Proceedings of Informing Science & IT Education Conference (InSITE) 2015

Cite as: Ojo, S. A, Thompson, A. F., Iyare, O., & Alese, B. K. (2015). On information integrity measurement with secure hash algorithm (SHA). *Proceedings of Informing Science & IT Education Conference (InSITE) 2015,* 327-338. Retrieved from http://Proceedings.InformingScience.org/InSITE2015/InSITE15p327-338Ojo1550.pdf

On Information Integrity Measurement with Secure Hash Algorithm (SHA)

S. A. Ojo, A. F. Thompson, O. Iyare, and B. K. Alese
Computer Science Department, Federal University of Technology, Akure

shegun21@yahoo.com, afthompson@futa.edu.ng, oiyare@futa.edu.ng, bkalese@futa.edu.ng

Abstract

The "information age" as often referred to the modern society, has become heavily dependent on information systems. As this dependency increases, the threat to information security has also gained ground. Societies need to cater for the security of information, and this has led to the development of different information security techniques most notable of which is cryptography. Cryptographic Hash functions are used to achieve a number of security goals like authenticity, digital signatures, pseudo-random number generation, digital steganography, digital time stamping. The strength of a cryptographic hash function can be summarized into its vulnerability to attack and computational time. This work therefore, reviews existing standard cryptographic hash functions, their construction and their application areas. The secured hash function (SHA) was selected and implemented based on its comparative worth over others. The implemented cryptographic hash function is evaluated for performance using a cryptographic evaluation standard.

Keywords: Cryptography, Attack, SHA, Steganography, Information Security, Authentication.

Introduction

Nations, Organizations and Individuals have always held information at high regards. The value placed on information is no exception in the modern society. The information age as the modern society is synonymously referred to; has come to be dependent on information systems more than ever. The core of these systems being the information they process. As this dependency increases, it is not surprising that the threat to information security has also gained ground. It is of note that threat to information security is not peculiar to the information age. Societies have always catered for the security of information over the centuries. This has led to the development of different information security techniques, notable among them is cryptography. Information security refers to the precautions taken to keep information from unauthorized access, modification and use. As our society becomes more networked than ever, information becomes more vulnerable to these security threats than ever. The core dimensions of information security being authenticity, confidentiality and integrity. Authentication is the proof of identity or ownership of information. Confidentiality is ensuring that only authorized users

Editor: Eli Cohen

have access to information. Integrity is the assurance that a piece of information has not been altered or changed without authorization.

Historically, the security of information was achieved by a combination of physical security techniques and trust. Seals and signatures were proof of authenticity, this is evidenced in the royal seals and signets by past emperors. Confidentially was achieved by closing up documents and lockers. Integrity of information was mainly dependent on trust. Dedicated manpower such as the royal postal service was entrusted with the task of transmitting information. The degree of information security was dependent on the difficulty encountered in compromising these security techniques and breaching trust. In the information age, though the demand for information security still remains the same. Many of these physical security techniques are not only insufficient, but many are inapplicable. The networked age has increased the risk of information security. Unprotected data residing on open and un-trusted networks (e.g. the Internet) can be easily accessed, copied or modified (Kahn, 1996). In order to counter the security threat in the modern age, a prominent technique that has evolved is encryption. The science of encryption is called cryptography. Cryptography has always existed in varying forms over centuries. As confidentiality of information is a core tenet in military and diplomatic affairs, it is not surprising that cryptographic techniques have been used for many centuries to protect military and diplomatic information. Most of these techniques are encryption schemes that convert a message into a cryptogram by an invertible operation (encryption) depending on a small piece of secret information (the key). The cryptogram is unintelligible for an unauthorized person who intercepts it, but can be reconverted (decrypted) into the message by an authorized receiver who has been given knowledge of the key (Kahn, 1996).

The use of cryptography in information technology is dependent on cryptographic hash functions. A hash function is a function that takes a relatively arbitrary amount of input and produces an output of fixed size (John Edward, 2003). This property makes them useful in data structure, checksum algorithms for error detection, digital signature in information security etc. Cryptographic hash functions have another property that is beyond hash functions - it is very difficult to find two different inputs that produces the same output. This property provides a high level of certainty though not absolute that different input values would produce different output signature in most of the cases. This make cryptographic hash functions the hash functions that are used in information security related applications.

Related Works

According to Sobti and Geetha (2012), Cryptographic Hash functions are one of the most important tool in the field of cryptography and are used to achieve a number of security goals. In Wikipedia ("Information Security," 2014) the basic properties of the Hash function that enable them to withstand to a satisfactory level all known cryptanalytic attack was highlighted. These properties include:

- **Preimage resistance:** Which says that, given a hash **h** it should be difficult to find any message **m** such that **h = hash (m**). This concept is related to that of one-way function. Functions that lack this property are vulnerable to preimage attacks.
- **Second preimage resistance:** Given an input $\mathbf{m_1}$ it should be difficult to find another input $\mathbf{m_2}$ such that $\mathbf{m_1 \neq m_2}$ and **hash** $(\mathbf{m_1})$ = **hash** $(\mathbf{m_2})$. Functions that lack this property are vulnerable to second-preimage attacks.
- **Collision resistance:** It should be difficult to find two different messages $\mathbf{m_1}$ and $\mathbf{m_2}$ such that hash $(\mathbf{m_1})$ = **hash** $(\mathbf{m_2})$. Such a pair is called a cryptographic hash collision. This property is sometimes referred to as strong collision resistance.

Guauvravram (2003) referred to collision resistance property as collision freeness or strong collision resistance, second pre-image resistance as weak collision resistance and preimage resistance as one-wayness. Lai and Massey (1992) classified collision resistance as the strongest property of all three, hardest to satisfy and easiest to breach, and breaking it is the goal of most attacks on hash functions.

In Rogaway and Shrimpton (2014) the notion of the hash function security was extended. In this extension they defined seven different security notions, three based on pre-image resistance, three based on second pre-image resistance and one on collision resistance. Their work is based on generic concept of hash function family that is a finite set of hash functions with common domain and range.

Model Design

In this work, collision resistant hash function was used. The function 'h' satisfies the following conditions:

i. The argument X can be of arbitrary length and the result $h(X)$has a fixed length of n bits (with $n \geq 128$)
ii. The hash function must be one-way in the sense that given a Y in the image of h, it is "hard" to find a message X such that $h(X) = Y$, and given $X$and $h(X)$ it is "hard" to find a message $X' \neq X \exists\ h(X') = h(X)$
iii. The hash function must be collision resistant: this means that it is "hard" to find two distinct messages that hash to the same result that is, **x', x ∈ X, s.t. h(x') = h(x).** It should be difficult to find two different messages $m1$ and $m2$ such that $hash(m1) = hash(m2)$.

The first part of the second condition corresponds to the intuitive concept of one-wayness, that is, it is "hard" to find a pre-image of a given value in the range. The other part of this condition, is finding a second pre-image which requires at most 2^n operations. Thus, in order to improve on one-wayness of the collision resistance, the hash function with n-bit would be about $2^{n/2}$ operations with a birthday or square-root attack.

Furthermore, the hash functions used are based on compression function with fixed size input using iterated hash function. The information is divided into 't' blocks X_1 through X_t. If the total number of bits is not a multiple of the block length, the information has to be padded to the required length as:

$$H_0 = IV \quad (1)$$

where IV is the Initial value

$$H_i = f\ (X_i, H_{i-1})\ \forall\ i = 1, 2, \ldots t \quad (2)$$

$$h(X) = H_t \quad (3)$$

The result of the hash function is denoted with $h(X)$, the function f is the round function, while H_i is called the chaining variables. The choice of padding rule and initial value would be done to achieve an efficient secure hash function.

The SHA-1 Cryptographic Hash Function

The SHA algorithm was designed by NSA and published by NIST as federal standard FIPS 180 in 1993. SHA is another hash function inspired by MD4. Though not the most theoretically secure of all cryptographic hash functions, SHA-1 is used due to its simplicity, speed and architecture. SHA-1 is considered to be the successor to MD5.

The maximum message length of 2^{64}-1 is what can be accepted by an SHA-1 algorithm. This produces a 160-bit message digest as output. The compression function in 512-bit block processes the message. Each block is divided further into sixteen 32-bit words denoted by M_t for t = 0, 1, ---, 15. The compression function consists of four rounds, each round making up a sequence of twenty steps. A complete SHA-1 round consists of eighty steps where a block length of 512 bits is used together with a 160-bit chaining variable to finally produce a 160-bit hash value. The SHA-1 algorithm is designed to work in the following steps:

Step 1: Append padding bits

The original message is padded so that its length is congruent to 448modulo512. The padding is always added even if the message is already of the desired length. Padding consists of just a single 1 followed by the number of 0 bits necessary to make the original message even.

Step 2: Append length

A 64-bit block treated as an unsigned 64-bit integer (most significant byte first), and representing the length of the original message (before padding in step 1), is appended to the message. The entire message's length is now made a multiple of 512.

Step 3: Initialize the buffer

There exist a buffer consisting of five (5) registers of 32 bits are represented as A, B, C, D and E. This 160-bit buffer is used to hold temporary and final results of the compression function. These five registers are initialized to the following 32-bit integers (in hexadecimal notation). The first four registers in SHA-1 are exactly the same as the four registers used in MD5 algorithm. But in SHA-1, these values are stored in big-endian format, which means that the most significant byte of the word is placed in the low address byte position.

Step 4: Process message in 512-bit blocks

The compression function is divided into twenty sequential steps having four rounds of processing where each round is made up of twenty steps. The four rounds are structurally similar to one another with the only difference that each round uses a different Boolean function, which we refer to as f1, f2, f3, f4 and one of four different additive constants K_t (0 0 $\leq t \leq$ 79) which depends on the step under consideration.

Every step updates two of the five registers. The step operation which updates the value of the E register and rotates the value of the B register by 30 bit position to the left. This is represented in the following form:

A, B, C, D, E $\leftarrow$ (E + f_r (t, B, C, D) + [A <<< 5] + M_t + K_t), A, [B <<< 30], C, D where

A, B, C, D, E = the five registers of the SHA-1 buffer

t = the step number, $0 \leq t \leq 79$

f_r = the primitive logical function used in step t and round r

<<< s = the circular left shift of the 32-bit word by s bits

M_t = a 32-bit word derived from the current 512-bit input block

K_t = one of four additive constants

+ = addition modulo 2^{32}

Each Boolean function takes three 32-bit words as input and produces a 32-bit word as output.

Step 5: Output

After processing the last 512-bit message block t (assuming that the message is divided into t 512-bit blocks), we obtain a 160-bit message digest. The compression function uses a feed-forward operation where the chaining variable CV_k input to the first round is added to the output obtained after execution of step 80 to produce the next chaining variable CV_{k+1}. This addition is performed modulo 232 and independently for each of the five words in the buffer.

The word expansion technique introduced by SHA-1 augments the interdependency between every message block and the final message digest. In addition to the longer output of 160-bit message digest, SHA-1 simply strengthens the one-wayness, pre-image resistance, second pre-image resistance and collision resistance.

Implementation

The cryptographic system was designed, implemented and evaluated using Microsoft Visual Studio as the development environment, Microsoft C# as the programming language. Microsoft C# has a support library for the implementation of cryptographic hash functions on the windows operating system which shortened the project's development time. OpenSSL is used to evaluate the performance of the system's cryptographic hash function, the performance result in comparison with the established results of other hash functions and is analyzed with Microsoft Excel. To implement a hash function; there exist additional requirements to consider (aside the ones shown in figure 1), these requirements are:

- Speed
- Low Resource consumption

The designed hash function to be utilized in the cryptographic system must be able to execute as quickly as computationally possible without a compromise on it performance and speed. This is made possible by performing optimizations on such algorithmic processes of these functions.

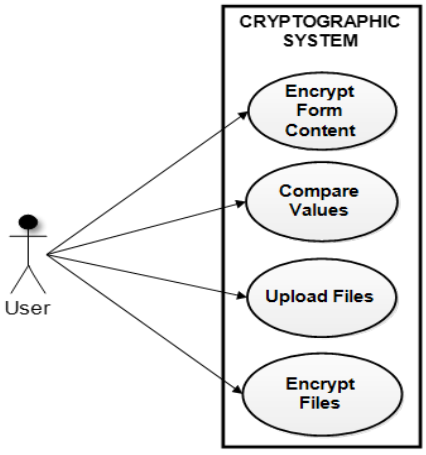

Figure 1: A use case diagram of the functional requirements of the cryptographic hash function

Results and Discussion

The application is designed to utilize a graphical user interface (GUI) based on the Microsoft windows traditional GUI elements. This user centered approach makes the system easy to use thereby hiding the complexity of the SHA-1 algorithm utilized in the implementation.

The strength of the model lies in the improvement of one-wayness of the collision resistance. The hash function with n-bit is about $2^{n/2}$ operations. This is shown in the time complexity analysis. The running time for the conventional hash function operations is:

$$T(n) = c_1 n^2 + c_2 n^2 + c_3(2^n - 1) + c_4 2^n$$
$$T(n) = (c_1 + c_2)n^2 + (c_3 - c_3 + c_4)2^n$$
$$T(n) = n^2 + 2^n$$
$$T(n) = log n^2 + log 2^n$$
$$T(n) = 2\log n + n log 2$$

While the enhanced hash function is:

$$T(n) = c_1 n^2 + c_2 n^2 + c_3(2^{n/2} - 1) + c_4 2^{n/2}$$
$$T(n) = (c_1 + c_2)n^2 + (c_3 - c_3 + c_4)2^{n/2}$$
$$T(n) = n^2 + 2^{n/2}$$
$$T(n) = log n^2 + log 2^{n/2}$$
$$T(n) = 2\log n + {}^{n}/_{2}\, log 2$$

Consequence upon the time complexity computation above, the enhanced hash function has an improved running time in comparison with the conventional.

As the input is typed into the input box, the encrypted output will be showing accordingly using SHA-1 encryption algorithm. If a character is added to the source input, the encrypted output changes entirely and is unique throughout the work for that particular input which cannot be duplicated with different input and the same Encrypted output.

$H(M_1) \neq H(M_2)$, if $M_1 \neq M_2$.

$H(M_1) = H(M_2)$ if $M_1 = M_2$

At the second part, the purpose of this part is to validate the conditions above; select and copy the encrypted copy from the first part for a particular input, right click and paste into the box named "*Encrypted Value*". Then enter a value to the real value part, at first enter the expected input string and press "Compare Values" button and also try with real value that is not equivalent to the previous input string inserted and do compare it again.

Performance Evaluation

Performance evaluation have been made with the standard command speed of the command open Secure Socket Layers (SSL). OpenSSL is a standard cryptographic toolkit implementing the secure socket layer (SSLv2/v3) and Transport Layer Security (TLS). The comparative evaluation on the MDx constructions is possible due to their implementations in the OpenSSL standard toolkit. The execution speed of this command gives the number of calculations the computer can make with a determined algorithm. The computer performs the calculations of the given command in a timeframe of three seconds. The result of the implemented SHA-1 algorithm in comparison to other dedicated cryptographic hash functions is given in table 1 and table 2 using a windows 7 Intel Pentium 2048MHz, 3GB RAM and Intel pentium IV 2048MHz, 1024Mb RAM respectively.

All measures are in 1000s of bytes per second processed. Presented are two tables (tables 1 and 2) showing the performance of different hash functions in two different machines.

Table 1: Performance of the cryptographic hash functions on Intel Pentium 2048MHz, 3GB RAM running Windows 7

HF/Block size	64 bytes	256 bytes	1024 bytes	8192 bytes
MD 2	1754.39k	2464.39k	2725.55k	2818.05k
MD 4	26959.42k	77394.81k	145022.45k	193320.28k
MD 5	22334.63k	66966.36k	131811.67k	184606.72k
HMAC (MD5)	14093.50k	46245.97k	108043.95k	177919.32k
SHA-1	**20952.16k**	**51399.42k**	**82104.32k**	**99304.03k**
RIPEMD 160	17604.76k	39678.38k	57750.87k	66701.99k

Table 2: Performance of the cryptographic hash functions on Pentium IV 2048MHz, 1024Mb RAM running Window 7

HF/Block size	64 bytes	256 bytes	1024 bytes	8192 bytes
MD2	2945.83k	4026.75k	4389.94k	4598.09k
MD4	36987.28k	98147.80k	160098.78k	193180.01k
MD5	38200.07k	104595.93k	184259.52k	233499.77k
HMAC (MD5)	21001.75k	68866.59k	151759.20K	227412.65K
SHA-1	**22561.54k**	**52878.08k**	**81529.17k**	**97498.45k**
RIPEMD 160	22330.28k	45506.56k	64100.69k	72015.87k

Discussion of Results

We can observe here that MD2 is a very slow algorithm, even though it belongs to the MD family; it has a very different algorithm for its computation. An important observation is that MD5 is faster (in table 2 only) than the obsolete (and already broken) MD4. Also, it turns that the difference of speed is bigger as the algorithms work on bigger blocks. So, normally, MD5 is much faster than MD4 when computing the digest of big messages (think of files for example).

The line charts shown in figure 2 and figure 3 reveal an interesting observation, while all algorithms get a better performance in the faster machine (i.e. Intel Pentium 2048MHz, 3GB RAM), SHA-1 gets similar performance. This means that if we get to have much better machines, RIPEMD160 (goes faster on faster machines) would get to be as fast as SHA-1, because it seems its performance does not depend significantly on the speed of the machine.

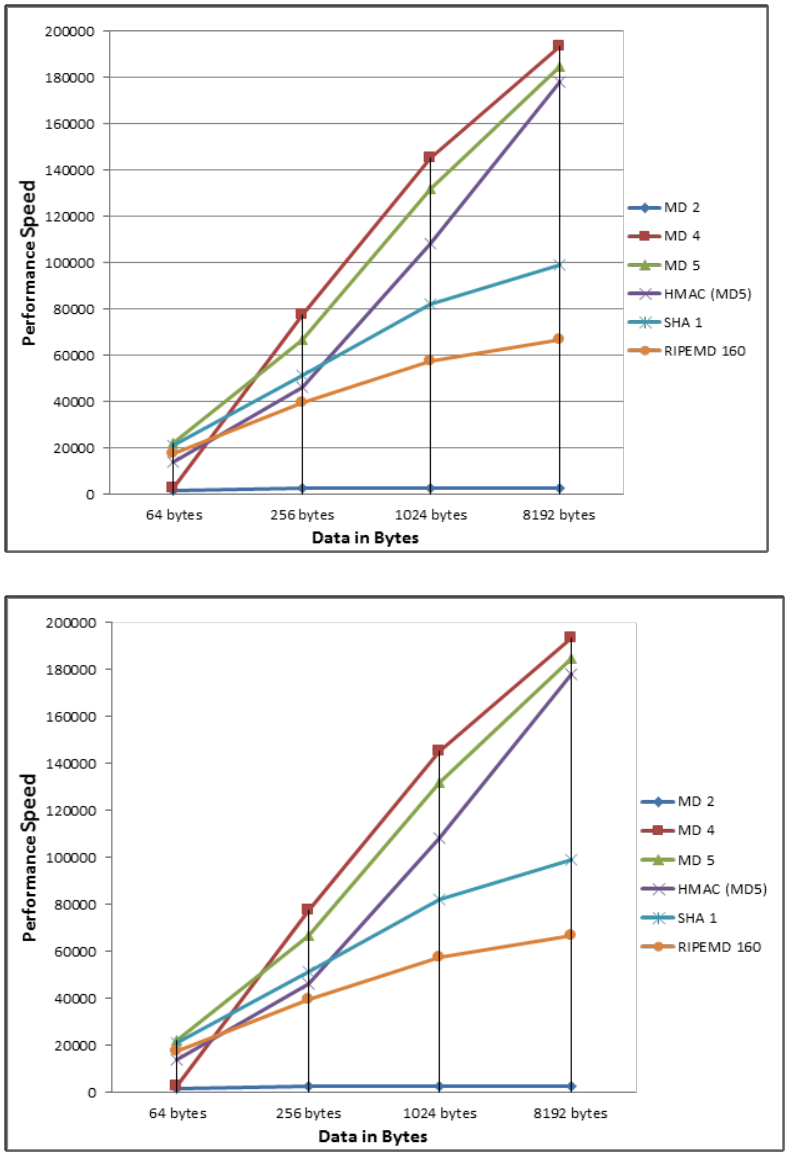

Figure 2: Progressive performance of the cryptographic hash functions on Intel Pentium2048MHz, 3GB RAM running Windows 7

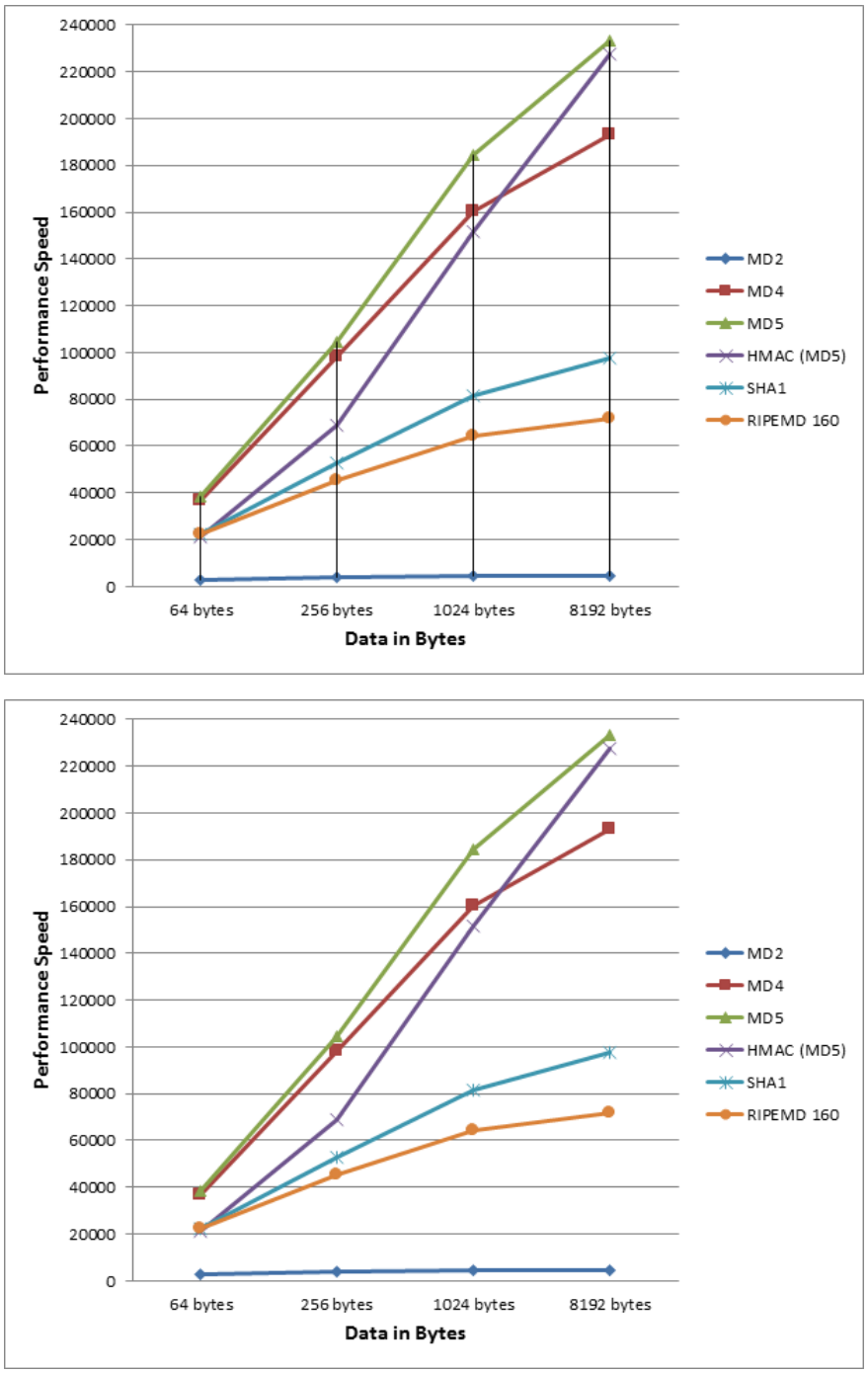

Figure 3: Progressive performance of the cryptographic hash functions on Pentium IV 2048MHz, 1024Mb RAM running Window 7

The results confirm a proven knowledge that the RIPEMD-160 and SHA-1 seems to be the most interesting hash functions if an output length of 160 bits is sufficient. For larger output lengths the recent hash functions of the SHA family can be used, although they should receive more extensive public cryptanalysis to find vulnerabilities beyond theoretical. The SHA-1 is the dominantly used cryptographic hash functions because it takes the most effort to develop an attack to threaten it practical application despite their relative simplicity.

Comparative Analysis of the Mdx-Class Hash Functions

From the reviewed literatures on the implementation of cryptographic hash functions, the observations made on the different hash functions from the MDx-class had being summarized in the tables 3, 4 and 5. Table 3 compares the structure of the different algorithms, and table 4.4summarizes the most important attacks that reveals their weaknesses. A summarized result of literature that compares the software performance of most of the algorithms in Table 5. They lead to comparable results. The early proposals of the MDx-class are more efficient than the more recent proposals.

Table 3: Structure of Cryptography hash functions (lengths are in bits)

Algorithm	Output length	Block length	Word length	Number of steps
MD4	128	512	32	3 X 16
MD5	128	512	32	4 X 16
HAVAL	128-256	1024	32	3, 4, or 5 X 32
RIPEMD	128	512	32	3 X 16 || 3 X 16
RIPEMD 128	128	512	32	4 X 16 || 4 X 16
RIPEMD 160	160	512	32	5 X 16 || 5 x 16
SHA	160	512	32	4 x 20
SHA-1	160	512	32	4 x 20
SHA 224/256	224/256	512	32	1 x 64
SHA 384/512	384/512	1024	64	1 x 80

Table 4: Known attacks for Cryptography hash functions

Algorithm	Weakness
MD4	Collisions (also pre-images for two-round reduced versions)
MD5	Pseudo collisions and collision for compression function
HAVAL	Collisions for three round version
RIPEMD	Collision for two round reduced versions
SHA	Theoretical collision attack
SHA-1	Slide attack

Table 5: Software performance of Cryptography hash functions.

Algorithm	Cycles/ byte	Relative performance
MD4	4.7	1
MD5	7.2	0.65
RIPEMD 160	18	0.26
SHA	15	0.31
SHA-1	15	0.31
SHA 224/256	39	0.12
SHA 384/512	83	0.06

This comparison affirms the balance of the SHA-1 cryptographic hash functions across the evaluation criteria used in the tables. This makes it no surprising the dominant implementation of the SHA-1 algorithm.

Conclusion

Cryptographic hash functions which are the basic building blocks for computer cryptography were the focus of this work. This work reveals hash functions as important and versatile cryptographic building blocks, their constructions, weaknesses and applications are reviewed. Their strength in relation to other important cryptographic tools such as block ciphers and pseudorandom function were also presented. The standard and widely used dedicated hash functions follow the design principle of Merkle-Damgard iterated hash function construction. The SHA-1 cryptographic hash function which is the most recommended in reviewed literatures and real life application due to the balance between it security and performance. The SHA-1 based cryptographic application is implemented with C-Sharp and evaluated using the OpenSSL command toolkit. The performance evaluation of the application affirms the reasons why the SHA-1 cryptographic hash function is widely in use.

The achievement of the aim and associated objectives of the work presented reappraises the relevance of cryptographic hash functions. The dependence of computer cryptography on cryptographic hash functions reveals that the continual relevance of cryptography as an information security approach is dependent on how far we can go in improving cryptographic hash functions, the building block of cryptography.

In the course of conducting this research work, we found out that many of the design and evaluation of these cryptographic hash functions can be complex and resource consuming in time, computational effort and power. Based on the knowledge and experience gained on the work it is recommended that new cryptographic hash functions should be developed by first evaluating the maximum possible extensions that can be made on existing ones before developing from scratch. This reduces the time and effort resources invested in developing new cryptographic hash functions.

References

Gauravram, P. (2003). *Cryptographic hash functions: Cryptanalysis, design and applications*. Ph.D. thesis, Faculty of Information Technology, Queensland University of Technology, Brisbane, Australia.

Information Security. (2014). In Wikipedia. Retrieved from http://en.wikipedia.org/wiki/Information_security

Kahn, D. (1996). The Codebreakers: The comprehensive history of secret communication from ancient times to the Internet (Revised edition). Scribner.

Lai, X., & Massey, J. L. (1992). *Hash function based on block ciphers.* In EUROCRYPT, 1992, pp.55-70.

Matyas, S., Meyer, C., & Oseas, J. (1985). Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, *27*(10A), 5658–5659.

Rogaway, P., & Shrimpton, T. (2004). *Cryptographic hash function basics: Definitions, implications and separations for preimage resistance, second preimage resistance, and collision resistance*. inFSE, pp.371-388.

Silva, J. E. (2003). *An overview of cryptographic hash functions and their uses*. GIAC Security Essentials Practical Version 1.4b Option.

Sobti, R., & Geetha, G. (2012). Cryptographic hash functions: A review. *IJCSI International Journal of Computer Science Issues, 9*(2), 461-479.

Biographies

S. A. Ojo [information not available]

Dr. (Mrs) A. F. Thompson is a lecturer I in the department of Computer Science, Federal University of Technology, Akure. Her areas of research are Computer and network security, Biometric computing.

Miss O. Iyare is a lecturer II in the department of Computer Science, Federal University of Technology, Akure. Her areas of research are Computer and network security, softcomputing.

Dr. B. K. Alese is an associate professor in the department of Computer Science, Federal University of Technology, Akure. His areas of research are Computer and network security, Quantum computing, Digital Signal Processing and Fault Tolerant Computing. He is the current Chair Occupant of the First Bank Professorial Chair in Computer Science, Federal University of Tecchnology, Akure