New Hash Function Based on Chaos Theory (CHA-1)

Mahmoud Maqableh, Azman Bin Samsudin and Mohammad A. Alia

School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia

Summary
Cryptographic hash function has been used extensively in many cryptographic protocols. Many of the hash functions generate the message digest thru a randomizing process of the original message. Subsequently a chaos system also generates random behavior, but at the same time a chaos system is completely deterministic. In this paper, we propose a new hash function (CHA-1) based on chaos, which produces 160-bit hash digest, accepts message length less than 2^{80} bits, and has a security factor 2^{80} of brute-force attack.

Key words:
Hash Functions, SHA-1, Chaos Theory, Logistic Map and Collision Search Attacks.

1. Introduction

Secure Hash Algorithm (SHA-1) was issued by the National Institute of Standards and Technology in 1995 as a FIPS [1]. Hash functions such as SHA-1 are normally used as data integrity primitive in more complicated cryptographic protocols. SHA-1 is one of the most popular hash functions, which accepts message with length less than 2^{64} bits and generates 160-bit message digest. SHA-1 has been adopted by many institutions, deployed as an important component in various cryptographic schemes, as well as being implemented in most commercial security systems and products.

Lately there has been significant advancement in the analysis of hash function. In February 2005 SHA-1, has been compromised when a new way of finding collisions in SHA-1 was discovered [2]. From the report, collisions in SHA-1 can be found with less than 2^{69} hash operations. Later, in August 2005, another group of cryptanalysis scientists discovered an improved attack on SHA-1; this time the complexity of the new attack is claimed to be 2^{63} [3].

Many studies on digital chaotic cryptographic have been proposed [4-8]. Chaos system has attracted much attention in the field of cryptography due to its properties such as deterministic and sensitive to the initial values. As indicated earlier in previous paragraphs, researchers are urgently looking for new hash function that might be able to replace SHA-1. In this paper, we are proposing a new hash function, Chaos Hash Algorithm 1 (CHA-1), which is based on chaos theory. CHA-1 accepts message with length less than 2^{80} bits and produces unique message digest of length 160-bit.

The new proposed hash function appears to have benefited from the chaotic nature of the chaos system. One glaring benefit of chaos based crypto primitives is that the ordinary cryptanalysis methods are not applicable anymore. The current state of the art in cryptanalysis employs many different methods such as statistical analysis and brute force exhaustive search. However, such cryptanalysis methods cannot be applied to chaos cryptosystem, because of the random nature and the unpredictability of chaos functions. Brute force will not work either if the chaos based crypto primitives employ a huge key space in its implementation.

2. Logistic Map

Hash function is almost similar to a pseudo random number generator in terms of the randomized output required. Therefore, a dynamic system like chaos is very suitable to be used as the engine for a hash function. However, not all chaotic maps are suitable for cryptographic purposes. Among the promising chaotic maps that can be used for cryptographic purposes is the Logistic Map which we had used in our implementation.

Logistic Map is simple, fast, sensitive to the initial conditions, unpredictable and it is a one-way-function. Logistic Map is a recursive function which takes a real number (X_n), $0 \le X_n \le 1$ as an input, and produce a real number, X_{n+1}, between *0* and *1* (inclusive) as indicated by Equation 1. Various sequences of the Logistic Map can be generated from different initial values of X_0 and r. Figure 1 shows the graph of the Logistic Map plotted from Equation 1.

$$X_n = r(1 - X_{n-1})X_{n-1}; \quad \text{where } r \in [0,4], X_n \in (0,1), \text{and } n \in \mathbb{N} \tag{1}$$

Manuscript received February 5, 2008
Manuscript revised February 20, 2008

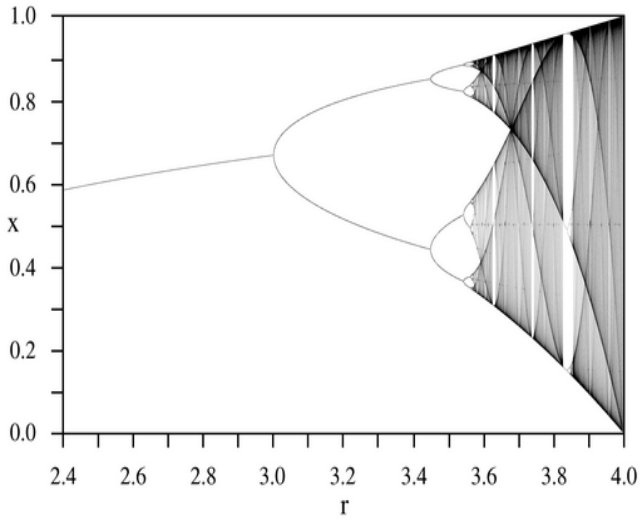

Figure 1: Bifurcation diagram for the Logistic Map [9]

3. Proposed Solution

Figure 2 briefly illustrates the general outline of CHA-1. Chaos Logistic Map is used as the core of CHA-1 at two phases, in Phase 3 and Phase 4. As in other chaos systems, apart from sensitive to the initial conditions, the Logistic Map is also influenced by the recursive parameters (X_n and r). This property has been exploited in the design of CHA-1. As shown in Figure 2, the message digest produced by the Logistic Map in Phase 4 will be heavily influenced by the original message (X_n, in terms of Logistic Map) and the initial conditions (X_0 and r, in terms of Logistic Map) which were derived from another Logistic Map in Phase 3. This Section will describe further the detail design of each phase of CHA-1.

3.1 Chaos Hash Algorithm-1 (CHA-1)

As shown in Figure 2, CHA-1 has four main parts: *R*-function, X_0-function, Logistic Map equation that takes the parameters from both the *R*-function and X_0-function, and another Logistic Map equation which takes the message as the input parameter to produce the final message digest. CHA-1 accepts message with length less than 2^{80} bits and processes each 160-bit block each time from the original document as inputs to the X_0-function and *R*-function. The first function of CHA-1 is X_0-function, which is designed to produce 160-bit output to be use as an input to the Logistic Map in Phase 3 as the initial value X_0. The second function of CHA-1 is the *R*-function, which is designed to produce the incremental value r, where r is the second parameter for the Phase 3 Logistic Map. The third Phase of CHA-1 is the process of finding a real value between 0.33 and 0.59 which is the chaos region of the Logistic Map. In Phase 3 the initial value, X_0, and the initial value, r, will be used as the input parameters to the Logistic Map. CHA-1 takes the first 64-bit from the result of *R*-function to be incremented with the value of r to calculate the next value of X (X_{n+1}) according to Equation 1. In Phase 4, CHA-1 uses the result from Phase 3 as the initial value, and recursively producing X_n by using 64-bit value at a time from the original message.

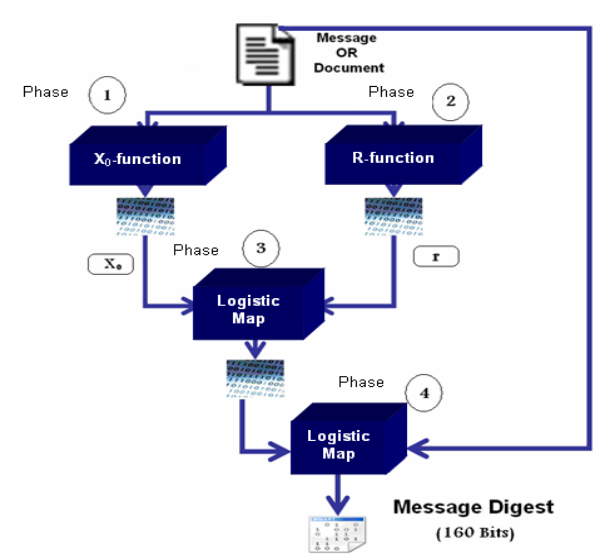

Figure 2: Overview of CHA-1

3.2 X_0 - Function

Logistic Map is sensitive to the initial conditions. Any changes to the initial values will produce totally different output as shown in Figure 1. Therefore, in designing CHA-1 we use two additional functions (X_0-function and *R*-function) which will produce the initial value of X_0 and the incremental value of r to the Logistic Map in Phase 3 from the information of the original message. As shown in Figure 2 the X_0-function processes the input message and produces the X_0 value which later will be used as the initial parameter for the Logistic Map in Phase 3. X_0-function is a simple function which uses three simple non-commutative operations: XOR, addition mod 2^{32}, and 32-bit left circular shift. Message will be divided into 160-bit blocks and each block is further divided into five, 32-bit buffers, namely A, B, C, D, and E. X_0-function processes 160-bit block each time from the input message, as illustrated by Figure 3.

After processing a block of message the X_0-function produces a 160-bit value as the output. This 160-bit value

is then will be XORed with the next 160-bit block message shown as A`, B`, C`, D`, E` at the bottom of Figure 3. The result of the XOR operation will be used as the input to the next iteration of the X_0-function. This process will continue until the function finishes processing the whole message. If needed, the end of the last block will be padded by '1' followed by '0' to complete the last 160-bit block. The final result of X_0-function will be converted to a real value between 0.33 and 0.59, which will become the initial value of the Logistic Map in Phase 3.

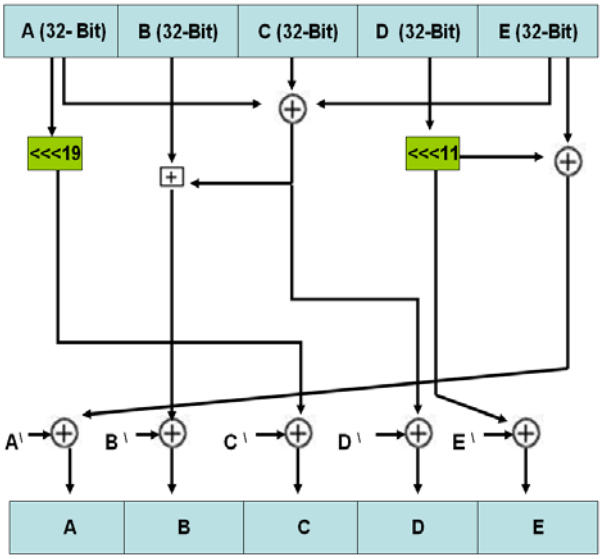

Figure 3: X_0-function

The purpose of having Phase 1 and 2 as described in Section 3.2 ad 3.3, is to produce high quality initial values. The output of X_0-function is then converted to a value between 0.33 and 0.59 which then will be used as the initial value of X_0 for the Logistic Map found in Phase 3. The first 64-bit from the *R*-function is converted to a 160 bits real number value between 0 and 1. We use this value as the incremental value *r* in Logistic Map (Phase 3) as well as the seed to the next 64-bit value. The result of Phase 3 will then be used in the fourth Phase as shown in Figure 5.

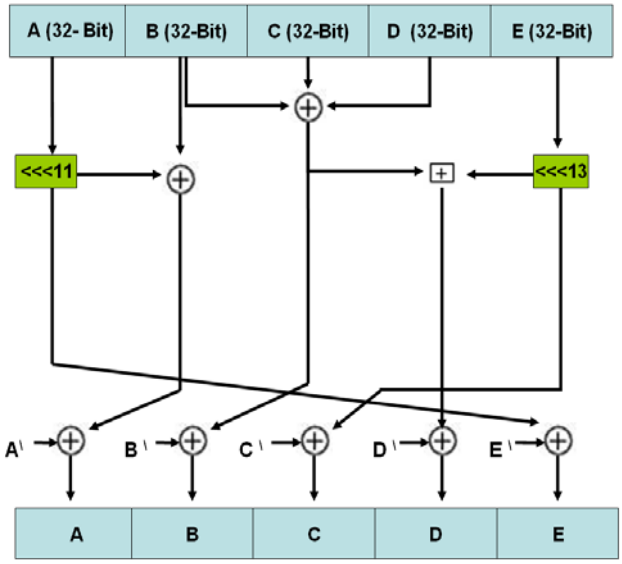

Figure 4: *R*-function

3.3 *R*-Function

Similar to the X_0-function, the *R*-function processes 160-bit block from the input message each time. The output of the *R*-function is a 160-bit block value. This value will be XORed with the next 160-bit from the original input as shown in Figure 4. The result of the previous operation is taken as the input to the *R*-function of the next iteration. After all the message blocks being processed, the result will be a 160-bit value, this value will be used as the incremental value *r* in the Logistic Map equation in Phase 3. The process is repeated until the message is exhausted. The *R*-function is a simple function which uses three non-commutative operations, similar to the X_0-function.

3.4 The Fourth Phase of CHA-1

After we apply Logistic Map in Phase 3, we have the 160-bit value as the output. This value will be used as the initial value for the Logistic Map in Phase 4. In Phase 4 we take 64-bits from the original message (each time) to be used as the incremental value *r*. This process is repeated with 64-bit value at a time until the message is exhausted. At the end, one 160-bit value which will be a point in the bifurcation diagram is used as the hash value (message digest) for the message.

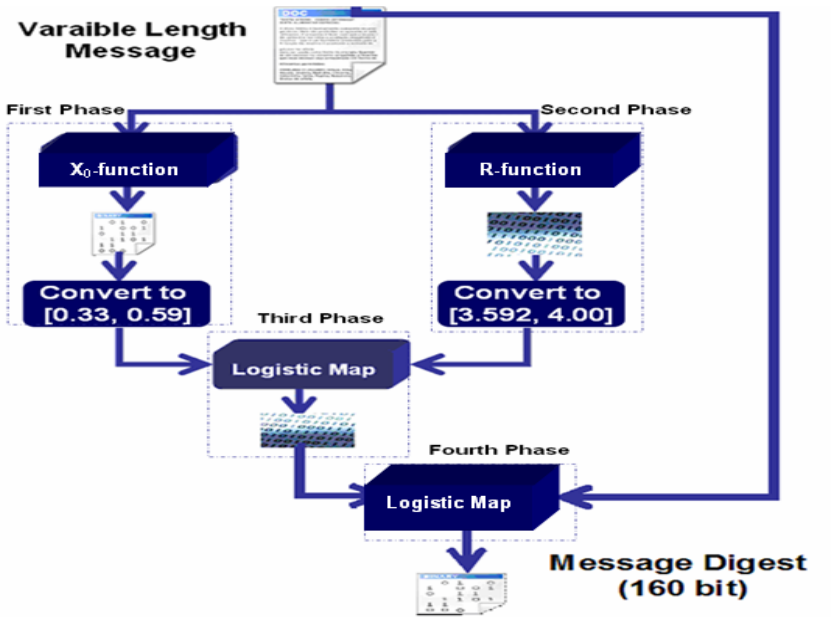

Figure 5: Detail Framework of CHA-1

4. Results and Discussion

In this Section, we highlight some of the experimental results based on different input messages. In addition, we also make some comparison between SHA-1 and CHA-1 in terms of the execution time. We use the same set of data as the input message, and calculate the time taken by both hash functions. We implemented CHA-1 using C Language and GNU Multiple Precision Arithmetic Library (GMP-4.2.1) [10]. The comparison between SHA-1 and CHA-1 was done on a PC with 2.4 GHz Celeron(R), 200 MB memory and 40 GB hard-disk capacities.

Figure 6 shows the comparison between SHA-1 and CHA-1 in terms of the execution time. We start the test with 10 KB of data as the input to both algorithms. For large data set (larger than 90 KB), the performance of SHA-1 is always better than the performance of CHA-1 as shown in Figure 6. When size of the input message was about 1000 KB, we can see SHA-1 is about three time faster than CHA-1.

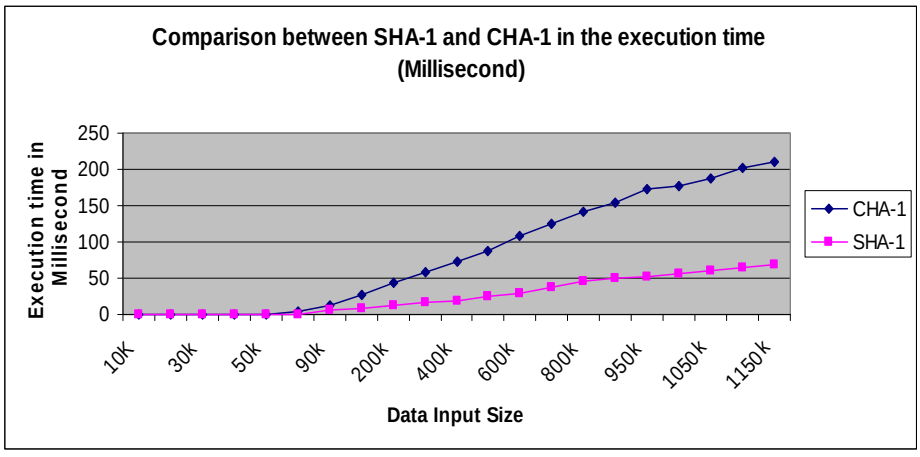

Figure 6: Comparison between SHA-1 and CHA-1 in the execution time (millisecond)

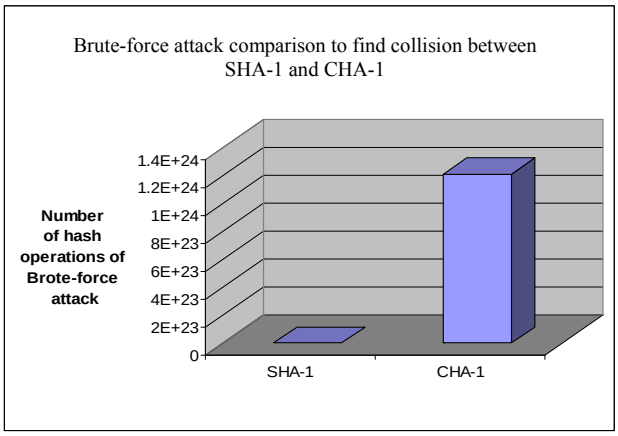

Figure 7: Brute-force attack comparison between SHA-1 and CHA-1

Even though CHA-1 is slower, CHA-1 is better than SHA-1 in terms of security. As mentioned earlier, with the advance in cryptanalysis the security factor for SHA-1 has gone down to only 2^{63} (hash operation-brute-force) while the security factor for CHA-1 is still 2^{80} (hash operations brute-force). CHA-1 can also accept any message less than 2^{80} bits, compared to SHA-1 which can only accept message size less than 2^{64} bit. The qualitative analysis between SHA-1 and CHA-1 is further described by Table 1.

Table 1: Comparison between SHA-1 and CHA-1 Properties

Properties	SHA-1 (bits)	CHA-1 (bits)
Message size	2^{64}	2^{80}
Block size	512	160
Word size	32	32
Message digest size	160	160
Security	2^{63}	2^{80}

4.1 Randomness Test

Five different tests were conducted for both algorithms (SHA-1, CHA-1) on five different data sets (1 KB, 2 KB, 3 KB, 4 KB, 5 KB), to analyze the randomness of the hash value produced. First, we take the input message with all ones, and we record the hash value (control sample). Then we change one bit in the input message from '1' to '0' and we examine how many bits got affected by the change, by comparing the result of the hash value with the result of the control. We perform the test 512 times with 512 test data by changing the i^{th} bit of the input message (refer to Figure 8). Finally, we calculate the average and the standard deviation for both algorithms. The result is tabled out in Table 2 and plotted in Figure 9.

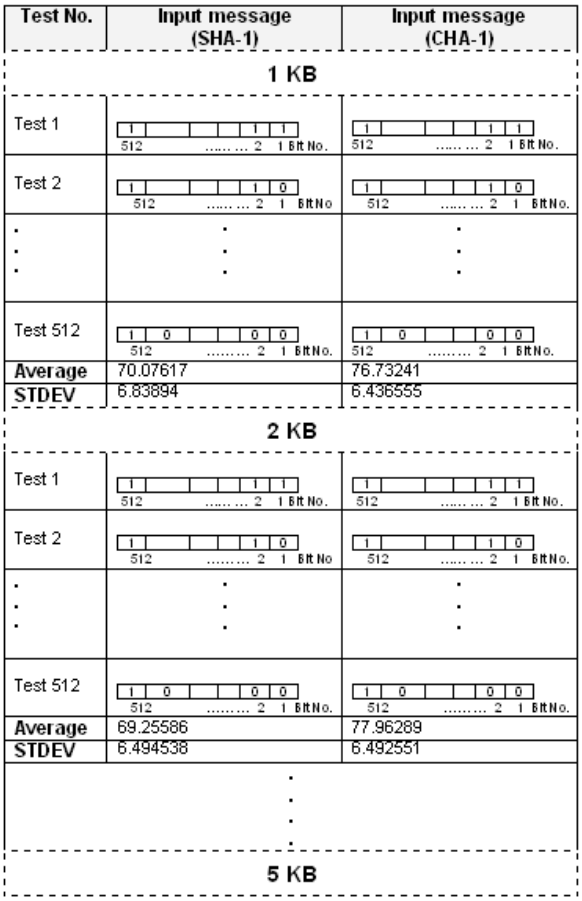

Figure 8: Randomness test design

Table 2: Comparison between SHA-1 and CHA-1 in bit randomness test

Message Input Size	SHA-1		CHA-1	
	Average	STDEV	Average	STDEV
1 KB	70.07617	6.83894	76.73241	6.436555
2 KB	69.25586	6.494538	77.96289	6.492551
3 KB	70.43164	6.321090	77.55664	6.185704
4 KB	69.57227	6.924718	77.5957	6.721073
5 KB	69.96875	6.850872	77.08203	5.877842

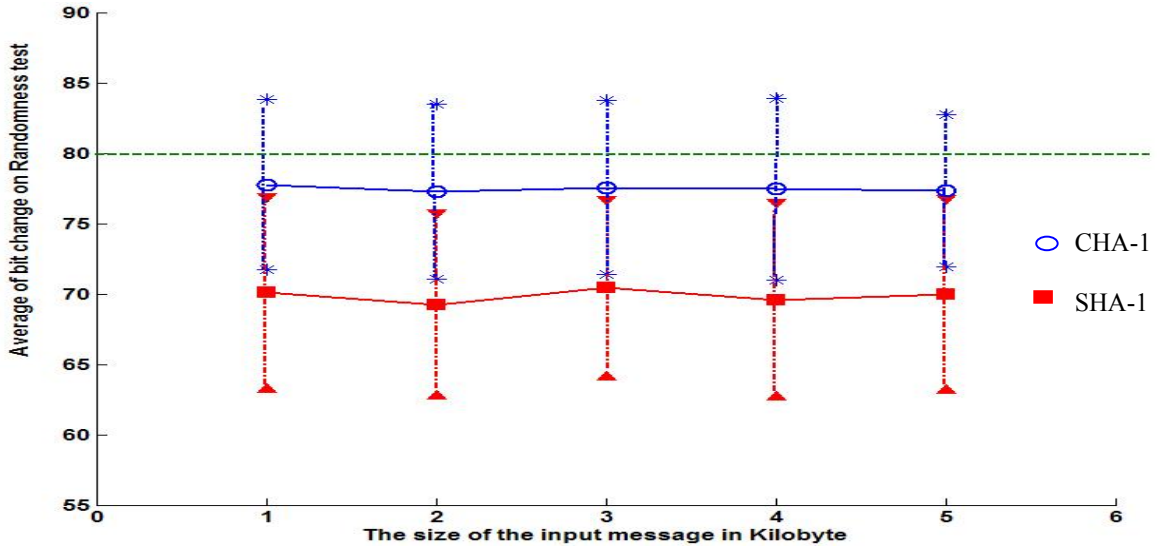

Figure 9: Comparison between SHA-1 and CHA-1 in the bit random test

In the first test, SHA-1 randomness test average at 70.07617 with standard deviation 6.83894, while CHA-1 average at 77.46875 with standard deviation 5.032019. The test shows the average of CHA-1 was better than the average of SHA-1 and the standard deviation of CHA-1 is also better than SHA-1 (the average of CHA-1 was close to 80 and the CHA-1 standard deviation was less than of SHA-1 standard deviation). Other tests on the other data sets produce similar scenarios as well. Therefore, generally we can conclude that CHA-1 produces a randomize message digest which is at least as randomize as SHA-1 message digest.

5. Conclusion

In this paper, we proposed a new hash function (CHA-1) based on the Logistic Map of the chaos theory. Logistic Map function generates random output but at the same time it is completely deterministic and sensitive to initial values which highly suitable to be used for cryptographic hash function. The chaotic nature of Logistic Map had been explored in this paper to produce a cryptographic hash function that is at least as good as SHA-1. The proposed new hash algorithm (CHA-1) produce hash value of 160-bit, accept any message length less than 2^{80} bits, and having security factor of 2^{80}.

Acknowledgments

The authors would like to express their thanks to Universiti Sains Malaysia (USM) for supporting this study.

References

[1] NIST, "Secure Hash Standard," *Federal Information Processing Standard, FIPS-180-1*, 1995.

[2] X. Wang, Y. L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1," *In Victor Shoup, editor, Advances in Cryptology—CRYPTO '05, Lecture Notes in Computer Science, Springer*, v. 3621, pp. 17–36, 2005.

[3] X. Wang, "Wikipedia, The Free Encyclopedia," *http://en.wikipedia.org/w/index.php?title=Xiaoyun_Wang &oldid=39379778*, 2006.

[4] M. S. Baptista, "Cryptography With Chaos," *Phys. Lett. A*, vol. 240, pp. 50–54, 1998.

[5] E. Alvarez, A. Fernandez, P. Garcia, J. Jimenez, and A. Marcano, "New Approach to Chaotic Encryption," *Phys. Lett. A*, pp. 373–375, 1999.

[6] R. Schmitz, J. Franklin, "Use of Chaotic Dynamical Systems in Cryptography," 338, pp. 429-441, 2001.

[7] W. K. Wong, and L. P. Lee, "A Modified Chaotic Cryptographic Method," *Comput Phys Commun 138: 234–6*, 2001.

[8] K. W. Wong, "A Fast Chaotic Cryptographic Scheme With Dynamic Look-Up Table," *Phys. Lett. A* 298, (2002) 238.

[9] Wikipedia contributors, "Logistic Map," *http://en.wikipedia.org/w/index.php?title=Logistic_map& oldid=41048057*, 2006.

[10] http://swox.com/gmp/,"GMP Arithmetic Without Limitations," release: 4.2.1, 2006.

Mahmoud Maqableh is a lecturer at the school of Managment Information System (MIS), University of Jordan. He received the B.Sc. degree in Computer Science from the Al Al-Bayt University, Jordan, in 2004. He obtained his M.Sc. degree in Computer Science from Universiti Sains Malaysia, Malaysia in 2006. His research interests are in the field of Cryptography, and E-Business Security.

Azman Samsudin is a lecturer at the School of Computer Sciences, Universiti Sains Malaysia. He received the B.Sc. degree in Computer Science from the University of Rochester, USA, in 1989. He obtained his M.Sc. and Ph.D. degrees in Computer Science from University of Denver, USA, in 1993 and 1998, respectively. His research interests are in the field of Cryptography, Interconnection Switching Networks, and Parallel Distributed Computing.

Mohammad A. Alia received his B.S. degree in Computer Sciences and M.S. degree in Information Technology from Al-Zaytoonah University in 2000 and 2003, respectively. During 2000-2004, he was an instructor of Computer Sciences and Information Technology at Al-Zaytoonah University-Jordan. Then, he worked as a lecturer at Al-Quds University in Saudi Arabia from 2004 - 2005. Currently he is a PhD student at the School of Computer Sciences, Universiti Sains Malaysia.