Commun Nonlinear Sci Numer Simulat 16 (2011) 2810–2821

Contents lists available at ScienceDirect

Commun Nonlinear Sci Numer Simulat

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

Parallel hash function construction based on coupled map lattices

Yong Wang [a,b,c,*], Kwok-Wo Wong [b], Di Xiao [c]

[a] Key Laboratory of Electronic Commerce and Logistics, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[b] Department of Electronic Engineering, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon Tong, Hong Kong
[c] College of Computer Science and Engineering, Chongqing University, Chongqing 400044, China

ARTICLE INFO

Article history:
Received 3 May 2010
Received in revised form 20 September 2010
Accepted 1 October 2010
Available online 16 October 2010

Keywords:
Hash function
Coupled map lattices
Chaos-based cryptography
Parallel

ABSTRACT

In this paper, the parallel structure of hash function is analyzed. Then, a parallel hash function based on coupled map lattices is proposed. The message is partitioned into blocks with fixed length. The message block is firstly processed by the hash round function. The final hash value is the mixed result of all the outputs of the hash round functions. The hash round functions are mainly implemented by the coupled map lattices and can work in a parallel mode, which guarantees good security and high efficiency. Theoretic analyses and numerical simulations both show that the proposed hash algorithm possesses good statistical properties, strong collision resistance and high efficiency. These properties make it a good candidate for hash on parallel computing platform.

1. Introduction

Hash function is one of the major tools in cryptography, which is usually used for data integrity in conjunction with digital signature schemes. Hash function takes a message as input and produces an output referred to as a hash value, or simply hash. More precisely, a hash function h maps bit strings of arbitrary finite length to strings of fixed length. Conventional hash functions, such as MD5 and SHA, involve logical operations or multi-round iterations of some available ciphers. Although each step is simple, the number of processing rounds could be huge even if the message is very short. Moreover, recent investigations on the collision frequencies reveal many undiscovered flaws in the well-known methods, such as MD5, SHA1, and RIPEMD [1–3]. As a result, the research on the design of secure and efficient keyed/unkeyed hash functions attracts more and more attentions.

As a ubiquitous phenomenon in nature, chaos is a kind of deterministic random-like process found in nonlinear dynamical systems. It is employed for data protection due to its attractive features such as sensitive to initial value, random-like and ergodic. Chaos-based hash function is attracting more and more interests of researchers. Based on Baptista's encryption method, Wong developed a scheme combining encryption and hashing [4]. Some hash algorithms based on simple chaotic maps, such as the piecewise linear chaotic map and tent map, are proposed [5–7], which have higher efficiency. Furthermore, the researches on the security of chaos-based hash function are studied. Deng, etc. finds a potential flaw in a chaos-based hash algorithm and presents some measures to improve its security [8,9]. In Ref. [10], a special hash function based on chaos is proposed, which has the abilities of modification detection and localization. In recent years, some hash functions based on spatiotemporal chaotic system are presented, which can prevent attackers from breaking the hash function by predicting the chaotic series effectively [11,12]. Compared with simple chaotic maps, spatiotemporal chaos possesses two additional merits

* Corresponding author at: Key Laboratory of Electronic Commerce and Logistics, Chongqing University of Posts and Telecommunications, Chongqing 400065, China. Fax: +86 23 62461172.
E-mail address: wangyong_cqupt@163.com (Y. Wang).

1007-5704/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cnsns.2010.10.001

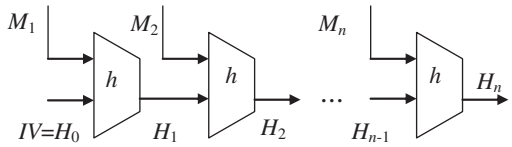

Fig. 1. General structure of iterated hash function.

for cryptographic purpose. Due to the finite computing precision, chaotic orbits will eventually become periodic. The period of spatiotemporal chaos is longer than that of simple chaotic maps [13]. In particular, the period of chaotic orbits generated by a system with a large number of chaotic coupled oscillators is too long to be reached in practical communications. Therefore, periodicity can be practically avoided in spatiotemporal chaotic systems [14]. Moreover, spatiotemporal chaotic system is a high dimensional chaotic system, which has a number of positive Lyapunov exponents that guarantee the complex dynamical behavior. It is more difficult or even impossible to predict the time series generated by spatiotemporal chaos.

Although the spatiotemporal chaotic system has some advantages in designing hash function, more computation are needed in producing a hash value. In general, the efficiency of the hash function based on spatiotemporal is lower than that of the one based on simple chaotic maps. On the other hand, personal computers equipped with double-core processor become more and more popular in recent years. In most of hash functions, an iterated structure shown in Fig. 1 is employed [7,15], which is a sequential structure. It is hard for the hash functions with this structure to make full use of the advantage of double-core processor computers. To the best of our knowledge, only a few parallel hash functions have been proposed [16,17]. With the aid of parallel computing, we can greatly improve the efficiency of the hash function based on spatiotemporal chaos. However, there are some loopholes in the parallel hash functions based chaotic maps. In Ref. [18], a forgery attack on the algorithms in [16,17] is proposed. Thus, further studies on parallel hash function are needed, which is very helpful to make this kind of hash functions apply in practical lives.

In this paper, the aim is to design a parallel hash function with high efficiency based on spatiotemporal chaos. The message is partitioned into 128-bit message blocks. One of the popular spatiotemporal chaotic model named coupled map lattices (CML) is used to generate hash round values. All the message blocks are processed in parallel. The hash value is obtained from the mixed results of the all hash round values. The rest of the paper is organized as follows: In Section 2, the preliminary knowledge of hash functions is introduced. In Section 3, the details of the proposed hash function are described. The security and performance of the proposed hash function are analyzed in Section 4, while the effect of finite computing precision is discussed in Section 5. Finally, conclusions are drawn in Section 6.

2. Preliminaries

Based on the difference of the relation between lattices, the coupled map lattices can be classified into globally coupled maps and nearest-neighbor coupled maps. In this paper, the nearest-neighbor coupled-map lattices (NCML) is employed as the model to generating hash values.

Definition 1. The NCML used in this paper is described as follows:

$$x_{n+1}(i) = (1-\varepsilon)f(x_n(i)) + \varepsilon f(x_n(i+1)), \tag{1}$$

where $n = 1,2,\ldots$ is the time index or state index; $i = 1,2,\ldots,N$ is the lattice index; f is a chaotic map and $\varepsilon \in (0,1)$ is a coupling constant. The periodic boundary conditions, $x_n(N+i) = x_n(i)$, is used in this system. Here the tent map is taken as the local chaotic map, given as:

$$x_{i+1} = \begin{cases} x_i/b, & x_i \leqslant b, \\ (1-x_i)/(1-b), & x_i > b, \end{cases} \tag{2}$$

where $b \in (0,1)$ is a constant.

Definition 2. A hash function $h(x)$ is a function possessing the following properties [15]:

(i) h maps an input x arbitrary finite length to an output $y = h$ of fixed length.
(ii) given h and x, it is easy to compute $y = h(x)$.
(iii) For essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find x such that $h(x) = y$ when given any y for which a corresponding input is not known.
(iv) It is computationally infeasible to find any two distinct inputs$x$ and x' which hash to the same output, i.e., such that $h(x) = h(x')$.