Chaos, Solitons and Fractals 87 (2016) 216–225

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

Nonlinear Science, and Nonequilibrium and Complex Phenomena

journal homepage: www.elsevier.com/locate/chaos

# A novel keyed parallel hashing scheme based on a new chaotic system

Meysam Asgari Chenaghlu*, Shahram Jamali, Narjes Nikzad Khasmakhi

*Department of Computer Engineering and Information Technology, Faculty of Technical Engineering, University of Mohaghegh Ardabili, Iran*

ARTICLE INFO

*Article history:*
Received 1 January 2016
Revised 27 March 2016
Accepted 9 April 2016
Available online 22 April 2016

*Keywords:*
Cryptography
Chaotic system
Keyed hash function
Meet-in-the-middle attack
Crypto-currency
Chaos coin

ABSTRACT

Hash functions play important role in the information security era. Although there are different methods to design these functions, in recent years chaos theory has emerged as a strong solution in this area. Chaotic hash functions use one-dimensional maps such as logistic and tent, or employ complex multi-dimensional maps which are typically insecure or slow and most of them has been successfully attacked. In this paper, we propose a new chaotic system and employ it to design a secure and fast hash function. The improved security factor has roots in the hyper sensitivity of the proposed chaotic map while properties like speed and security can be parameterized. On the other hand, the proposed hash function has a dynamic random array of functions and can be implemented by a parallel architecture. This data-level parallel architecture makes it fast to generate the hash value. Statistical simulations show success of the proposed hashing scheme. Cryptanalysis of proposed function, such as key sensitivity, meet-in-the-middle attack, collision, preimage resistance and high level attacks, proves security of the proposed function.



## 1. Introduction

A hash function is a one-way function that inputs an arbitrary length message denoted by $M$ and compresses it with nonlinear internal operations to $h(M)$ that can have fixed or variable length [1]. These functions can be categorized into two categories of cryptographic and non-cryptographic [2]. Non-cryptographic hash functions are used for storage indexing and such purposes that security countermeasures like preimage resistance does not apply to them. Having less collisions would be enough for a non-cryptographic hash function [3]. On the other hand, cryptographic hash functions must satisfy properties like collision and preimage resistance [4]. Collisions for a hash function like $h$, can be described as having two different messages of $M_1$ and $M_2$ that produce the same hash value, in other words, it can be said that $h(M_1) = h(M_2)$. Finding collisions for a hash function is done via searching through random inputs to reach two messages that produce the same output. Another scenario happens when an attacker like *Alice* with knowledge of a hashed output such as $H$ tries to find a message $M$ that produces $h(M) = H$. If such a message is found then it's said that $h$ is not first preimage resistant. In other words, for a pre-specified output, it is not computationally infeasible to find an input that produces the same output. Second preimage property is violated when *Alice* knows $h(M_1)$ and also $M_1$ and tries to find another message such as $M_2$ that holds condition of $h(M_1) = h(M_2)$ [5].

Hash function can be categorized as keyed or unkeyed based on its inputs. A keyed hash function denoted by $h_k$ is a hash function with the extra input of key. These functions index the hash value space by $k$. Thus, without knowledge of $k$, computing $h_k(M)$ or with knowledge of $M$ and $h_k(M)$ finding $k$ must be computationally infeasible. Other properties of unkeyed hash functions that has been described earlier apply to keyed hash functions too [2].

Due to these characteristics of hash functions, many cryptographic and non-cryptographic hash algorithms has been proposed. Most of these functions such as MD family of hash functions (*MD*4, *MD*5, *MD*6) [6], and *SHA* family (SHA1,SHA2) [7] use the Merkle–Damgård scheme and bitwise operations on input message to produce the final hash value. These functions hardly rely on nonlinear bitwise operations and implemented scheme. Wang and Yu showed that the Merkle–Damgård scheme is not secure [8] and also, other practical attacks on these functions such as [9] has been applied.

Sensitivity to input message and key as described earlier are the most important properties of a cryptographic function to posses and a hash function must have this functionality too. These properties are also described as confusion and diffusion by Shannon [10]. On the other hand, a chaotic map has these properties as sensitivity to initial values and design of hash functions with chaotic maps due to mentioned properties has been widely explored [11–25]. But unfortunately most of chaotic hash functions use Merkle–Damgård or a variant of that scheme which has been proven to be an insecure scheme [8]. Also, multi-dimension, slow or semi-random chaotic maps that has been used as core of these

* Corresponding author.
*E-mail address:* Asgari@student.uma.ac.ir (M. Asgari Chenaghlu).

http://dx.doi.org/10.1016/j.chaos.2016.04.007
0960-0779/

functions, reduces their performance metrics such as hashing speed and security. Successful attacks on some of these functions has been applied due to their weak cryptographic nature [26–29].

Hash function with described properties is variously used as a core element of cryptographic protocols, secure transactions and crypto-currencies. Bitcoin, as the most well known Crypto-currency uses Sha256 for HashCash proof of work function which provides security over transactions made between peers in Bitcoin network [30]. HashCash has been used to prevent spammers by making them spend certain amount of time to compute stamp of any message or E-Mail. If a message without stamp is received, then it is rejected and considered as spam [31]. However in crypto-currencies the ideology is different. A transaction is made from one peer in network to other one with approve of other peers. Approve of other peers is done by hashing input data and sending it to other peers in network. Hashing power measured by *GH/s* (Giga Hash per Second) is the metric used for measuring hashing speed of a cyrpto-currency network. A faster network provides faster transaction; in other words, with more peers contributing in network and with faster CPU's, transactions will be made much more faster [32]. But the bottleneck rises when Sha256 has no multi-threading ability and is not a keyed hash function. Thus many different implementations of this function are made to make it run over GPU with multi data and processing threads. In this case, threads does not mean that Sha256 has been broken into several functions, but instead, several instances (multi-processes) of this function run over different inputs of data (multi-data) to make it faster. Thus, design and implementation of a new hash function, which can be faster, multi-threaded, keyed and much secure with a different scheme than Merkle–Damgård while keeping all of mentioned properties parameterizable is highly desired in industry and the literature [33].

In this paper, first we propose a chaotic system that has more sensitivity to initial values and can be constructed using any one dimensional chaotic map. Simplicity and high chaotic behavior of our proposed system is shown in various simulations. Next, we use this chaotic system as our core element of hash function and propose our chaotic keyed hash function that has multi-threading ability and its properties like speed and security are parameterizable. We examine our algorithm against generic attacks and results show its robust cryptographic behavior.

Rest of this paper is organized as follow. Section 2 describes previous chaotic systems and chaotic hash functions. Our approach to chaotic system design is presented in Section 3. The new hashing scheme and proposed hash function are represented in Section 4 while Section 5 analyzes performance metrics such as statistical analysis, cryptanalysis attacks and speed tests. Finally, Section 6 concludes the whole paper and describes future works.

## 2. Background

In this section chaotic maps and relative hash functions are discussed. Section 2.1 covers related chaotic maps and systems while Section 2.2 describes relative works about chaotic hash functions.

### 2.1. Chaotic systems

Simplicity of implementation and complex chaotic behavior of one dimensional chaotic maps, made them a popular tool for cryptographic applications such as hash functions. Logistic map as one of these maps is defined by Eq. (1). Although this map has range of (0, 4] for parameter $r$, it shows chaotic behavior only in range of [3.57, 4] and even in this range, the map loses its chaotic behavior by some parameters [34]. Fig. 1 shows bifurcation diagram of this map.

$$X_{n+1} = L_r(X_n) = r.X_n.(1 - X_n) \quad (1)$$

Sine and tent maps are other examples of one dimensional chaotic maps that have similar behavior to logistic map; specially sine map has very similar bifurcation diagram with the logistic map and shares same problem in its chaotic behavior. To overcome this problems, Zhou et al. proposed a chaotic system with two one dimensional maps as seeds that will be denoted as *LS* map in rest of this paper for seeds of logistic and sine map. Their system shows beta distribution with $\alpha = \beta$ on output chaotic series [35]. Fig. 2 shows bifurcation diagram of *LS* map.

In cryptographic applications, having any sort of non-uniform distribution over a random number generator can be used by attackers to break the cipher or algorithm using differential attacks or its variants [36]. For beta distributions with $\alpha = \beta$, because of mostly distributed values over range of [0,0.1) and (0.9,1], it can be much easier to attack the proposed method. Fig. 2 shows distribution of the *LS* map over output range. As another shortcoming of the *LS* system, obtaining output from a sine map can be much slower in comparison to a logistic map. Thus, the *LS* system has seeds with a different computation complexity which makes it more vulnerable to side-channel timing attacks [37].

Another solution is presented by San-Um and Srichavengsup in [24]. Their approach employs a *Sinusoidal Nonlinearity* to create a chaotic system. Multiplication, addition and subtraction operations are usually supported by CPU in low level, however trigonometric operations like *sin* and *cos* are not supported by CPU in low level and higher level libraries are needed to be implemented and run over low level operations like multiplication and division [38]. Employing such operations in hash function design greatly reduces its hashing speed and imposes extra overhead. A typical multiply operation on a *Pentium IV* processor has 4 clock cycles latency while this value for *sin* operation is about 90–100 clock cycles [39].

A much robust solution to design chaotic systems out of one dimension chaotic maps is presented in [25]. This approach benefits from multi coupling of a chaotic map on different time samples that nearly ensures better chaotic behavior. But in the case of logistic map, parameter range expansion occurs for $r$ at [2, 4] which slightly improves logistic map's parameter range.

To fulfill properties like speed, simplicity of implementation, parameter range expansion, random behavior and security, we propose a new chaotic system which is described in next section and furthermore, a series of random-number tests are applied to ensure the security of proposed system for cryptographic applications.

### 2.2. Chaotic hash functions

Chaotic hash functions can be categorized in different ways according to their chaotic scheme or multi-threading ability. Chaotic maps as base element of chaotic hash functions are more describing for categorizing them. Thus, we adopted some categories from [34] and added other categories as follows:

- Simple map-based hash functions
- Complex map-based hash functions
- New chaotic system based hash functions
- Parallel and Complex structured chaotic hash functions
- Conventional hash functions with chaotic modifications

Most simple chaotic map-based hash functions, such as [14,15] use a simple algorithm with one dimensional chaotic maps in comparison with other categories. In [14] a hash function using tent map and another variable like $\beta$ is proposed. Output series of tent map with a Merkle–Damgård scheme is used to obtain the final hash value. Problems of the tent map such as non-chaotic input ranges are also carried to hash function design. In [15] tent map with a simple *XOR* scheme is proposed. Any hash function that produces output size less than 256 bits is not secure; this design