

基于混沌神经网络和 C-MD 结构的带密钥哈希函数

陈立全^{1,2}, 朱宇航¹, 王宇¹, 秦中元¹, 马旻¹

(1. 东南大学网络空间安全学院, 江苏 南京 210096; 2. 网络通信与安全紫金山实验室, 江苏 南京 211111)

摘要: 近年来, 被广泛使用的 MD5、SHA-1 等哈希算法存在不同程度的安全隐患, 现在通用的 SHA-2 算法迭代结构与 SHA-1 算法相同, 使得其存在被攻破的可能性。而 SHA-3 由于其内部结构复杂, 实现复杂度较高。设计并实现了基于混沌神经网络和 C-MD (chaotic neural network-Merkle-Damgard) 结构的带密钥哈希函数, 为提高安全性改进了 Merkle-Damgard 结构, 并提出 C-MD 结构, 将该结构应用于哈希函数设计可以抵抗中间相遇攻击、多碰撞攻击以及针对长信息的第二原像攻击; 使用混沌神经网络作为压缩函数, 以提高哈希函数复杂度, 增强函数的抗碰撞性, 支持函数输出多种长度; 设计一个明文预处理器, 使用耦合映像格子产生与明文长度相关的混沌序列对明文进行填充, 增强哈希函数抵抗长度扩展攻击的能力。仿真实验结果表明, 提出的哈希函数效率优于 SHA-2、SHA-3 等的同类型混沌哈希函数, 能够抵御第二原像攻击、多碰撞攻击和差分攻击等多种攻击方式, 同时具有更好的抗碰撞性和映射均匀性。此外, 提出的哈希函数可以输出不同长度的散列值, 能够较好地应用在数字签名、密钥生成、基于哈希的消息认证码、确定性随机比特发生器等领域。

关键词: 哈希函数; 混沌神经网络; C-MD 结构; 耦合映像格子

中图分类号: TP311

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2023033

New hash function based on C-MD structure and chaotic neural network

CHEN Liquan^{1,2}, ZHU Yuhang¹, WANG Yu¹, QIN Zhongyuan¹, MA Yang¹

1. School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China

2. Purple Mountain Laboratories for Network and Communication Security, Nanjing 211111, China

Abstract: In recent years, widely used hash algorithms such as MD5 and SHA-1 have been found to have varying degrees of security risks. The iterative structure of the SHA-2 algorithm is similar to that of SHA-1, making it vulnerable to attacks as well. Meanwhile, SHA-3 has a complex internal structure and low implementation efficiency. To address these issues, a keyed hash function was designed and implemented based on chaotic neural network and

收稿日期: 2022-05-12; 修回日期: 2022-11-11

通信作者: 陈立全, lqchen@seu.edu.cn

基金项目: 国家重点研发计划 (2020YFE0200600)

Foundation Item: The National Key R&D Program of China (2020YFE0200600)

引用格式: 陈立全, 朱宇航, 王宇, 等. 基于混沌神经网络和 C-MD 结构的带密钥哈希函数[J]. 网络与信息安全学报, 2023, 9(3): 1-15.

Citation Format: CHEN L Q, ZHU Y H, WANG Y, et al. New hash function based on C-MD structure and chaotic neural network[J]. Chinese Journal of Network and Information Security, 2023, 9(3): 1-15.

C-MD structure. The approach involved improving the Merkle-Damgard structure by proposing the chaotic neural network Merkle-Damgard (C-MD) structure. This structure can be used to design a hash function that can withstand attacks such as the middle attack, multiple collision attack, and second pre-image attack for long information. Besides, the chaotic neural network was used as the compression function to increase the complexity of the hash function and improve its collision resistance, while also enabling it to output multiple lengths. Moreover, a plaintext preprocessor was designed, which used the coupled image lattice to generate chaos sequence related to the length of the plaintext to fill the plaintext, thus enhancing the ability of the hash function to resist length expansion attacks. Simulation results demonstrate that the proposed hash function performs faster than SHA-2, SHA-3 and the same type of chaotic hash function proposed by Teh et al. It can resist second pre-image attack, multi-collision attack and differential attack, while also exhibiting better collision resistance and mapping uniformity. In addition, the proposed Hash function can output Hash values of different lengths, making it suitable for use in digital signature, key generation, Hash-based message authentication code, deterministic random bit generator, and other application fields.

Keywords: hash function, chaotic neural network, C-MD structure, coupled mapping lattice

0 引言

随着计算机的普及、信息化时代的到来,人们的生活与互联网紧密地结合在了一起。与此同时,信息的泄露、篡改、伪造等各类信息安全问题逐渐引起社会的广泛关注^[1]。目前被广泛采用的密码技术有对称加密算法、非对称加密算法、哈希函数和消息认证码等^[2-3]。哈希函数又称散列算法,是一种将任意长度数据压缩为指定长度的算法。算法所产生的哈希值被称为消息摘要或散列值。哈希函数可以对任意一段明文产生指定摘要,一旦数据发生微小的改变,散列值也会产生巨大变化。因此,哈希函数常被用于数据完整性验证、电子签名、鉴别协议、零知识证明等^[4-5]。

哈希函数需要具有原像稳固性、第二原像稳固性、碰撞稳固性等性质。但传统的基于 Merkle-Damgard(MD)结构的哈希算法,如 MD5、SHA-0、SHA-1 等未完全满足上述性质,被证明存在安全性风险,如不能抵抗长度扩展攻击、定点碰撞攻击和多碰撞攻击^[6-7]。2005 年,王小云等采用差分和 XOR 相结合的分析方法对 MD5 等哈希函数进行碰撞攻击^[7];同年王小云团队宣布对 SHA-1 的 80 轮碰撞攻击仅需复杂度为 2^{69} 次操作^[8]。Gilbert^[9]和 Hawkes^[10]等公布了 SHA2 系列算法存在局部碰撞的安全风险。2009 年, NIST 宣布 SHA-2 将取代 MD5、SHA-0 和 SHA-1 等哈希算法,并开始寻求新的哈希标准。2015 年,Keccak

提出的算法成为 SHA-3 的新标准。但 Keccak 设计的哈希函数涉及大量的内部位操作,导致计算开销大、硬件实现复杂^[11]。虽然它的设计是轻量级的,但在计算开销和时间效率上仍不如 SHA-1 和 MD5。2017 年 2 月 23 日, Google 宣布与 Centrum Wiskunde&Informatica (CWI) Amsterdam 合作创建了两个具有相同 SHA-1 值和不同内容的 PDF 文件,表明 SHA-1 算法已被证明不安全。由于 MD5 和 SHA-1 已被证明不安全, SHA-3 存在计算复杂度较高的问题,设计高效而安全的哈希函数是密码学的一个重要研究方向,具有现实意义。

近年来,基于混沌神经网络的哈希函数^[12-13]所具有的混沌和神经网络的非线性安全性引起了相关研究人员的广泛关注^[14]。混沌是由非线性动力系统产生的一种确定性类随机过程^[15,16]。混沌行为是不稳定的,其依赖于初始条件和控制参数进行不规则运动,这种不规则的运动是不确定的、不可重复的、不可预测的^[17]。由于上述性质,混沌系统常被用于密码算法和哈希函数的设计。神经网络是一种模仿生物神经网络的结构,是一种功能强大的计算模型,其具有良好的灵活性、非线性、单向性和数据压缩性。因此,有学者尝试将神经网络和混沌理论相结合构造哈希函数^[18]。Xiao 等^[19]提出了一种基于混沌神经网络的哈希和加密结合的方案,但其密钥非随机数,方案存在安全性风险。Lian 等^[20-21]利用神经网络的扩散

特性和混沌的混淆特性构造了一个哈希函数,但该函数没有考虑多碰撞攻击。Yang 等^[22]设计了一种基于 16 个帐篷映射组成网络结构的混沌散列算法。此算法散列过程中,每个帐篷映射的初始值和控制参数由上一个块的输出决定。Huang 等^[23]提出一种并行键控哈希函数,改进了 Xiao 方案中密钥非随机的问题。He 等^[13]设计了一种基于混沌神经网络构造单向哈希函数的算法,神经网络由两个混沌映射初始化,明文分块后依次输入神经网络,散列值从神经网络输出层结果中提取。Abdoun 等^[24]将混沌神经网络与 Merkle-Damgard 结构结合构造了一种新型哈希算法。上述方案中,将混沌系统和神经网络应用在哈希函数的设计中提升了系统的随机性和安全性。如何运用混沌系统的不稳定性、敏感性以及神经网络的单向性、压缩性构造出满足原像稳固、第二原像稳固以及碰撞稳固的哈希函数仍是一项充满挑战性的工作。同时由于 MD5 和 SHA-1 已被证明不安全,SHA-3 存在计算缺陷,本文研究设计了一种新的混沌哈希算法,该算法使用耦合映射格子和卷积神经网络,并考虑了对长度扩展、Joux 多重碰撞攻击和其他攻击方法的抵抗能力。

本文将混沌系统和神经网络相结合构造压缩函数,改进了 MD 迭代结构,提出了 chaotic neural network-Merkle-Damgar 结构(下文简称 C-MD 结构),并通过上述方案设计了一款哈希函数。新哈希函数在保证速度的前提下既可以抵抗第二原像攻击、多碰撞攻击、差分攻击以及长度扩展攻击等传统攻击,又可以抵抗量子攻击。本文的主要贡献如下。

1) 从安全性角度出发改进了 MD 结构,提出了 C-MD 结构。通过增加前后级连接的步函数,迭代次数受密钥与明文影响等方式,该迭代结构具有更高的安全性,可以抵御多碰撞攻击。

2) 增加了预处理阶段。在正式运行哈希函数前,使用耦合映像格子(CML, coupled map lattice)生成混沌序列进行明文填充,提高了函数的抗长度扩展攻击能力,针对原始明文进行混淆和置乱,以提高函数抵御第二原像攻击的能力。

3) 构造了一个前后级联的混沌神经网络作为 C-MD 迭代结构的压缩函数,提升了系统的复

杂度,增强了函数的扩展性。另外,所提方案通过调节神经网络的节点可以轻易改变哈希函数的输出长度,增强了哈希函数的实用性。

1 相关工作

1.1 耦合映像格子

耦合映像格子是一个将连续状态变量离散化的非线性动力系统^[25]。该系统在空间域中划分出大量格子,每个格子都是简单的混沌系统。格子之间相互耦合,相互影响,极大提高了系统的复杂度。耦合映像格子模型数学定义如式(1)、式(2)所示。

$$g_n = \frac{\varepsilon}{2} (f(x_n(i+1)) + f(x_n(i-1))) \quad (1)$$

$$x_{n+1}(i) = (1 - \varepsilon)f(x_n(i)) + g_n \quad (2)$$

其中, i 为离散空间坐标,取值为 $i=1,2,\dots,\text{Len}$, Len 为格子大小, n 为离散时间坐标, $\varepsilon \in (0,1)$ 为耦合系数,边缘条件满足 $x_n(L) = x_n(0)$ 。当 f 为混沌映射时,系统的动力学特征也表现为混沌的。

本文采用的混沌映射为 Logistic 映射,其算法复杂度低且被广泛应用,动力学方程式如式(3)所示。

$$f(x) = \mu \cdot x(1 - x) \quad (3)$$

其中,混沌系数 $\mu \in (3.569\,945\,6, 4]$, $x \in (0,1)$ 。本文选取 $\mu=3.999\,997$,以保证耦合映像格子中的映射始终处于混沌状态,从而使耦合映像格子保持混沌状态。图 1 是 $\mu=3.999\,997$ 且 Len 为 32 时, CML 系统的时空混沌图。

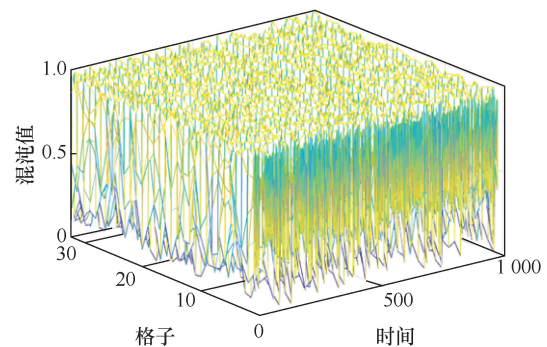


图1 CML 系统的时空混沌图
Figure 1 Space-time chaos diagram of CML system

1.2 C-MD 迭代结构

哈希函数需要满足碰撞稳固、原像稳固和第

二原像稳固等安全特性。面对长度未知的数据,使用者难以有效构造满足上述安全性要求的单一函数。因此,主流哈希函数通常使用迭代结构来满足上述要求。压缩函数每次处理固定长度的数据,通过多次迭代压缩函数,完成对全部数据的处理。压缩函数是一种映射,定义见式(4)。

$$f: \{0,1\}^t \rightarrow \{0,1\}^u \quad (4)$$

式中, f 为压缩函数的映射, u 为压缩函数输出长度 ($u > 0$), t 为数据分组的长度 ($t > 0$)。

MD 结构是一种经典的哈希函数迭代结构^[26,27],如图 2 所示。Damgard 证明了 MD 结构是碰撞稳固性继承的,即如果压缩函数是碰撞稳固的,则迭代该压缩函数所构造的哈希函数也具有碰撞稳固的性质。MD 结构在许多经典的哈希算法中有应用,如 SHA-1 算法、MD5 算法等。图 2 中 IV 为初始向量, f 为压缩函数, M_i 为分组明文, Hash 为 MD 结构迭代后生成的散列值。

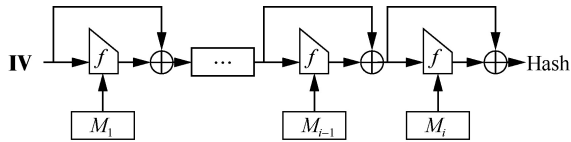


图 2 MD 结构
Figure 2 MD structure

经典的 MD 迭代结构具有碰撞稳固性结构,近年来对散列函数碰撞问题的研究取得了突破性的进展,发现了 MD5、SHA-1 等在抗碰撞性方面的一些缺陷。为了抵抗针对长信息的第二原像攻击、中间相遇攻击、多碰撞攻击等,本文改进了 MD 结构,并提出 C-MD 结构。该结构使用了混沌神经网络 (Chaotic neural network),增强了结构的复杂度,可以有效抵御上述攻击。C-MD 结

构如图 3 所示。其中, IV 表示初始向量, Cq_i 表示第 $i-1$ ($i = 1, 2, \dots, n$) 块密文 C_i 为初始格子产生的混沌序列, f_i 为压缩函数,每个压缩函数的参数都不相同,其参数由 CML 所生成的混沌序列决定。 S_1 、 S_2 为压缩函数的步函数,是一种可逆函数。 n 代表数据块的总数量。图中 $\oplus$ 符号代表异或操作。C-MD 结构的输入为长度是 t bit 的分组数据,输出为长度是 u bit 的中间链接变量。该结构使用新型明文填充方式,除了填充原始消息的长度之外,其余的占位符都是由耦合格子映射产生的。这种填充方式可以保持压缩函数的抗碰撞性。如图 3 所示, C-MD 结构在 MD 迭代结构的基础上额外增加了两个步函数,使得迭代结构前后级间耦合度更高。

1.3 混沌神经网络

混沌神经网络采用耦合的混沌振荡子作为单个神经元。相比传统的混沌系统,混沌神经网络拥有更加优秀的混淆和扩展特性,因此其适用于各种基于混沌的密码设计。多通道输入单通道输出的神经网络具有良好的压缩性质,对输入向量、神经元权重和控制参数等具有敏感性。混沌神经网络同时具有单向性,即很容易通过输入向量计算输出向量,且很难通过输出向量逆向计算输入值。此外,混沌神经网络具有混沌和扩散的特点,可以在迭代过程中混淆和扩散数据。

混沌神经网络具有上述优秀的密码学性质,近年来其被广泛用于各类新型密码系统设计。本文构造了一种混沌神经网络作为哈希函数的压缩函数,该混沌神经网络拥有 K 层卷积层,输入层具有 8 个神经元,中间层及输出层均为 4 个神经元。混沌神经网络压缩函数结构如图 4 所示。图中输入数据经过输入层处理后传送至卷积层,各

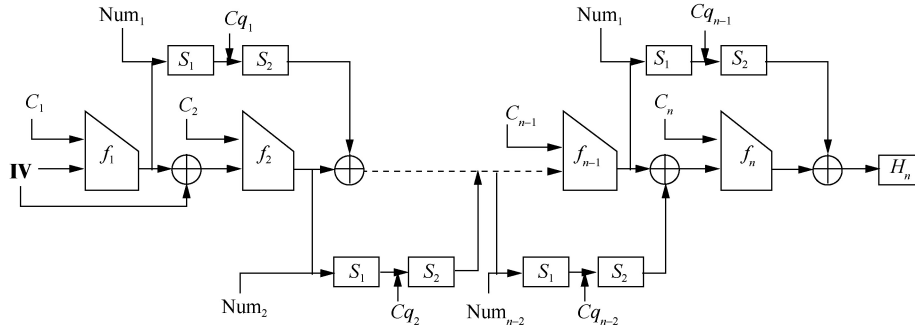


图 3 C-MD 结构
Figure 3 C-MD structure

卷积层将运算结果输出到数据拼接器中。数据拼接器将得到的浮点数转化为十六进制数，并拼接为十六进制字符串。最后将各卷积层的十六进制字符串相异或得到输出向量 h 。

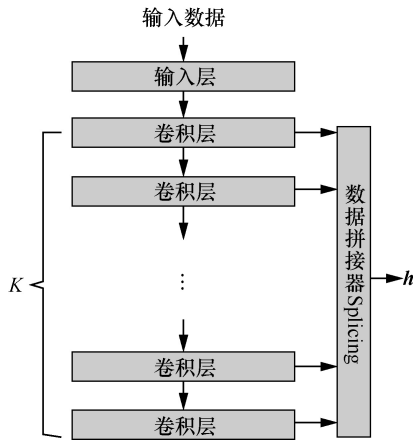


图4 混沌神经网络压缩函数结构

Figure 4 Chaotic neural network compression function structure

2 本文算法描述

本节具体描述基于混沌神经网络和 C-MD 结构的带密钥哈希函数的计算流程。设计的哈希函数共有 6 个处理步骤，其中步骤 1 和步骤 2 为预处理阶段，进行明文填充和明文加密。步骤 3~步骤 6 为哈希函数散列值生成阶段，该阶段主要对预处理阶段输出的数据进行分组、重加密、压缩与迭代，算法流程如图 5 所示。

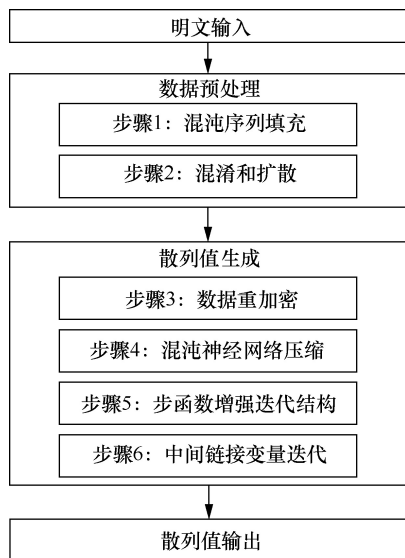


图5 基于 C-MD 的哈希函数算法流程

Figure 5 Flow of Hash function algorithm based on C-MD

函数的输入：不同长度的明文数据。

函数的输出： u bit 散列值。

预处理阶段：步骤 1 与步骤 2 均属于预处理阶段，此阶段的目的是将明文填充并加密，以供后续步骤进行处理。本文提出了一种新型数据填充方式，所有的填充数据为耦合映射格子所产生的混沌序列。这种填充方式增加了明文的安全性并且保证了压缩函数的抗碰撞性不变。数据预处理阶段流程如图 6 所示，在此阶段首先使用混沌序列对明文数据进行填充，然后对明文数据进行混淆和扩散。下面具体描述预处理阶段的两个步骤。

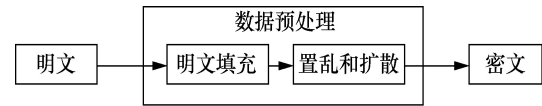


图6 数据预处理阶段流程

Figure 6 Flow of data preprocessing stage

步骤 1 设置分组长度为 t ，首先判断输入的明文数据长度加上标识符长度是否为分组长度 t 的倍数，若是，则跳至步骤 2；若否，则对明文进行填充。填充规则如图 7 所示。

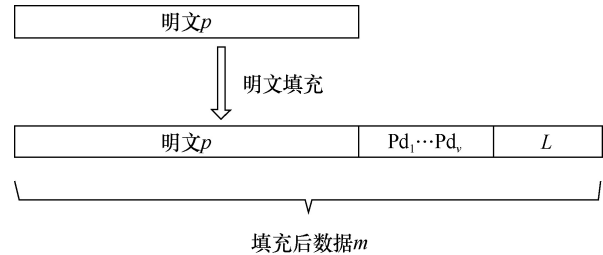


图7 填充规则

Figure 7 Padding rule

图 7 中 p 为初始明文， $Pd_j (1 \leq j \leq v)$ 表示占位符， L 代表明文长度，用 64 bit 二进制数据表示，用于记录原始明文长度， m 为填充后数据。

明文填充长度为 v bit，由式(5)计算得到。

$$v = (q - 1) \cdot t - \text{mod}[(L + 64), t] \quad (5)$$

其中， t 为分组长度， L 为明文长度， q 为与用户密钥相关的随机数，计算过程见式(6)，其决定了随机额外填充的分组块数。然后设置耦合映射格子的初始格子值 $\mathbf{LA} = [1, 3, 7, 9, 13, 27, 31, 37, 39, 49, 51, 57, 63, 67, 73, 79, 81, 91, 93, 97, 99, 11, 23, 27, 29, 33, 39, 41, 51, 11, 23, 5]$ 。系统通过 Logistic 映射迭代生成伪

随机序列, 将该伪随机序列作为系统默认密钥 **Key**。系统默认密钥为后续计算所使用的密钥。耦合映射格子的耦合系数 ε_0 由式(6)计算得出。

$$\begin{cases} \text{Key} = \text{Logistic}(\text{mod}(L \cdot \text{Mat} \cdot \text{userKey}, 1), \mu) \\ \varepsilon_0 = \text{mod}(\text{key_cof}^T \cdot \text{Key}, 1) \\ q = \text{floor}(\text{mod}(L \cdot \text{Mat} \cdot \text{userKey}, 255)) \end{cases} \quad (6)$$

其中, L 为明文长度, **Mat** 为密钥变换矩阵, μ 为 Logistic 映射分型参数, 其值为 3.999 997。key_cof 为密钥系数向量, 耦合映射格子以 **LA** 为初始格子值, 以 ε_0 为耦合系数, 生成混沌序列 **Pd**, 然后将 **Pd** 转化为二进制序列得到明文的填充序列。

将原始明文 p 、混沌序列 **Pd** 以及明文长度 L 拼接得到填充后的明文序列 m , $m=(p||\text{Pd}||L)$ 。

步骤 2 将步骤 1 中填充完成的数进行混淆和扩散并将数据进行分组, 具体方式如下。

① 利用耦合映射格子产生混沌序列 **cq**; 将序列 **cq** 与明文 m 按位异或, 完成对明文 m 的扩散操作, 生成序列 c_1 。

② 使用约瑟夫混淆算法对序列 c_1 进行混淆操作得到密文序列 c , 其中, 约瑟夫混淆算法的起始位置 **start**、步长 **step** 两个参数也由耦合映射格子产生的混沌序列提供。

③ 将密文序列 c 以 t 比特数据为一组, 分为 n 组数据, 分组数据用 M_i 表示, i 为分组序号。

至此, 预处理阶段结束, 下面进入哈希函数散列值生成阶段。首先, 将预处理步骤中输出的分组数据 M_i 进行重加密, 得到加密的分组数据 C_i , 其中 i 为数据分组序号。随后, 将加密后的分组数据依次输入增强迭代结构中进行数据压缩与迭代, 生成中间链接变量 H_i 。处理完所有分组数据后得到最终的中间链接变量 H_n , 该值即哈希函数生成的散列值。重加密过程与增强迭代结构中所有参数均由耦合映射格子所产生的混沌序列提供。此处的耦合映射格子受初始格子值 **LA**、用户密钥所生成的系统密钥 **Key**、分组数据 M_i 、分组密文 C_i 以及中间链接变量 H_i 控制生成混沌序列。哈希函数散列值生成阶段如图 8 所示。

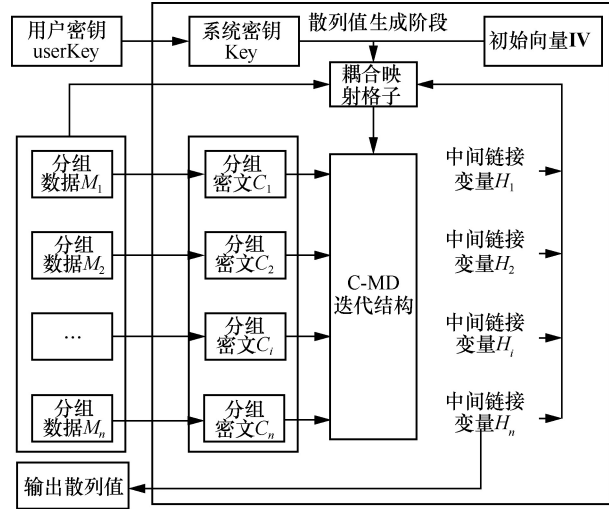


图 8 哈希函数散列值生成阶段
Figure 8 Hash value generation stage

步骤 3 为了保证加密强度, 需要对每一个数据分组进行重加密。每个分组使用不同密钥进行加密以提高安全性。加密方式如下。

① 设置初始向量 $\mathbf{IV} = [101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 1, 3, 5]$, 此向量是哈希函数中间链接变量的初始值, 也是处理第 1 个分组数据的耦合映射格子的初始格子值。

② 按照式(7)计算各个加密模块的 CML 的耦合参数 ε 。

$$\varepsilon_i = \begin{cases} \text{mod}(\text{key_cof}^T \cdot \text{Key}, 1), & i = 1 \\ \text{mod}(C_{i-1}^T \cdot \text{Key}, 1), & 1 < i \leq n \end{cases} \quad (7)$$

其中, **Key** 是系统默认密钥, **key_cof** 为密钥系数向量, 由系统设置, C_i 是第 i 个加密后分组数据。

③ 按照式(8)对 32 个格子进行初始化设置。

$$\begin{cases} \mathbf{LA}_i = \text{mod}(\text{sqrt}(M_i \oplus H_{i-1}), 1), \\ 1 < i \leq n \end{cases} \quad (8)$$

其中, M_i 为预处理步骤输出的第 i 个分组数据。将初始格子值 **LA** 和耦合参数 ε_i 作为 CML 的参数, 生成混沌序列 Q , 其将被用于后续的重加密之中。

④ 按照式(9)对每一个分组的明文进行加密:

$$\begin{cases} C_i(1) = M_i(1) \oplus Q(1) \oplus M_i(\text{end}) \\ C_i(j+1) = C_i(j) \oplus Q(j+1) \oplus M_i(j+1) \\ C_i = \text{Joseph}(C_i, \text{start}, \text{step}) \end{cases} \quad (9)$$

其中, C_i 为加密后的密文分组序列, j 为分组数据中元素的序号 ($0 < j < t+1$), t 为分组数据长度, i 为当前的明文块的次序。 Q 是由耦合格子映射生成的混沌序列。 start 是约瑟夫置乱的开始位置, step 是置乱的间隔距离, 它们都是由耦合格子映射生成的。

重加密后的分组密文 C_i 依次被输入 C-MD 迭代结构中进行压缩与迭代。 C-MD 迭代结构使用混沌神经网络作为压缩函数, 增加了 Step₁、Step₂ 两个可逆的步函数, 增加了结构的复杂度, 处理第一个分组数据与其他分组数据方式不同。 上述这些改进可以使该结构有效抵御第二原像攻击、多碰撞攻击、差分攻击以及长度扩展攻击等, 提升哈希函数的安全性。 下面详细介绍数据压缩及迭代过程。

步骤 4 将步骤 3 所输出的密文分组数据按照 64 bit 为一组划分为 8 组, 输入混沌神经网络中。 将此 64 bit 数据按照 double 型数据读入, 再通过每个节点的权重转换为浮点数。 单层混沌神经网络结构如图 9 所示。

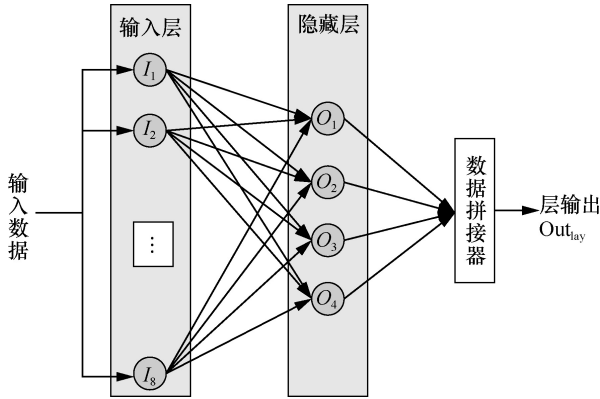


图 9 单层混沌神经网络结构

Figure 9 Structure diagram of single-layer chaotic neural network

混沌神经网络按照式(10)计算输入层输出向量。

$$\begin{cases} W_1 = [1/2^8, 1/2^{12}, 1/2^{16}, \dots, 1/2^{32}, 1/2^8]^T \\ \text{Input} = [\text{input}_{k0}, \text{input}_{k1}, \dots, \text{input}_{k7}]^T \\ I_l = f(W_1^T \cdot \text{Input}, Q_l) \end{cases} \quad (10)$$

其中, W_1 是输入层节点的权值。 Input 是输入向量, 其中 $k \in [0, 63]$, $l \in [0, 7]$ 。 I_l 是输入层的输出

结果。 Q_l 为式(8)所产生的混沌序列提供的控制参数。 f 为激活函数, 该激活函数为分段线性混沌映射 (PLCM, piecewise linear chaotic map), 定义式如下。

$$f(x, q) = \begin{cases} \frac{x}{q}, & 0 \leq x < q \\ \frac{x-q}{0.5-q}, & q \leq x < 0.5 \\ \frac{1-q-x}{0.5-q}, & 0.5 \leq x < 1-q \\ \frac{1-x}{q}, & 1-q \leq x \leq 1 \end{cases} \quad (11)$$

其中, q 是分段线性映射的控制参数, 其值由耦合映射格子产生的混沌序列提供, x 是分段映射的初始参数。 激活函数的输出即该输入层神经元的输出 I_l 。

设输入层的神经元为 l , 第一个隐藏层的神经元为 j , 则输入神经元 l 与隐藏层神经元 j 之间的权重为 $W_2(l, j) \in (0, 1)$, 第一个隐藏层的每个神经元 j 的阈值为 $\delta_j \in [0, 1]$, 都由 CML 所生成的混沌序列决定。

设权重矩阵 W_2 的第 j 列为 W_2^j 。 Q_j 为式(8)中混沌序列提供的控制参数。 输出如式(12)所示。

$$\begin{cases} y = I \cdot W_2^j + \delta_j \\ O_j = f(\text{mod}(y, 1), Q_j), j \in \{0, 1, 2, 3\} \end{cases} \quad (12)$$

O_j 是第一个隐藏层第 j 个节点的输出结果。 将 O_j 转换为十六进制数, 并与层内其他节点输出结果串接, 得到层输出 Out_{lay} , lay 为神经网络的层序号。

经过 K 层神经网络运算后得到每层的输出 (本文中 K 取 3), 然后按照式(13)计算第 i 个密文分组的压缩数据 h_i 。

$$h_i = \text{Out}_1 \oplus \text{Out}_2 \oplus \text{Out}_3 \quad (13)$$

步骤 5 为了保证 C-MD 的安全性, 需将步函数设置为可逆函数。 本文中步函数的输出 S_1 和 S_2 由式(14)计算得到。

$$\begin{cases} S_1 = \text{Num}_{i-1} \oplus h_{i-1}, & i = 2, 3, \dots, n \\ S_2 = S_1 \oplus Cq_{i-1}, & i = 2, 3, \dots, n \end{cases} \quad (14)$$

其中, Num_i 为 CML 生成的与 h_i 等长的混沌序列,

该 CML 耦合系数由 L 生成, 初始格子值为 h_{i-1} 。将 Num_{i-1} 与 h_{i-1} 异或求得 S_1 。 Cq_{i-1} 为 CML 产生的混沌序列, 将 Cq_{i-1} 与 S_1 异或得到 S_2 。

步骤 6 中间链接变量迭代, 上一个数据块的压缩数据经过步函数处理后得到输出向量 S_2 ; S_2 与本数据块的压缩数据相异或得到中间链接变量 H_i , 如式(15)所示。

$$\begin{cases} H_i = S_2 \oplus h_i, & i = 2, 3, \dots, n \\ H_1 = \mathbf{IV} \oplus h_1, & i = 1 \end{cases} \quad (15)$$

迭代步骤 4~步骤 6, 依次处理所有密文分组, 最终生成的中间链接变量 H_n 即明文数据的散列值。

3 哈希函数的性能表现和安全性分析

本节对所设计的 C-MDhash 进行安全性分析和性能测试。首先分析哈希函数的性能表现, 其中包含对散列值分布的统计、散列值 H 对于明文和密钥的敏感性的测试, 以及散列值的抗碰撞性的仿真。随后分析本文所提出的迭代结构对于暴力攻击和密码学分析的抵抗能力。最后分析所设计哈希函数的实现复杂度并与其他哈希函数进行对比。

3.1 敏感性测试

密码学哈希函数需要对明文和密钥高度敏感, 理想情况下, 即使 1 bit 明文的变化, 也要有散列值中 50% 的位翻转。本节通过对输入的明文和密钥做出微小的修改, 构造 6 个测试样例, 对所设计的哈希函数进行测试。设置用户密钥为 $\text{userKey} = \text{"THIS_IS_USERS_K3Y_123456"}$ 。设置 CML 的参数 $\mu = 3.999\ 997$ 。条件 1 为原始测试文本, 首先固定密钥 userKey , 通过对条件 1 明文文本做出不易察觉的微小改变。如字母级别修改、单词级别修改、标点符号修改、插入空格等生成条件 2~条件 5 测试样例。其中, 字母级别修改是在明文中随机选择一个字母, 将其修改为 ASCII 码相邻的字母, 即只修改明文 1 bit 信息。通过固定明文, 对用户密钥做出微小修改, 生成条件 6 测试样例。

条件 1 原始明文消息是 Southeast University is a university with a long history and profound deposits. The university was founded in 1902 as

SanJiang Normal School, and later experienced the important development periods of LiangJiang Normal School, Nanjing Higher Normal School, National Southeast University, and National Central University. 1952 The National College of Technology was founded on the site of the National Central University, with the former School of Engineering of National Central University as the main body, and the related departments of Fudan University, Jiao Tong University, Zhejiang University, JinLing University and so on. In May 1988, the school was renamed Southeast University. In April 2000, the former Southeast University, Nanjing Railway Medical College and Nanjing Communications College were merged, and Nanjing Geological School was merged to form the new Southeast University. In almost 120 years of history, always with the world of southeast university, the heart is the motherland, for the advancement of science, national rejuvenation and unyielding, the pursuit of excellence, gradually formed a "rigorous, pragmatic, unity, forge ahead" good school spirit and "the scientific name, country," with the talent of the school-running idea, made the school motto of "goodness" spirit.

条件 2 将最后一句话的最后一个单词的第一个字母 s 改为 t。

条件 3 将最后一句话中 goodness 更改为 good。

条件 4 将第一个句号改为逗号。

条件 5 在名称 SanJiang 中间添加一个空格。

条件 6 使明文与条件 1 相同, 将用户密钥修改为 $\text{userKey} = \text{"THIS_IS_ANOTHER_USERS_K3Y"}$ 。

条件 2~条件 5 测试样例分别从比特级别、单词级别以及标点符号、空格等特殊字符的改变来反映哈希函数对明文变化的敏感性, 条件 6 测试样例通过密钥变化来反映哈希函数对密钥的敏感性。上述样例较好地涵盖了常见测试场景, 较好地反映 C-MD 哈希对明文变动和密钥变动的敏感性。不同条件下的十六进制散列测试结果如表 1 所示。

表 1 6 种条件下的十六进制散列值
Table 1 Hexadecimal Hash values under six conditions

条件	散列值
1	6BC35778C2CB3219D4DDE819655B04A521AA1A FE792638D0CB4E23240017DE0F
2	8148C288C3C6CF5B0A9E7FF0579F6A276237F313 C3393B210706B6641279B983
3	CFE8EB351410CA6C8CBA42037B950340BD9117D 4EAD2751613A2D1DC628CE85F
4	A0A84576903A0FB0B7945C7CFED47A6FF25A427 DD12CC343B72D5D2CCE78A238
5	7406B5B68EDDA9D39467E939E7B430DB4D497C9 1F1E0B03F509509DE5507AE96
6	F3D04C5A3C931EBF0AA8C83E9957425459BD7DF FF7AA4982425B8371A5653128

由表 1 可知, 明文和密钥微小的改变也会导致所生成的散列值产生较大的变化。下面进行数理上的统计分析, 进一步量化本文所设计哈希函数的性能。

3.2 统计分析

散列值的分布是否均匀是衡量哈希函数性能的一项重要指标。散列值分布均匀时信息熵最大, 这意味着攻击者无法从统计分析中获取额外的信息。理想的哈希函数应遵守严格雪崩准则, 即翻转明文(二进制格式)的任意一位, 散列值 H 的每一个输出位变化概率为 50%^[28,29]。

在实验测试中, 本文通过对任意一条明文 M , 每次随机改变明文中 1 bit 信息, 重新计算其散列值。将此过程分别重复 N 次 ($N=256, 512, 1\ 024, 2\ 048, 10\ 000$), 计算以下 6 个统计指标, 以衡量哈希函数的映射均匀性。

(1) 平均变化比特数

$$\bar{B} = \frac{1}{N} \sum_{i=1}^N B_i$$

(2) 平均变化率

$$P = \frac{\bar{B}}{u} \cdot 100\%$$

(3) 比特变换比特数标准差

$$\Delta B = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (B_i - \bar{B})^2}$$

(4) 变化概率标准差

$$\Delta P = \sqrt{\frac{1}{N-1} \sum_{i=1}^N \left(\frac{B_i}{u} - P \right)^2} \cdot 100\%$$

(5) 最小变化比特数

$$B_{\min} = \min(B_i)$$

(6) 最大变化比特数

$$B_{\max} = \max(B_i)$$

本文实验中输出的散列值长度 $u=256$ bit, 表 2 分别给出了 $N=256, 512, 1\ 024, 2\ 048$ 和 $10\ 000$ 时算法的 6 项性能指标。

表 2 性能统计
Table 2 Statistical performance

次数 N	$\bar{B}$	P	ΔB	ΔP	$B_{\min}$	$B_{\max}$
256	127.50	49.81	7.9048	3.09	108	149
512	127.83	49.94	7.9852	3.12	106	150
1 024	128.02	50.01	8.2398	3.22	106	153
2 048	128.02	50.01	7.9457	3.10	104	153
10 000	127.99	50.00	8.0752	3.15	99	157

对于类似的算法而言, 平均变化位数 $\bar{B}$ 的理想值为 $u/2$, 本文实验中理想值为 128, 同时比特平均变化率 P 理想值为 50%。从表中可以看出 $\bar{B}$ 和 P 都接近理想值。因此, 本文哈希函数具有较好的映射均匀性。

将本文所提出的 C-MDhash 与同类型哈希函数进行性能比较, 对比实验见表 3。

从表 3 的对比分析可以看出, 散列值长度 $u=128$ bit 时, C-MDhash 各个指标与理论值相接近。散列值长度 $u=256$ bit 时, C-MDhash 的平均变化比特数 $\bar{B}$ 为 127.99, 比特平均变化概率 P 为 50.00%, 接近理论值 128 和 50%。 ΔB 与 ΔP 的值分别为 8.0752 和 3.15, 都非常小, 表明该函数的混乱和扩散性很稳定, 能够抵抗统计攻击和差分攻击。C-MDhash 在上述 6 个指标中的表现优于 MD5、SHA-2、SHA-3 等算法, 说明本文哈希函数具有较好的抗统计分析性能。

3.3 加密模块安全性分析

C-MDhash 中使用明文加密与分组数据重加密来保障数据安全。这两个加密模块的安全性由耦合映射格子产生的混沌序列的随机性保证。

表 3 对比实验 ($N=10\ 000$)
Table 3 Comparative experimental ($N=10\ 000$)

算法	$\bar{B}$	P	ΔB	ΔP	$B_{\min}$	$B_{\max}$
MD5	64.03	50.02	5.660 0	4.42	40	82
文献[23]	63.88	49.91	5.750 0	4.50	—	—
文献[30]	63.99	49.99	5.660 0	4.40	40	85
文献[31]	63.92	43.94	7.430 5	5.80	—	—
文献[32]	64.02	50.01	5.640 0	4.40	—	—
文献[33]	64.00	50.00	5.440 0	4.25	—	—
文献[34]	63.91	49.94	5.617 2	4.39	—	—
文献[35]	63.15	50.11	5.770 0	4.51	—	—
文献[36]	63.88	49.91	5.660 0	4.41	42	84
C-MDhash (128 bit)	64.04	50.03	5.570 1	4.35	43	85
文献[37]	127.98	50.04	8.106 0	3.12	99	156
文献[38]	127.70	49.88	8.220 0	3.21	99	156
SHA-2 (256 bit)	128.81	50.03	8.109 3	3.56	100	155
SHA-3 (256 bit)	128.43	50.10	8.109 3	3.76	100	155
C-MDhash (256 bit)	127.99	50.00	8.075 2	3.15	99	157

3.3.1 CML 系统混沌行为分析

本文中 CML 使用 Logistic 映射作为混沌映射，其分岔如图 10 所示。由图 10 可以看出，当 $\mu \in (3.569\ 945\ 6, 4]$ ，映射处于混沌区间。

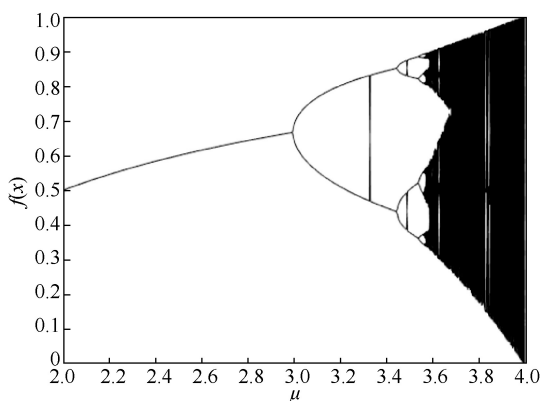


图 10 Logistic 映射分岔
Figure 10 Logistic mapping bifurcation diagram

对于 CML 系统，在 $\mu=3.999\ 997$ ， $\varepsilon=0.25$ 时的空间振幅情况如图 11 所示。可以看出，序列基本填满 $(0,1]$ 空间，具有混沌的特性。

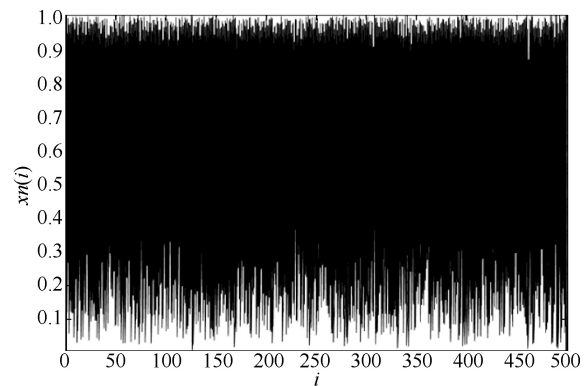


图 11 CML 系统空间振幅
Figure 11 Space amplification of CML system

3.3.2 NIST 随机性测试

NIST 测试包内包含 16 种测试方法，主要来判定要测试的二进制序列是否存在非随机的可能性。对于任意长度的二进制序列，若 NIST 测试的各项检测结果 P-value 大于标准值 0.01，说明通过该项测试，若能通过 NIST 的 16 项测试，则证明该序列具有良好的随机性。

下面通过设置 CML 的参数 $\mu=3.999\ 997$ ，每次随机更改初始格子值 $\mathbf{IV}$ 以及 CML 系统的耦合系数，生成长度为 2.56×10^5 的二进制序列，将此过程重复 1 000 次，对所生成的序列进行 NIST 随机性测试。各项测试结果如表 4 所示。

由实验结果知，1 000 个实验样本各项测试的 P-value 值都大于 0.01，通过率为 100%，说明混沌序列通过国际标准 NIST 随机性测试，该序列集合具有良好的随机性，因此 CML 在 $\mu=3.999\ 997$ ， $\varepsilon \in (0,1)$ 的情况下可以生成随机性较好的混沌序列。本文所设计数据预处理中的加密模块与分组数据重加密模块使用 CML 系统产生的混沌序进行数据扩散，然后以混沌序列为参数进行数据置乱。所以，加密模块的安全性依赖于 CML 系统所产生序列的随机性，因此加密模块的安全性也得以保障。

3.4 哈希函数密码分析

3.4.1 抗原像和第二原像攻击

原像攻击指的是，对于任意给定的哈希码 h ，攻击者试图找到一个消息 M 使得 $H(M)=h$ 。第二原像攻击指，对于任意给定的明文 M ，其哈希码为 h ，攻击者试图找到一个新的明文 $M'(M' \neq M)$ 使得 $H(M')=h$ 。本文所提哈希函数的 C-MD 结

构增加了迭代过程中预处理的消息块和随机块的数量。在不知道结构中已经处理的消息块的数量和随机添加的伪随机消息块的数量情况下，攻击的计算复杂度至少为 2^u 。本文中散列值长度 $u=256$ bit，因此原像攻击与第二原像攻击计算上不可行，本文所设计的哈希函数可以抵抗原像攻击与第二原像攻击。

表4 混沌序列 NIST 测试结果
Table 4 Chaotic sequence NIST test results

指标	P-value 通过率	结果
Frequency	100%	Random
Frequency within a Block	100%	Random
Run	100%	Random
Longest Run of Ones in a Block	100%	Random
Binary Matrix Rank	100%	Random
Discrete Fourier Transform	100%	Random
Non-Overlapping Template Matching	100%	Random
Overlapping Template Matching	100%	Random
Maurer's Universal Statistical	100%	Random
Linear Complexity	100%	Random
Serial	100%	Random
Approximate Entropy	100%	Random
Cummulative Sums (Forward)	100%	Random
Cummulative Sums (Reverse)	100%	Random
Random Excursions	100%	Random
Random Excursions Variant	100%	Random

3.4.2 抗长度扩展攻击

针对 MD 迭代结构，在不知道明文 M_1 的情况下，若已知哈希值 $H(M_1)=h$ ，则 $H(M_1||M_2)=f(h, M_2)$ ，其中 f 为压缩函数， H 为哈希函数， $||$ 为拼接符号。因此，对任意未知的 M ，若知道其哈希值 $H(M)$ 和其长度 l ，则可以计算 $H(M||<l>||y)$ ， y 为任意后缀。

下面分析本文所提出的 C-MDhash 在抵抗长度扩展攻击时的安全等级。C-MDhash 在预处理阶段使用与明文和用户密钥相关的伪随机序列对明文进行填充，随机填充 n 块伪随机数据块，且在末尾填充明文长度。故对于任意明文 M ，其哈希值为 $H(M||Pd||l)$ ， Pd 为伪随机序列， l 为明文 M 长度。由于填充了伪随机序列和明文长度，增加了数据随机性，在密钥不被泄露的情况下，攻

击者无法伪造出相同的混沌序列 Pd ，从而无法利用哈希值 $H(M||Pd||l)$ 和 l 进行长度扩展攻击，因此 C-MDhash 可以抵抗长度扩展攻击。

3.4.3 抗中间相遇攻击

中间相遇攻击是生日攻击的变体，在迭代过程的中间寻找链接变量来攻击哈希函数。本文在填充阶段对明文进行加密，对于伪造的消息，即使有 1 bit 的变化，也会导致中间连接变量和最终变量之间存在差异。此外，由于混沌系统对初值的敏感性，即使晶格初值的微小变化也会导致压缩函数的不同输出。因此，本文提出的哈希函数可以有效防止中间相遇攻击。

3.4.4 抗多碰撞攻击

若能找到多个不同的消息 $M_1, M_2, M_3, \dots, M_r$ 使得 $H(M_1)=H(M_2)=H(M_3)=\dots=H(M_r)$ ，则称之为多碰撞攻击。一个随机的 Hash 函数，找到一个 r 重的多碰撞需要哈希 $2^{n(r-1)/r}$ 个消息。当 r 很大时，这个值接近于 2^n ，即找到一个多碰撞所消耗的代价比普通的生日攻击所要求的 $2^{n/2}$ 复杂。2004 年，Joux^[39]证明对于输出长度是 n 的 Hash 函数，可以经过 $(lbr)2^{n/2}$ 次压缩函数的询问获得 r 重多碰撞，其攻击方法如下。

① 设 Hash 函数 H 的初始值 IV 为 h_0 ，压缩函数为 f 。

② 对 $i=1, \dots, r$ ，调用碰撞发现算法 C 找到 $B_i' \neq B_i$ ，使得 $f(h_{i-1}, B_i') = f(h_{i-1}, B_i) = h_i$ 。

③ 填充并输出 2^r 个形式为 $(b_1||\dots||b_r||padding)$ 的消息，其中 b_i 是 B_i' 、 B_i 中的一个。

这样，要找到 2^r 个碰撞需要 $r2^{n/2}$ 次操作。

本文所提出的 C-MD 迭代结构在 MD 迭代结构上增加了步函数，来抵抗多碰撞攻击，下面论证该结构的安全性。

假设现在通过一种碰撞发现算法 C 找到 $B_i' \neq B_i$ ，使得 $f(h_{i-1}, B_i') = f(h_{i-1}, B_i) = h_i$ ($i=1, \dots, r$)。此时其欲构建多碰撞攻击，则还需要破解以下关系。

① $\text{encryption}(B_i') = C_i', \text{encryption}(B_i) = C_i$ 。

② $S_2(Cq_{i-1}', S_1(\text{Num}_{i-1}, h_{i-1}')) \oplus h_i' = S_2(Cq_{i-1}, S_1(\text{Num}_{i-1}, h_{i-1})) \oplus h_i$

③ $S_2(Cq_i', S_1(\text{Num}_i, h_i')) = S_2(Cq_i, S_1(\text{Num}_i, h_i))$ 。

其中， encryption 为 C-MD 的重加密过程， B_i

为明文, C_i 为密文, S_1 、 S_2 是 C-MD 结构中的两个步函数, Num_{i-1} 由明文长度 L 和中间链接变量 h_{i-2} 决定。在密钥不被泄露的前提下, 上述 3 个条件均难以满足。若攻击者可以找到压缩函数 f 的碰撞, 无法同时满足条件 2 和条件 3, 则也无法实现多碰撞攻击, 因此本文所设计的 C-MDhash 在不泄露密钥的前提下抗多碰撞攻击。

3.4.5 抗差分攻击

差分攻击是专注于消息输入差异对输出结果影响的密码分析。一般采用像素数变化率 (NPCR, number of pixels change rate) 和统一平均变化强度 (UACI, unified average changing intensity) 来评估算法对差异攻击的抵抗能力。NPCR 与 UACI 的定义见式(18)、式(19)。

$$\text{NPCR} = \frac{\sum_{i,j} D(i,j)}{u} \cdot 100\% \quad (16)$$

$$\text{UACI} = \frac{1}{u} \left[\sum_{i,j} \frac{|h_1(i,j) - h_2(i,j)|}{255} \right] \cdot 100\% \quad (17)$$

其中, h_1 和 h_2 表示两个不同的散列值, u 表示散列值的长度。如果 $h_1(i,j) \neq h_2(i,j)$, 那么 $D(i,j)=1$, 否则 $D(i,j)=0$ 。表 5 和表 6 给出了 C-MDhash 与 SHA-2、SHA-3 以及其他同类型的哈希函数的 NPCR 和 UACI 值。

理论上, NPCR 的最佳值为 99.6093%, UACI 的最佳值为 33.4635%。从表 4 和表 5 可以看出, C-MDhash 的结果接近于最佳值。因此, 本文所提出的算法可以有效抵抗差分攻击。

表 5 哈希函数 NPCR
Table 5 The Hash function NPCR

哈希函数	散列值长度		
	$u=128$	$u=256$	$u=512$
SHA-2	—	99.609 5%	99.609 4%
SHA-3	—	99.609 4%	99.609 4%
文献[30]	89.093 2%	89.093 2%	—
文献[31]	97.832 9%	—	—
文献[32]	96.012 2%	—	—
C-MDhash	99.612 6%	99.600 3%	99.600 6%

表 6 哈希函数 UACI
Table 6 The Hash function UACI

哈希函数	散列值长度		
	$u=128$	$u=256$	$u=512$
SHA-2	—	28.305 7%	30.019 9%
SHA-3	—	30.085 7%	28.514 1%
文献[30]	31.542 3%	30.194 2%	—
文献[31]	32.802 0%	—	—
文献[32]	31.982 2%	—	—
C-MDhash	31.879 4%	34.218 2%	33.129 8%

3.4.6 哈希函数碰撞测试

实验中使用两个哈希函数的绝对差值和散列值中相同位置具有相同 ASCII 码数的分布来测试本文所提算法的抗碰撞性。碰撞分析的测试步骤如下。

① 任选一段明文计算其散列值, 以 ASCII 码格式保存。

② 随机改变明文的 1 bit 数据生成新的明文, 计算新明文的散列值, 以 ASCII 格式保存。

③ 使用式(18)、式(19)计算两散列值中同一位置具有相同 ASCII 码的个数。

$$\omega = \sum_{j=1}^N g(t(e_j), t(e'_j)) \quad (18)$$

$$g(x, y) = \begin{cases} 0, & x = y \\ 1, & x \neq y \end{cases} \quad (19)$$

④ 使用式(20)计算两个散列值之间的绝对差值。

$$d = \sum_{j=1}^N |t(e_j) - t(e'_j)| \quad (20)$$

其中, e_j 和 e'_j 分别是两个散列值的第 j 个 ASCII 字符, $t(e_j)$ 表示 ASCII 码 e_j 相应的十进制数值。 ω 的值越小或者 d 的值越大, 哈希函数的抗碰撞性越强。理论上, 若假设散列值随机均匀分布, 则 N 次独立实验中 ω 的分布情况可由式(21)计算得到。

$$W_N(\omega) = N \cdot P(\omega) = N \frac{s!}{\omega!(s-\omega)!} \left(\frac{1}{2^8}\right)^\omega \left(1 - \frac{1}{2^8}\right)^{s-\omega} \quad (21)$$

其中, $\omega \in [1, s]$, $s=u/8$ 。表 7 给出了在相同条件下进行 10 000 次实验, 散列值长度 $u=128$ bit、256 bit

两种情况下, d 的最大值、最小值以及平均值, 并且给出了其他同类型哈希函数的结果。从实验结果可以看出, 当散列长度 $u=256$ bit 时, d 的平均值为 2 840.07, 最大值为 4 125, 都大于 SHA-2 和 SHA-3 的值。当 $u=128$ bit 时, d 的平均值为 1380, 最大值为 2 168, 优于 MD5 的值, 且与其他同类型哈希函数的值相似, 说明 C-MDhash 具有良好的抗碰撞性。

表 8 给出了理论上 ω 的分布情况以及实验测得的 ω 的分布情况。从表 8 中可以看出实验结果与理论分布情况非常吻合, 这同样说明 C-MDhash 具有良好的抗碰撞性。

表 7 散列值间绝对差值 ($N=10\ 000$)
Table 7 The absolute difference between Hash values ($N=10\ 000$)

算法	Max	Min	Mean
MD5 (128 bit)	2 074	590	1 304
SHA-2 (256 bit)	3 819	1 789	2 707.10
SHA-3 (256 bit)	3 895	1 686	2 776.16
文献[13]	2 312	683	1 504
文献[30]	2 397	403	1 364
文献[31]	2 221	696	1 506
文献[32]	2 321	584	1 366
文献[33]	2 379	630	1 360.9
文献[34]	2 156	658	1 431.3
文献[36]	2 213	730	1 426.23
文献[37]	—	—	2 715
文献[38]	3 831	1 695	2 715.39
文献[40]	2 418	796	1 598.6
C-MDhash (128 bit)	2 168	517	1 380
C-MDhash (256 bit)	4 125	1 553	2 840.07

表 8 散列值在相同位置具有相同 ASCII 码数的分布
Table 8 Distribution of the number of ASCII characters with the same value at the same location in the hash value

相同 ASCII 码数	碰撞次数理论值	碰撞次数实验值
0	8 822.2	8 851
1	1 107.8	1 027
2	67.3	114
3	2.6	8
4	0.1	0

3.4.7 哈希函数复杂度分析

C-MDHash 在明文进行预处理阶段和分组重加密阶段, 对于长度为 L 的明文, 采用按位异或和约瑟夫置乱操作, 这两个操作的时间复杂度均为 $O(L)$ 。预处理和分组重加密阶段 CML 迭代生成长度为 L 的混沌序列, 在格子大小固定为 32 的情况下, CML 生成和明文等长的混沌序列, 时间复杂度为 $O(L)$ 。故预处理和分组重加密阶段算法时间复杂度为 $O(L)$ 。散列值生成阶段, 每个分组的大小为 512 bit, 压缩函数为混沌神经网络, 分组数据输入 3 层混沌神经网络, 输入层执行 8 次乘法运算, 隐藏层执行 8×4 次乘法运算, 输出层执行 4×4 次乘法运算, 故对于 n 个分组的数据其复杂度为 $56n$, 即 $O(n)$ 。CML 对 n 个分组分别生成长度为 48 的混沌序列作为神经网络权重, 算法复杂度也为 $O(n)$ 。该算法时间复杂度为线性的, 理论上达到了较好的复杂度, 下面通过实验测试 C-MDHash 的效率, 并与其他哈希函数进行对比。

本文实验使用 Matlab 2020b 在 Intel Core i7-10700@2.90 GHz 处理器, 8 GB RAM, Windows10 操作系统上实现了所提出的哈希函数。表 9 给出了 C-MDhash (256 bit) 以及其他同类型哈希函数速率比较。

表 9 哈希函数速率比较
Table 9 Hashing speed comparison

算法	速率/(Gbit·s ⁻¹)	平台
MD5	4.758	Intel Core i7, 32 GB RAM
SHA-2 (256 bit)	2.120	Intel Core i7, 32 GB RAM
SHA-3(256 bit)	2.622	Intel Core i7, 32 GB RAM
文献[18]	0.666	Intel Core i7, 16 GB RAM
文献[32]	2.550	Intel Core i7, 16 GB RAM
文献[37]	1.234	Intel Core i5, 8 GB RAM
文献[41]	0.697	Intel Core i7, 16 GB RAM
文献[42]	0.933	Intel Core i5, 16 GB RAM
C-MDhash (256 bit)	3.060	Intel Core i7, 8 GB RAM

从表 9 中可以看出, 本文提出的 C-MDhash 对比文献[32]等同类型的混沌哈希函数和 SHA-2、SHA-3 具有更快的处理速率, 但其速率低于 MD5、SHA-1 等哈希函数。原因是 C-MDhash 为了提高安全性, 增加了预处理和分组重加密, 并且使用了混沌神经网络, 数据压缩过程中运算

对象为浮点数, 运算速率低于传统哈希函数的整数运算, 因此降低了哈希函数速率。这一速率差距可以通过调整神经网络结构或修改神经元处理的数据类型来改善。

4 结束语

本文设计并实现了一种基于混沌神经网络和C-MD结构的带密钥哈希函数。该哈希函数基于C-MD结构, 使用混沌神经网络作为压缩函数, 并利用耦合映射格子产生混沌序列提供加密、压缩等过程的参数。理论分析和实验测试结果表明, 该哈希函数对于明文和密钥具有较高的敏感性, 同时可以抵抗针对长信息的第二原像攻击、中间相遇攻击、差分攻击以及多碰撞攻击。在性能分析中, 实验结果表明提出的哈希函数在牺牲一些速率的前提下比MD5、SHA-1函数拥有更好的抗碰撞性和安全性, 并且在实现速率方面优于SHA-2、SHA-3。此外, 本文提出的哈希函数算法可以输出不同长度的散列值, 更适用于数字签名、密钥生成、基于哈希的消息认证码、确定性随机比特发生器等多个领域。

参考文献:

- [1] LIN Z, GUYEUX C, YU S, et al. On the use of chaotic iterations to design keyed hash function[J]. Cluster Computing, 2019, 22(1).
- [2] ISLAM S H. Provably secure dynamic identity-based three-factor password authentication scheme using extended chaotic maps[J]. Nonlinear Dynamics, 2014, 78(3).
- [3] CHAIN K, KUO W. A new digital signature scheme based on chaotic maps[J]. Nonlinear dynamics, 2013, 74(4): 1003-1012.
- [4] DATCU O, MACOVEI C, HOBINCUI R. Chaos based cryptographic pseudo-random number generator template with dynamic state change[J]. Applied Sciences, 2020, 10(2).
- [5] WANG M, WANG X, ZHAO T, et al. Spatiotemporal chaos in improved cross coupled map lattice and its application in a bit-level image encryption scheme[J]. Information Sciences, 2021, 544: 1-24.
- [6] WANG Y, LIAO X, XIAO D, et al. One-way hash function construction based on 2D coupled map lattices[J]. Information Sciences, 2008, 178(5): 1391-1406.
- [7] WANG X, YU H. How to break MD5 and other hash functions[C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. 2005. 19-35.
- [8] WANG X, YIN Y L, YU H. Finding collisions in the full SHA-1[C]//Annual International Cryptology Conference. 2005. 17-36.
- [9] GILBERT H, HANDSCHUH H. Security analysis of SHA-256 and sisters[C]//International Workshop on Selected Areas in Cryptography. 2003. 175-193.
- [10] HAWKES P, PADDON M, ROSE G G. On corrective patterns for the SHA-2 family[J]. IACR Cryptol. ePrint Arch. 2004, 2004: 207.
- [11] TUTUEVA A V, KARIMOV A I, MOYSIS L, et al. Construction of one-way hash functions with increased key space using adaptive chaotic maps[J]. Chaos, Solitons & Fractals, 2020, 141: 110344.
- [12] LI Y, XIAO D, DENG S, et al. Parallel Hash function construction based on chaotic maps with changeable parameters[J]. Neural Computing and Applications, 2011, 20(8): 1305-1312.
- [13] HE B, LEI P, PU Q, et al. A method for designing hash function based on chaotic neural network[C]//International Workshop on Cloud Computing and Information Security (CCIS). 2013.
- [14] ROSENBLATT F. The perceptron: a probabilistic model for information storage and organization in the brain[J]. Psychological Review, 1958, 65(6): 386.
- [15] LORENZ E N. Deterministic nonperiodic flow[J]. Journal of Atmospheric Sciences, 1963, 20(2): 130-141.
- [16] HILBORN R C. Chaos and nonlinear dynamics: an introduction for scientists and engineers[M]. Oxford University Press on Demand, 2000.
- [17] DACHSELT F, SCHWARZ W. Chaos and cryptography[J]. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 2001, 48(12): 1498-1509.
- [18] TEH J S, SAMSUDIN A, AKHAVAN A. Parallel chaotic hash function based on the shuffle-exchange network[J]. Nonlinear Dynamics, 2015, 81(3): 1067-1079.
- [19] XIAO D, LIAO X. A combined hash and encryption scheme by chaotic neural network[C]//International Symposium on Neural Networks. Springer, 2004. 633-638.
- [20] LIAN S, SUN J, WANG Z. Secure hash function based on neural network[J]. Neurocomputing, 2006, 69(16-18): 2346-2350.
- [21] LIAN S, LIU Z, REN Z, et al. Hash function based on chaotic neural networks[C]//2006 IEEE International Symposium on Circuits and Systems. 2006. 4.
- [22] YANG H, WONG K, LIAO X, et al. One-way hash function construction based on chaotic map network[J]. Chaos, Solitons & Fractals, 2009, 41(5): 2566-2574.
- [23] HUANG Z. A more secure parallel keyed hash function based on chaotic neural network[J]. Communications in Nonlinear Science and Numerical Simulation, 2011, 16(8): 3245-3256.
- [24] ABDOUN N. Design, implementation and analysis of keyed hash functions based on chaotic maps and neural networks[R]. 2019.
- [25] KANEKO K. Pattern dynamics in spatiotemporal chaos: Pattern selection, diffusion of defect and pattern competition intermittency[J]. Physica D: Nonlinear Phenomena, 1989, 34(1-2): 1-41.
- [26] CORON J, DODIS Y, MALINAUD C, et al. Merkle-damgård revisited: how to construct a hash function[C]//Annual International Cryptology Conference. 2005. 430-448.
- [27] MERKLE R C. Secrecy, authentication, and public key systems[R]. Stanford University, 1979.
- [28] SHANNON C E. Communication theory of secrecy systems[J]. The Bell System Technical Journal, 1949, 28(4): 656-715.
- [29] FEISTEL H. Cryptography and computer privacy[J]. Scientific

- American. 1973, 228(5): 15-23.
- [30] LI Y, GE G, XIA D. Chaotic hash function based on the dynamic S-box with variable parameters[J]. Nonlinear Dynamics. 2016, 84(4): 2387-2402.
- [31] XIAO D, LIAO X, DENG S. One-way hash function construction based on the chaotic map with changeable-parameter[J]. Chaos, Solitons & Fractals. 2005, 24(1): 65-71.
- [32] TEH J S, ALAWIDA M, HO J J. Unkeyed hash function based on chaotic sponge construction and fixed-point arithmetic[J]. Nonlinear Dynamics. 2020, 100(1): 713-729.
- [33] TEH J S, TAN K, ALAWIDA M. A chaos-based keyed hash function based on fixed point representation[J]. Cluster Computing. 2019, 22(2): 649-660.
- [34] XIAO D, LIAO X, DENG S. Parallel keyed hash function construction based on chaotic maps[J]. Physics Letters A. 2008, 372(26): 4682-4688.
- [35] HUANG W, WANG L. A hash function based on sponge structure with chaotic map for spinal codes[C]//2019 International Conference on Computer, Information and Telecommunication Systems (CITS). IEEE, 2019. 1-5.
- [36] ABDOUN N, EL ASSAD S, DEFORGES O, et al. Design and security analysis of two robust keyed hash functions based on chaotic neural networks[J]. Journal of Ambient Intelligence and Humanized Computing. 2020, 11(5): 2137-2161.
- [37] ALAWIDA M, SAMSUDIN A, ALAJARMEH N, et al. A novel hash function based on a chaotic sponge and DNA sequence[J]. IEEE Access, 2021, 9: 17882-17897.
- [38] ABDOUN N, EL ASSAD S, MANH HOANG T, et al. Designing two secure keyed hash functions based on sponge construction and the chaotic neural network[J]. Entropy, 2020, 22(9): 1012.
- [39] JOUX A. Multicollisions in iterated hash functions. Application to cascaded constructions[C]//Annual International Cryptology Conference. 2004: 306-316.
- [40] LUO Y L, DU M H. One-way hash function construction based on the spatiotemporal chaotic system[J]. Chinese Physics B. 2012, 21(6): 60503.
- [41] LIU H, KADIR A, LIU J. Keyed hash function using hyper chaotic system with time-varying parameters perturbation[J]. IEEE Access. 2019, 7: 37211-37219.
- [42] AHMAD M, KHURANA S, SINGH S, et al. A simple secure hash function scheme using multiple chaotic maps[J]. 3D Research. 2017, 8(2): 13.

[作者简介]



陈立全（1976—），男，广西玉林人，东南大学教授、博士生导师，主要研究方向为密码与安全协议、物联网安全。



朱宇航（1998—），男，安徽安庆人，东南大学硕士生，主要研究方向为数据安全与隐私保护，安全协议。



王宇（1995—），男，黑龙江哈尔滨人，东南大学博士生，主要研究方向为混沌图像加密、密码学。



秦中元（1974—），男，河南安阳人，博士，东南大学副教授，主要研究方向为人工智能安全、网络安全协议。



马阳（1980—），男，江苏南京人，东南大学博士生，主要研究方向为安全协议、网络流量分析、网络空间安全威胁识别、信息安全系统。