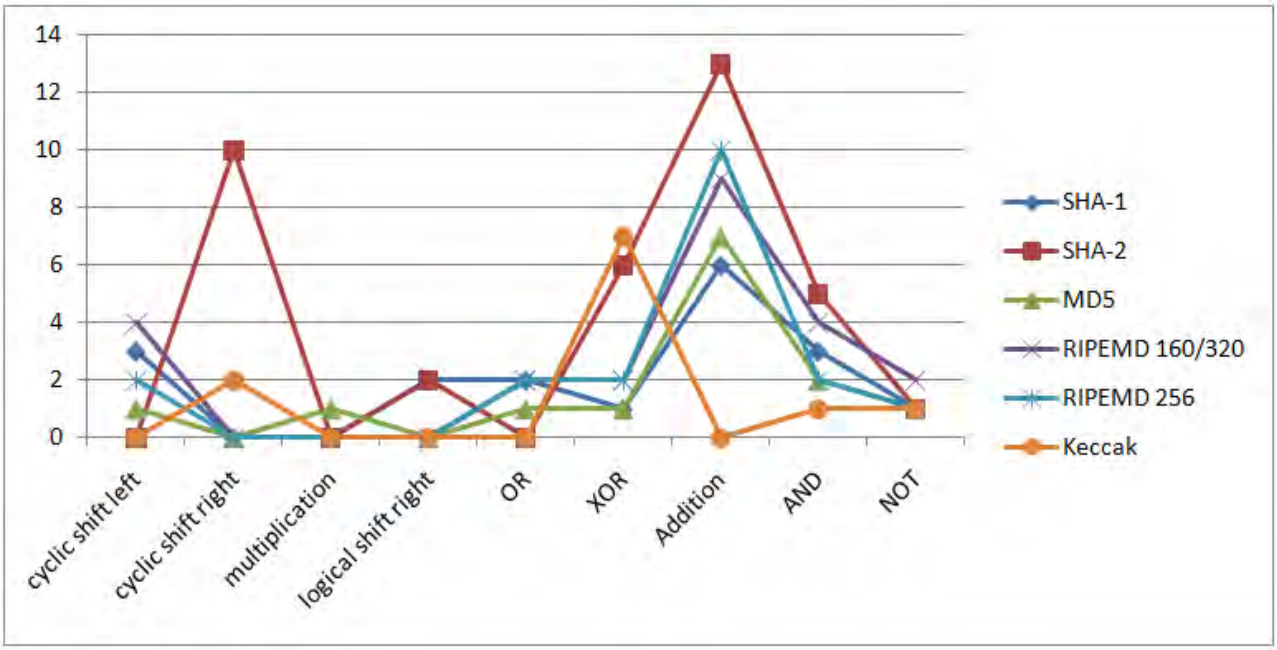

Fig. 2. Basic operations and number of elements to be implemented in hashing processors for SHA-1, SHA-2, MD-5, and RIPEMD algorithms execution

Conclusion

The paper considers the types of hashing algorithms and basic operations of most widely used hashing algorithms built "from scratch". The analysis of these algorithms allowed to determine basic operations, a number of the operations that can simultaneously be performed by digital elements, and a number of these elements needed to be implemented in hashing processors for algorithms SHA-1, SHA-256/512, MD-5, RIPEMD-160/256/320, and Keccak execution. The results of the analysis can be used for development of high-performance multifunctional hashing processors.

References

[1] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code in C", 2nd ed, Canada: John Wiley & Sons, Inc., 1996.

[2] National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES) (FIPS PUB 197), November 26, 2001.

[3] S. Miyaguchi, K. Ohta, and M. Iwata: "128-bit hash function (N-hash)", NTT Review, 2(6), November 1990, pp. 128–132.

[4] R. Shirey, “Internet Security Glossary, Version 2”, RFC 4949, August 2007, p. 35.

[5] J. Katz, Y. Lindell, „Introduction to modern Cryptography“, USA: Chapman and Hall/CRC, August 2007.

[6] V. Aurora, " Lifetimes of cryptographic hash functions", http://valerieaurora.org [Online]. Available: http://valerieaurora.org/hash.html [Accessed: Oct. 1, 2013].

[7] R. Rivest, “The MD5 Message-Digest Algorithm”, RFC 1321, April 1992.

[8] D. Eastlake 3rd, P. Jones, “US Secure Hash Algorithm 1 (SHA1)”, RFC 3174 , September 2001

[9] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, “The Keccak reference”, http://keccak.noekeon.org/, January 14, 21 [Online]. Available: http://keccak.noekeon.org/Keccak-reference-3.0.pdf. [Accessed: Oct. 19, 2013].

[10] The National Institute of Standards and Technology (NIST). “NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition”, http://www.nist.gov. [Online]. Available: http://www.nist.gov/itl/csd/sha-100212.cfm. [Accessed: Oct. 19, 2013].

[11] National Institute of Standards and Technology (NIST). FIPS-180-2: Secure Hash Standard, August 2002.

[12] B. Preneel, H. Dobbertin, A. Bosselaers, “The Cryptographic Hash Function RIPEMD-160”, CryptoBytes, Volume 3, No. 2, pp. 9 - 14, Autumn 1997. [Online]. Available: ftp://ftp.rsasecurity.com/ pub/cryptobytes/ crypto3n2.pdf. [Accessed: Oct. 1, 2013].