Optik 127 (2016) 4484–4489

Contents lists available at ScienceDirect

Optik

journal homepage: www.elsevier.de/ijleo

Collision analysis and improvement of a hash function based on chaotic tent map

Yantao Li*

College of Computer and Information Sciences, Southwest University, Chongqing 400715, China

ARTICLE INFO

Article history:
Received 26 December 2015
Accepted 25 January 2016

Keywords:
Collision analysis
Message extension
Aggregation operation
Hash function
Performance analysis

ABSTRACT

We analyze the computational collision problem on a hash algorithm based on chaotic tent map, and then present an improvement of the original algorithm in this paper. More specifically, we utilize message extension to enhance the correlation of plaintexts in the message and aggregation operation to improve the correlation of sequences of message blocks, which significantly increase the sensitivity between message and hash values, thereby greatly resisting the collision. Finally, we evaluate the performance of the improved algorithm by computer simulation, and the results show that it can resist the computational collision and can satisfy the requirements of a more secure hash algorithm.

1. Introduction

The network communication security issue is increasingly becoming more and more severe with the development of science and technology [1,2]. In the common context of network communication, the attacks of interception, interruption, modification, and fabrication are the main threats [3–6]. Message authentication is able to authenticate a message and to provide integrity and authenticity assurances on the message, which can be used for resisting the above attacks [7]. Hash function is any algorithm that maps data of arbitrary length to data with a fixed length, and it has been widely used in the area of modern information security, which is the key technology to implement integrity protection, message authentication, and digital signature. Therefore, the one-way hash function is becoming the research hotspot of information security area. In cryptography, some hash algorithms have been utilized as standards of hash functions [8,9]. Chaos has some inherent merits, such as one way, sensitivity, and ergodicity, which are suitable for designing irreversible, forgery resistance, and sensitive hash algorithm [10]. With the development of applications of chaos theory, some hash algorithms based on chaos are proposed [10–22]. Some of them have been proven insecure [3–5,23], and others need further investigation.

In [19], the authors propose a novel hash function based on chaotic tent map with changeable parameter. Based on the analysis, however, we find that the algorithm cannot resist the computational collision. Then, we provide an improvement associated to the original algorithm. More specifically, we utilize message extension to enhance the correlation of plaintexts in the message and aggregation operation to improve the correlation of sequences of message blocks, which significantly increase the sensitivity between message and hash values, thereby greatly resisting the collision. Finally, we simulate the improved algorithm and evaluate its performance, and the results show that the improved algorithm satisfies the requirements of a more secure hash algorithm.

The rest of this paper is organized as follows: Section 2 reviews the original algorithm, Section 3 provides the computational collision on the original algorithm in detail, and Section 4 presents an improvement by utilizing message extension and aggregation operation. In Section 5, we evaluate the performance of the improved algorithm by computer simulation and present conclusions in Section 6.

2. Review of the original algorithm

In this section, we first present the chaotic tent map used in the original algorithm, and then describe the original algorithm.

* Tel.: +8613637922551; fax: + 862368252352.
E-mail addresses: yantaoli@foxmail.com, liyantao@live.com, yantaoli@swu.edu.cn

http://dx.doi.org/10.1016/j.ijleo.2016.01.176
0030-4026/© 2016 Elsevier GmbH. All rights reserved.

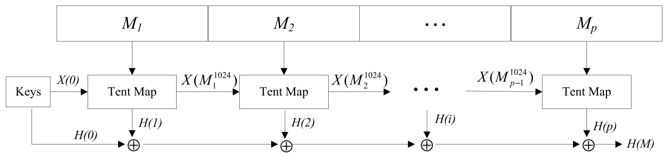

Fig. 1. The structure of the original algorithm.

2.1. Chaotic map

The chaotic map chosen in the original algorithm is the piecewise linear chaotic map and it is defined as:

$$X(k+1)=\begin{cases} X(k)/Q, & 0\le X(k)\le Q,\\ (X(k)-Q)/(0.5-Q), & Q\le X(k)<0.5,\\ (1-Q-X(k))/(0.5-Q), & 0.5\le X(k)<1-Q,\\ (1-X(k))/Q, & 1-Q\le X(k)\le 1,\end{cases}$$

where, $X(k)\in[0,\ 1]$ and Q is the control parameter satisfying $0<Q<0.5$. The secret key of the algorithm includes the initial state value $X(0)$.

2.2. Description of the original algorithm

The structure of the original algorithm can be illustrated in Fig. 1 and the length of the hash value is set to be $N=128$ bits. For the preprocessing step, the original message M' with arbitrary length is first padded with bits $\{1010...10\}$ and then leave 64 bits rightmost for indicating the length of the original message M', such that the length of the padded message is a multiple of 1024 bits. The final hash value $H(M)$ is generated by the following steps:

Step 1: The padded message M is first divided into p blocks $M=(M_1,M_2,\ldots,M_p)$, each block with a length of 1024 bits (N characters). Then each block M_i ($i=1,2,\ldots,p$) is further divided into N sub blocks, and each with a length of 8 bits (1 character) is denoted as $M_i^j(j=1,\ldots,N)$. Next, convert each sub block $M_i^j(j=1,\ldots,N)$ of the message block M_i ($i=1,2,\ldots,p$) into its corresponding ASCII code value.
Step 2: Choose $X(0)$ and $H(0)=\{0\}^N$ as the secret keys of the algorithm.
Step 3: For each sub block M_i^j of the message block M_i, we iterate the chaotic map M_i^j times with initial value of the last chaotic state $X(M_i^{j-1})$ or secret key $X(0)$ if and only if both $i=1$ and $j=1$, and the changeable parameter $Q=(\frac{i}{p}+\frac{j}{N})/2$, to generate the current state $X(M_i^j)$, and then round $X(M_i^j)$ to its nearest integer (0 or 1). When all sub blocks $M_i^j(j=1,\ldots,N)$ in the current M_i are processed, we obtain N values composed of 0 and 1. By cascading the N values, we obtain the intermediate hash value $H(i)$ with N bits.
Step 4: When all the blocks M_i ($i=1,2,\ldots,p$) in the padded message M are processed, we obtain the N-bit final hash value by $H(M)=H(p)\oplus H(p-1)\oplus\cdots\oplus H(1)\oplus H(0)$.

3. Collision analysis of the original algorithm

In this section, we will discuss the collision issue of the original algorithm, and show that the final hash value is not sensitive to the original message. More specifically, the original algorithm cannot resist collisions to some extent. The collision resistance of a hash function is defined as: it should be hard to find two distinct messages M and M' such that $H(M)=H(M')$.

From Section 2.2, we know that the binary values (0 or 1) in the intermediate hash value are obtained by rounding the

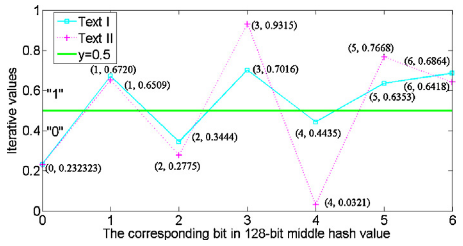

Fig. 2. Collision example derived from the chaotic map of the original algorithm.

state values $X(M_i^j)(i=1,2,\ldots,p;j=1,2,\ldots,N)$ to their nearest integers, and the final hash value is simply generated by $H(M)=H(p)\oplus H(p-1)\oplus\cdots\oplus H(1)\oplus H(0)$. As a result, we guess that the sensitivity of the original algorithm is very low and it may cause lots of collisions.

We conduct a collision test, which consists of computing the difference between the hash values obtained from the original message and from a modified version of this message. For the test, we first choose $X(0)=0.232323$ and $Q=(\frac{i}{p}+\frac{j}{N})/2$ as the initial value and changeable parameter of the chaotic map, respectively, which are the same settings as [19]. The reason is that we can increase the comparability of the collision test. Then, we select two similar plaintexts of messages with the same length, and the same content except for the first 16 bits (the first 2 characters). For example, Text I is $M'=adM_1^3\ldots M_i^j$, and Text II is $M''=bcM_1^3\ldots M_i^j$, where $i=p$ and $j=N$. As we can see, there are only 16 bits (two characters) are different between Texts I and II, and the remainder are the same.

For Text I: $M'=adM_1^3\ldots M_i^j$, the first two iterative values of the chaotic map are:

$$X(1)=T^{\mathrm{ASCII}(a)}(X(0),Q(0))=T^{97}(0.232323,0.0044)=0.6720$$

$$X(2)=T^{\mathrm{ASCII}(d)}(X(1),Q(1))=T^{100}(0.6720,0.0049)=0.3444$$

where, ASCII(*) represents the ASCII code value of character "*", and $T^{\mathrm{ASCII}(*)}(X(i),Q(i))$ indicates the iterative value by iterating the chaotic map ASCII(*) times with initial value $X(i)$ and control parameter $Q(i)$. Therefore, we can deduce that the first two bits of hash value $H(1)$ in Text I are "1" and "0", respectively.

On the other hand, for Text II: $M''=bcM_1^3\ldots M_i^j$, the first two iterative values of the chaotic map are:

$$X(1)=T^{\mathrm{ASCII}(b)}(X(0),Q(0))=T^{98}(0.232323,0.0044)=0.6509$$

$$X(2)=T^{\mathrm{ASCII}(c)}(X(1),Q(1))=T^{99}(0.6509,0.0049)=0.2775$$

The first two bits of the final hash value $H(1)$ in Text II are "1" and "0", respectively, which are equivalent to the results of Text I.

Next, for a specific collision test, we randomly choose two messages for Text I and Text II as: Text I: "adhongqing University" and Text II: "bchongqing University". The corresponding first 6 bits in the intermediate hash value are illustrated in Fig. 2. From Fig. 2, we can see that the first 6 bits of the intermediate hash value of Text I are "101011", while those of Text II are "101011" as well. Given that the first 2 bits of the two text messages are different, the corresponding hash values have at least 6 equivalent bits at the same locations, therefore, we conclude that the original algorithm has severe collisions and it is not sensitive and secure.

4. Improvement of the original algorithm

In order to improve the capability of collision resistance, sensitivity between input message and output hash value, and security of the original algorithm, we present an improvement method on the algorithm. The improvements of the original algorithm are performed in the following steps: (1) message extension in Step