基于空间伸缩结构的参数可控的混沌 Hash 函数*

廖东[1)] 王小敏[1)†] 张家树[2)] 张文芳[1)]

1) (西南交通大学信息科学与技术学院, 成都 610031)

2) (西南交通大学, 信号与信息处理四川省重点实验室, 成都 610031)

(2012 年 4 月 26 日收到; 2012 年 6 月 13 日收到修改稿)

结合并行 Hash 函数和多混沌的设计思想, 提出了一种基于空间伸缩结构的参数可控的混沌 Hash 函数构造方法. 该方法结合空间结构的伸缩特性, 使用动态密钥控制消息在空间的 "膨胀 - 收缩 - 置乱" 方式, 有效地提高了系统的混乱和扩散特性, 同时使用空间并行结构在提高 Hash 函数安全性的同时也提高了系统的执行效率. 研究结果表明: 新算法的并行计算速度快, 且产生的 Hash 序列满足均匀分布, 具有更为理想的混淆与扩散特性.

关键词: Hash 函数, 混沌系统, 空间伸缩, 参数可控

PACS: 05.45.Gg, 05.50.+q

1 引 言

密码 Hash 函数是实现数据完整性保护、消息认证和数字签名的关键技术, 在现代信息安全领域中得到广泛应用. 常见的密码 Hash 函数有 MD5, SHA-1, SHA-256 等, 但随着模差分方法 [1,2] 对 MD5 和 SHA-1 的成功攻击, 基于传统 Hash 函数的密码系统的安全性受到了极大的威胁, 使得 Hash 函数的设计成为目前信息安全和密码学界关注的热点.

混沌是一种始终局限于有限区域且轨道不重复的形态复杂的非线性运动形式, 是非线性确定性系统内在随机性的表现, 可以由十分简单的确定系统产生复杂的随机行为. 由于混沌系统在迭代中的信息损失, 使得混沌序列的信息量渐进趋于零, 因此对混沌序列进行正确的长期预测是不可行的, 以上性质使得混沌序列拥有单向 Hash 函数所要求的较好的不可逆性、防伪造性、初值敏感性 [3].

文献 [3—6] 分别使用混沌系统与传统单向 Hash 函数、神经网络和高维混沌序列的复合滤波器系统的结合, 提出基于混沌系统的密码 Hash 函数, 相较传统密码 Hash 函数, 增加了 Hash 函数的安全性, 但由于混沌系统使用了大量的浮点运算, 导致这些迭代型混沌 Hash 函数的时效性不如传统密码 Hash 函数. 在文献 [7, 8] 中曾提出使用并行设计来提升迭代型混沌 Hash 函数的计算效率, 但算法存在一定的安全问题 [9,10].

为了克服混沌 Hash 函数的安全和效率问题, 本文将混沌系统与空间变换结合起来, 提出了基于空间伸缩结构的参数可控的混沌 Hash 函数. 算法利用混沌耦合格子映射产生的轮密钥控制立方格子在空间上的扭转方向和顺序, 并使用扩展帐篷映射完成消息的混淆和扩散, 然后通过空间伸缩方式实现块内消息的随机变换. 各消息块的压缩值 (即中间链值) 再经扩展帐篷映射运算后得到最终的 Hash 值. 由于空间伸缩的置换特性和混沌映射的初值敏感特性, 本文给出的 Hash 函数具有良好的单向性和抗碰撞性. 此外, 空间伸缩变换的局部并行性和消息块可并行压缩的特点, 使得算法具有较快的散列速度.

* 国家自然科学基金 (批准号: 60903202, 61003245)、教育部博士点基金 (批准号: 20090184120024)、四川省青年基金 (批准号: 2011JQ0027)、中央高校基本科研业务费专项资金 (批准号: SWJTU11CX041, SWJTU12CX099)、铁道部重大项目 (批准号: 2012X004-A) 和西南交通大学基础研究基金 (批准号: 2008B08) 资助的课题.

† E-mail: xmwang@home.swjtu.edu.cn

2 算法的基本部件

耦合映像格子 (coupled map lattices, CML)[6] 是由空间和时间离散但状态变量连续的多个混沌系统组成的阵列. 它可以在短时间内把系统状态耦合并扩散到各个格子里, 有效克服数字化混沌的有限精度效应并增强了系统的复杂性. 利用 CML 的这种特点, 文献 [11] 设计了一个性能优良的流密码. 本文则用 CML 来生成压缩函数中的轮密钥.

双向耦合映像格子的动力学模型为

$$\begin{aligned} &x_n(i+1)\\ =&(1-\varepsilon)g(x_n(i))\\ &+\frac{\varepsilon}{2}[g(x_{n-1}(i))+g(x_{n+1}(i))], \end{aligned} \tag{1}$$

其中, $g(\cdot)$ 为混沌映射, i 为离散时间步数; n 为格子点坐标; 耦合系数 ε 满足 $0<\varepsilon<1$. 文中 $g(\cdot)$ 采用如下的扩展帐篷映射 [12]:

$$\begin{aligned} &g_\alpha(x_{i-1}):x_i\\ =&\begin{cases} \dfrac{x_{i-1}}{a}, & 0<x_{i-1}<a,\\ \dfrac{1-x_{i-1}}{1-a}, & a\leqslant x_{i-1}<1,\\ \beta, & x_{i-1}=0 \ \text{ or } \ 1. \end{cases} \end{aligned} \tag{2}$$

其中, α, β 满足: $0<\alpha,\beta<1$, $\beta\neq\alpha$, $g_\alpha(\beta)\neq\alpha$.

3 基于空间伸缩结构的混沌 Hash 函数设计

3.1 压缩函数的轮密钥生成算法

将 128-bit 的种子密钥分割成等长的 4 个 32-bit 块并量化为小数, 记为 $K_j(0)\in[0,1)$, $j=1,2,3,4$. 假设消息填充后有 s 个消息分组, 按照 (3) 式生成第 i $(i=1,2,\cdots,s)$ 个消息分组的 4 个长度为 48 bits 的轮子密钥 $\mathrm{sub}K_j(i)$, $j=1,2,3,4$, 具体操作如图 1 所示.

$$\begin{aligned} KK_j(i)=&(K_j(i)+K_{j+1 \bmod 4}(i)) \bmod 1,\\ K_j(i)=&(1-\varepsilon)g\left(KK_j(i-1)\right)\\ &+\frac{\varepsilon}{2}\big[g(K_{j+2 \bmod 4}(i-1))\\ &+g(K_{j+3 \bmod 4}(i-1))\big],\\ \mathrm{sub}K_j(i)=&\left\lfloor 2^{48}\times K_j(i)\right\rfloor \lll (i+j \bmod 16), \end{aligned} \tag{3}$$

其中, $\lfloor\cdot\rfloor$ 为向下取整操作, $x\lll y$ 表示 x 循环左移 y 位, $\varepsilon=1/3$, $g(\cdot)$ 为 (3) 式的扩展帐篷映射, 参数 $\alpha=1/(\mathrm{Length}(M)+1)$, 周期边界条件 $K_0(i)=K_4(i)$.

轮密钥 $\mathrm{sub}K_j(i)$, $j=1,2,3,4$ 控制 3.2.2 节轮变换的第 j 轮, 具体控制方式由图 2 的子密钥比特分布决定.

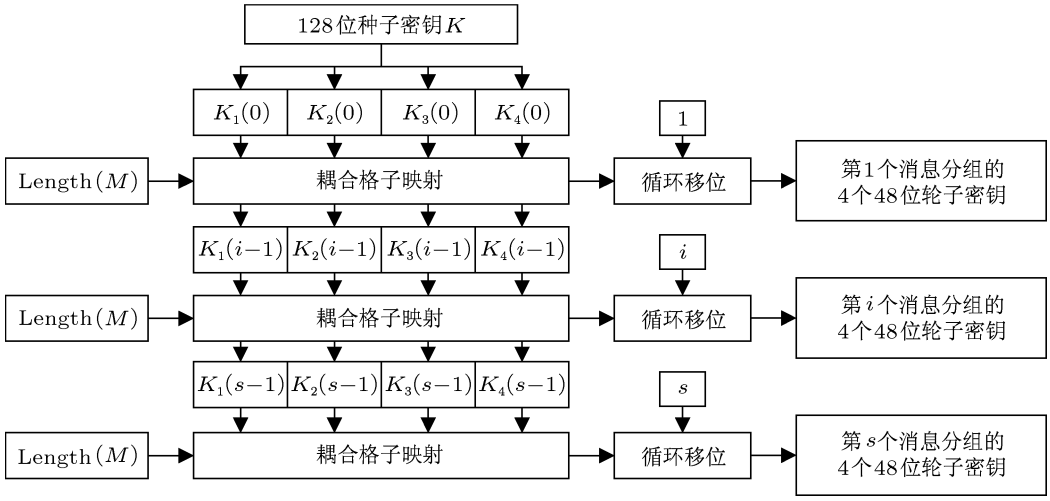

图 1 轮密钥生成方法

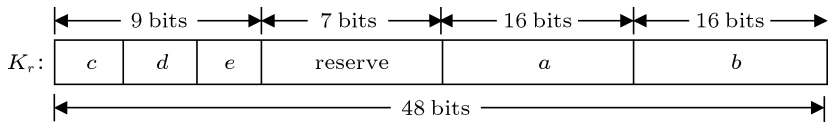

图 2 轮密钥 $\mathrm{sub}K_j(i)$ 的比特分布图

其中, c, d, e 均为 3-bit 字段, 由控制字 $C = d \oplus (c \wedge \bar{e})$ 按表 1 控制空间变换的 x, y, z 三个方向的迭代顺序. $a = a_1 a_2 \cdots a_{16}$ 和 $b = b_1 b_2 \cdots b_{16}$ 均为 16-bit字, 扩展为 32-bit字 $A = a_1 a_2 \cdots a_{16} a_{16} a_{15} \cdots a_1$ 和 $B = b_1 b_2 \cdots b_{16} b_{16} b_{15} \cdots b_1$ 用于控制 3.2 节中的压缩函数构造.

表 1 控制字与空间变换迭代顺序的关系

控制字 C	迭代顺序	控制字 C	迭代顺序
000	$x \to y \to z$	100	$y \to z \to x$
001	$x \to y \to z$	101	$z \to y \to x$
010	$x \to z \to y$	110	$z \to x \to y$
011	$y \to x \to z$	111	$z \to x \to y$

3.2 压缩函数的构造方法

3.2.1 消息预处理

采用 MD5 加固方式对消息 M 填充为 576-bit 的整数倍, 并分割成 576-bit 的消息分组 M_1, $M_2, \cdots M_s$, 其中 $M_s = * \cdots * 10 \cdots 0 || \mathrm{Length}(M)$, $\mathrm{Length}(M)$ 为 64 位二进制表示的原始消息 M 的长度.

将 576-bit 的消息分组表示成 18 个 32-bit 字 $M_i = (m_1, m_2, \cdots m_{18}), m_j \in [0, 2^{32})$, $j = 1, 2, \cdots, 18$, 按 (4) 式扩展为 27 个 32-bit 字 m_t, $t = 1, 2, \cdots, 27$, 然后按从上到下, 从左到右, 从前到后的顺序依次赋值给 3×3 空间立方结构的 27 个格子.

$$\begin{cases} m_t = m_{19-t} \oplus m, & 1 \leqslant t \leqslant 9 \\ m_t = (m_{t-9} \oplus m_{t-7} \oplus m_{t-4} \oplus m_{t-2}), & \\ \qquad 10 \leqslant t \leqslant 27, & \end{cases} \tag{4}$$

其中, $m = \left(m_j^1 \oplus m_j^2 \oplus \cdots \oplus m_j^{18} \oplus A\right) \ggg 10$.

3.2.2 轮变换

步骤 1 将 3×3 空间立方结构分别向 x, y, z 三个方向进行如图 3 所示的三维膨胀, 得到 3 个膨胀面. 膨胀后的立方格子记为 $B_{(i,j,k)}$, 表示 (i, j, k) 处的空间立方格子数据, $i, j, k = 1, 2, 3, 4$.

按 (5) 式得到在 x 方向上的膨胀面 $B_{(4,j,k)}$, $(j, k = 1, 2, 3)$. 同理, 可得到 y, z 方向的膨胀面 $B_{(i,4,k)}$, $(i, k = 1, 2, 3)$.

$$\begin{aligned} B_{(4,1,j)} =& (B_{(1,1,j)} \wedge B_{(2,1,j)}) \vee (\overline{B_{(1,1,j)}} \wedge B_{(3,1,j)}) \\ & \oplus B_{\mathrm{sum}}, \\ B_{(4,2,j)} =& (B_{(1,2,j)} \wedge B_{(3,2,j)}) \vee (B_{(2,2,j)} \wedge \overline{B_{(3,2,j)}}) \\ & \oplus \overline{B_{\mathrm{sum}}}, \\ B_{(4,3,j)} =& (B_{(1,3,j)} \oplus B_{(2,3,j)} \oplus B_{(3,3,j)}) \wedge B_{\mathrm{sum}}, \\ B_{\mathrm{sum}} =& \left(B + \sum_{i,j,k=1}^{3} B_{(i,j,k)} \right) \bmod 2^{32}, \end{aligned} \tag{5}$$

其中, B 为图 2 所示轮子密钥 $\mathrm{sub}K_j(i)$ 的部分.

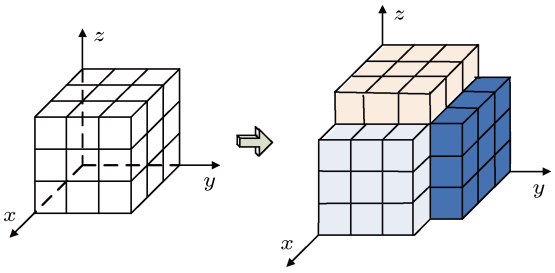

图 3 空间结构的三维消息膨胀

步骤 2 将膨胀后的所有立方格子数据转化为小数, 即 $B_{(i,j,k)} \leftarrow B_{(i,j,k)}/2^{32} \in [0, 1)$, 然后按照图 4 的方式将膨胀后的空间结构重新压缩为 3×3 标准模型, 其中 x, y, z 三个方向的收缩操作的执行顺序由控制字 C 按照表 1 决定.

在 x 方向, 膨胀面内的 9 个格子数据 $B_{(4,j,k)}$, $j, k = 1, 2, 3$ 按序编号为 $A, B, C, D, E, F, G, H, I$, x 方向的第三层格子 $B_{(3,j,k)}$, $j, k = 1, 2, 3$ 依序编号为 $B_1, B_2, \cdots, B_9$, 按图 4 进行混沌迭代. 类似地, x 方向的第二层格子 $B_{(2,j,k)}$ 和第一层格子 $B_{(1,j,k)}$ 分别编号为 $B_1, B_2, \cdots, B_9$ 后也进行同样方式的迭代. 这样就完成了 x 方向的格子由 4 层收缩为 3 层. y 方向和 z 方向的收缩方式与 x 方向类似, 此处不再赘述.

图中定义有两种运算:

$$x \oslash y = G_{\min(x,y)}(\max(x, y)),$$

$$x \boxplus y = x + y \bmod 1.$$

步骤 3 采用 (6) 式定义的 3-D Cat 映射 [13] 完成 3 维空间置乱, 同时将空间格子数据转换为 $[0, 2^{32})$ 整数, 即 $B_{(i,j,k)} \leftarrow 2^{48} \times B_{(i,j,k)} \bmod 2^{32}$.

$$\begin{bmatrix} i' \\ j' \\ k' \end{bmatrix} = \begin{bmatrix} 2 & 1 & 3 \\ 3 & 2 & 5 \\ 2 & 1 & 4 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \end{bmatrix} \bmod 3, \tag{6}$$

其中, (i, j, k) 为置乱前格子的位置, (i', j', k') 为置乱后格子的新位置.

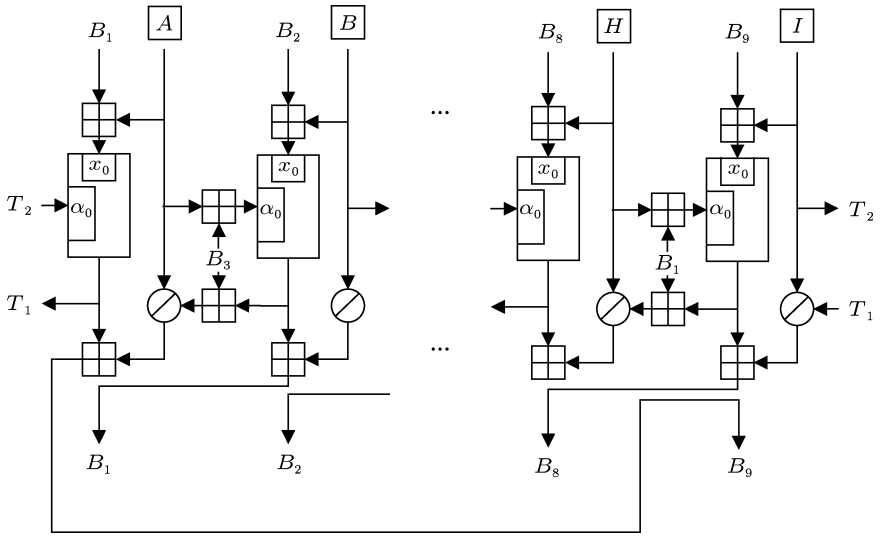

图 4 膨胀后的收缩结构示意图

注 $A, B, \cdots, I$ 为某方向 $(x, y,$ 或 $z)$ 膨胀面的 9 个格子数据, $B_1, B_2, \cdots, B_9$ 为相同方向上非膨胀层的 9 个格子数据.

将步骤 1—3 重复执行 R 次, 第 j 次轮变换中, 若 j 满足 $j \bmod 4 \neq 0$ 时使用轮子密钥 $\mathrm{sub}K_{j \bmod 4}(i)$, 否则使用 $\mathrm{sub}K_4(i)$. 压缩函数的结构如图 6(a) 所示.

3.2.3 输出变换

执行 R 轮后, 将立方格子数据转换为 27 个 $[0,1)$ 小数, 即 $B_{(i,j,k)} \leftarrow B_{(i,j,k)}/2^{32}$, 然后按图 5 方式对 27 个小数进行分层压缩 (27 个数对应图 5 的第一层数据). 压缩结构使用 (1) 式中的耦合格子映射, 其中扩展帐篷映射的参数 $\alpha = 1 - i/(\mathrm{Length}(M) + 1)$, i 表示消息分组的序号. 最后, 取第三层和第四层共 4 个 32 位小数作为当前消息分组的压缩结果, 记为 $C_i(j)$ $(i = 1, 2, 3, \cdots, s\,;\, j = 1, 2, 3, 4)$.

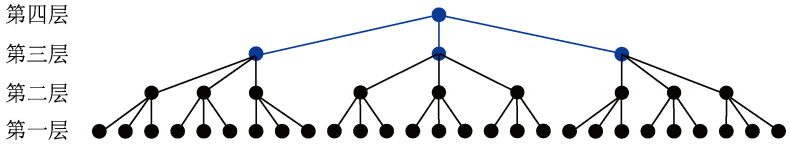

图 5 空间格子的分层压缩

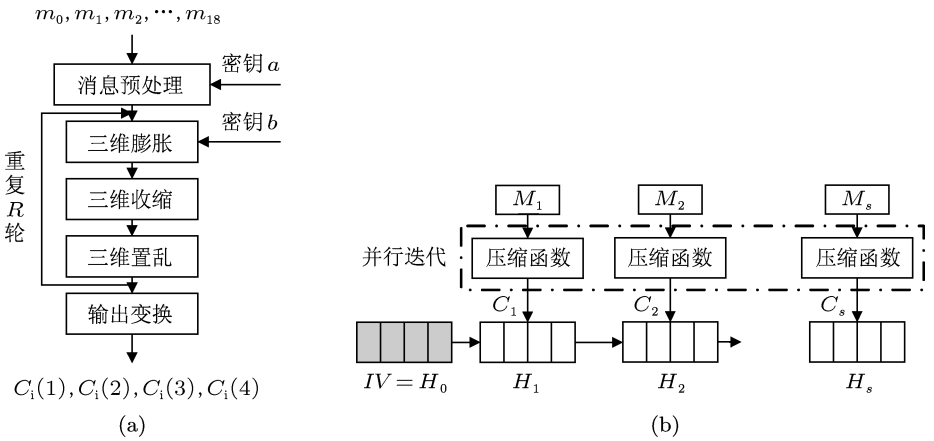

图 6 (a) 压缩函数的结构; (b) Hash 函数的整体结构

各消息块压缩后, 利用 (7) 式对 $C_i(j)$ 迭代 s 次得到 4 个 32 位的小数 $H_s(j)$, $j=1,2,3,4$, 将其转化为 32 位二进制整数, 即为 128 位 Hash 结果.

$$\begin{cases} H_0(j) = IV(j), \\ H_i(j) = g_\alpha(H_{i-1}(j), C_i(j)), \end{cases} \tag{7}$$

其中, $g_\alpha(\cdot)$ 为 (2) 式定义的扩展帐篷映射, $IV(j)$ 为 32 位初始小数, $H_i(j)$ 为中间链值, $i=1,2,3,\cdots,s; j=1,2,3,4$. 整个算法的结构如图 6(b) 所示.

4 Hash 函数的安全性及性能分析

4.1 并行结构的统计分析

本文算法采用空间伸缩结构完成 Hash 函数的压缩变换, Hash 函数的安全性主要依赖于压缩函数的安全性. 为了表述方便, 将经过 R 轮三维置乱的压缩函数记作 $Q^R(x)$, $Q^R(x)$ 的输出结果为 27 个 32-bit 字.

4.1.1 独立同分布特性分析

独立同分布特性是随机性统计分析中的一个重要的统计指标. 对本文的 $Q^R(x)$ 而言, 要求输入 x 发生微小变化时, 输出是相互独立的. 下面采用 χ^2 假设检验的方法来检验 $Q^R(x)$ 的独立同分布性, 具体步骤如下:

1) 将 (0, 1) 分为 m 个子区间 $[(i/m),(i+1/m)]$ $(i=0,1,\cdots,m-1)$;

2) 取 N 对 $Q^R(x)$ 和 $Q^R(x+\Delta x)$, 在 $m\times m$ 的连续格子中, 计算满足 $(i/m) < Q^R(x) < (i+1)/m$ 且 $(j/m) < Q^R(x+\Delta x) < (j+1)/m$ 的频数 n_{ij};

3) 计算

$$\chi^2 = N\left(\sum_{i=1}^{m}\sum_{j=1}^{m}\frac{n_{ij}^2}{\sum\limits_{j=1}^{m} n_{ij}\sum\limits_{i=1}^{m} n_{ij}} - 1\right). \tag{8}$$

测试中取 $m=64$, $N=10000$, Δx 为随机改变明文某一比特位, χ^2 拟合假设检验的结果如图 7.

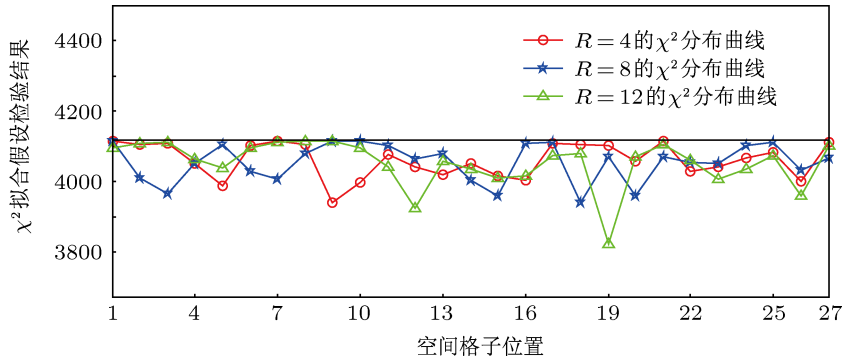

图 7 并行混沌迭代结构的独立同分布特性测试 (R 为迭代轮数)

结果表明: 当迭代轮数为 4, 8, 12 时, 输出结果的 χ^2 统计值均小于理论值 $\chi^2_{0.05}(63\times 63) = 4116.677$, 说明基于空间伸缩的压缩函数满足独立同分布特性. 综合考虑算法的效率和安全性, 选取迭代轮数 $R=4$.

4.1.2 随机分布特性分析

随机生成 10000 个样本空间, 在密钥固定且 $R=4$ 的条件下, 计算 Hash 值, 分析结果的每一个比特位, 计算其中为 "1" 的次数, 结果如图 8 所示.

图 8 的输出结果显示: 每一个比特为 "1" 的概率均接近 50%, 因此, 可以将并行迭代结构的输出等价于一个随机均匀分布.

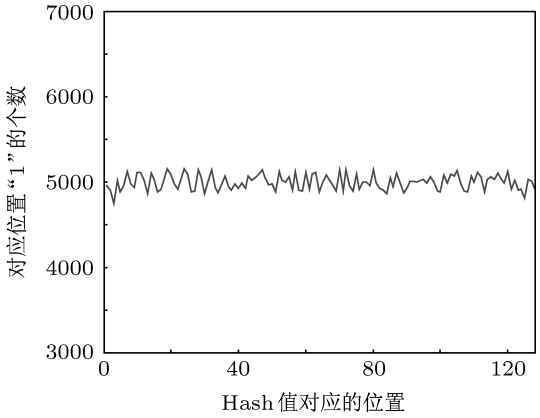

图 8 并行混沌迭代结构的随机分布特性测试

4.2 Hash 函数的安全性

上述的独立同分布特性和随机分布特性的测试结果表明, 4 轮并行迭代结构的输出在 Hash 空间呈均匀分布, 且输入与输出之间具有独立不相关性, 因此基于此并行迭代结构的 Hash 函数具有良好的混乱与扩散特性.

在 3.2.2 节中 B_9 的压缩函数可以表示为

$$T_1 = G(B_1 \boxplus A, I),$$

$$T_{B_1} = G(B_2 \boxplus B, A \boxplus B_3),$$

$$B_9 \leftarrow G(AT_{B_1} \boxplus B_3) \boxplus T_1).$$

同理得到 B_1—B_9 的计算公式, 其中 A—I 均由 B_i 和 subK_j, $j = 1, 2, 3, 4$ 作用的结果, 即并行迭代结构可以整理得:

$$\mathrm{New}B_i = Q^R(\mathrm{sub\ Key}_j, B_1, B_2, \cdots, B_9). \quad (9)$$

改变一个消息的最小比特位, 令 $m_1' = m_1 \oplus 1$, $m_i' = m_i$, $i = 2, 3, \cdots, 18$, 经过 3.2 节的 (4), (5) 式操作运算之后得到 $B_i' \neq B_i$, $i = 1, 2, \cdots, 27$, 说明通过非线性函数和多次的循环移位增强了系统的混淆和扩散特性.

由于 B_i, $i = 1, 2, \cdots, 27$ 均由消息块 M_j 得到, 故 (9) 式可以简化为: NewB_i = $Q^R(\mathrm{sub\ Key}_j, M_j)$, $j = 1, 2, \cdots, 27$, 其中 $Q^R(\cdot)$ 可以看作一个 576 位输入 32 位输出的非线性随机变换.

同理

$$\mathrm{New}B_i = Q^R(\mathrm{subKey}_j, M_j'), \quad j = 1, 2, \cdots, 27.$$

定理 1 如果 H 是一个 m bits 输入 n bits 输出的随机变换 ($m > n$), 并且得到的 n bits 输出是随机的, 那么任意取两个 m 比特的随机数 x_1 和 x_2, 则 $H(x_1) = H(x_2)$ 的概率为 $2^{-n} + 2^{-m} - 2^{-n-m}$ [14].

证明

如果 $x_1 = x_2$, 那么 $H(x_1) = H(x_2)$;

如果 $x_1 \neq x_2$, 那么 $H(x_1) = H(x_2)$ 的概率为 2^{-n}, 已知 $x_1 = x_2$ 的概率为 2^{-m}, 则 $x_1 \neq x_2$ 的概率为 $1 - 2^{-m}$;

综上所述, $H(x_1) = H(x_2)$ 的概率为 $2^{-m} + (1 - 2^{-m})2^{-n}$.

根据定理 1, NewB_i = NewB_i' 的概率为 $2^{-32} + 2^{-576} + 2^{-608}$, 其中 New$B_i$ 和 NewB_i' 对应比特位相等的概率为 $2^{-1} + 2^{-577}$, 最大偏离 $\delta = 2^{-577}$, 所以从一个完全随机的序列中区分出本算法的 NewB_i 的概率可以根据 (10) 式估算得出:

$$\Pr(x > -\delta\sqrt{\mathrm{NL}}) = \int_{-\delta\sqrt{\mathrm{NL}}}^{\infty} \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}x^2} \mathrm{d}x, \quad (10)$$

其中, NL 是实际尝试的次数. 计算得出, 经过 2^{1156} 次计算之后, 区分出 NewB_i 的可能性是 0.9772, 错误的概率是 0.0228.

上述分析都是基于单轮、单方向的操作, 当 $R = 4$ 且经轮密钥控制完成 x, y, z 三个方向进行 Hash 函数的运算, 将会大大加强算法的复杂性, 提高整个算法的安全性.

4.3 仿真实验及性能分析

4.3.1 Hash 值对明文和密钥的敏感性分析

理论上, 系统的混淆和扩散特性越好, 其相应的密钥和初值敏感性也越强. 对于 Hash 函数而言, Hash 结果的每一位只有 0、1 两种可能性, 因此理想的混乱与扩散效果是初值的任何细微变化都将导致结果的每一位以 50%的概率变化.

明文的敏感性测试: 固定 576 位明文消息, 每次改变明文消息的一位, 比较改变后的 Hash 值与改变前的变化. 分析结果如图 9 所示.

密钥的敏感性测试: 固定 128 位密钥, 每次改变密钥的一位, 比较改变后的 Hash 值与改变前的变化. 分析结果如图 10 所示.

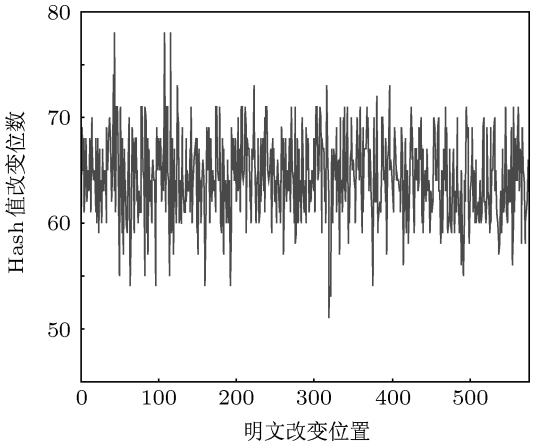

图 9 Hash 值对消息敏感性测试结果

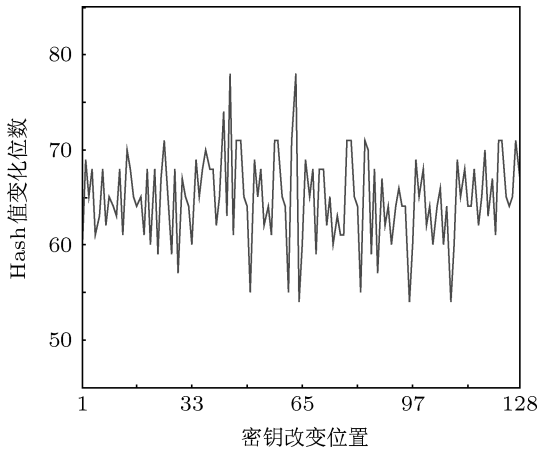

图 10 Hash 值对密钥敏感性测试结果

测试结果表明: Hash 值改变位数的变化均接近理想的 50%, 即是说, 明文消息和密钥的任一微小变换都会导致 Hash 值发生较大的变化, 进一步反映了所构造的 Hash 函数具有良好且稳定的明文和密钥敏感性, 保证了 Hash 函数可以有效地抵抗统计分析和选择明文攻击.

4.3.2 混乱与扩散性质统计分析

混乱与扩散被用来隐藏明文消息的冗余度, 加密体制中要求充分且均匀地利用密文空间, Hash 函数同样如此, 要尽量做到明文与对应的 Hash 密文不相关. 为了分析算法的混乱和扩散特性, 定义以下 4 个统计量:

平均变化比特数 $\overline{B}=\sum_{i=1}^{N}\frac{B_i}{N}$,

平均变化概率 $P=(\overline{B}/128)\times 100\%$,

B 的均方差 $\Delta B=\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(B_i-\overline{B}\right)^2}$,

P 的均方差 $\Delta P=\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(B_i/128-P\right)^2}$,

其中, N 为统计次数, B_i 为第 i 次测试结果变化的比特数.

初值敏感测试: 随机选取一段明文进行散列测试, 改变明文任意 1 bit 的值, 得到另一散列结果, 比较两者比特不同的数目.

密钥敏感性测试: 随机选取一段明文和密钥, 计算对应的 Hash 值, 保持明文不变, 改变密钥的 1 bit 数据, 比较两次输出的 Hash 值, N 次比较的结果如表 3 所示.

统计结果分析表明, 对于明文或者密钥的微小变化, 新算法下产生的 Hash 值平均变化比特数和平均变化概率都已非常接近理想状况, 相当充分和均匀地利用了密文空间. 而 $\Delta B, \Delta P$ 标志着散列混乱与散布性质的稳定性, 本文算法的 Δ 都已很小, 表明新算法对明文的混乱与散布能力强而稳定. 初值和密钥的极端敏感性, 以及平均稳定的散布性使攻击者无法得到任何有用的统计信息, 为抵御现有的已知密文攻击和差分线性攻击提供了很好的保证.

表 2 本文 Hash 算法初值敏感测试

试验次数 N	$\overline{B}$	P	ΔB	ΔP
10	65	50.78%	5.374838	4.20%
100	63.73	49.79%	6.500746	5.08%
1000	63.8872	49.88%	5.749042	4.49%
10000	63.8898	49.85%	5.627643	4.47%

表 3 本文 Hash 算法密钥敏感测试

试验次数 N	$\overline{B}$	P	ΔB	ΔP
10	62.3	48.67%	3.917199	3.06%
100	63.41	49.54%	4.648089	4.41%
1000	64.681	49.97%	4.325562	4.45%
10000	63.736	49.96%	4.239351	4.41%

4.3.3 扩散特性统计分析

完全性、雪崩效应和严格雪崩准则是评价一个分组密码算法扩散特性的重要指标. 所谓完全性是指算法输出的每一比特都与输入的所有比特有关; 所谓雪崩效应是指输入的任一比特改变都会造成输出平均半数比特的改变; 严格雪崩准则是指输入的任一比特改变都应导致输出每一比特以 1/2 的概率发生改变 [15].

为了描述方便, 给出一些定义. 用 $(\cdot)_j$ 表示序列的第 j 比特, $W_H(\cdot)$ 表示序列的汉明重量, $\#\{.\}$ 表示集合的势. 设 F 是一个 n 比特输入 m 比特输出的变换函数, 记输入序列 $x=(x_1,x_2,\cdots,x_n)$, 其中 $x_k\in\{0,1\}, k=1,2,\cdots$. 仅改变 x 的第 i 比特后的输入序列为 $x^i, i=1,2,\cdots,n$. 则对应的输出序列可以表示为 $F(x), F(x^{(i)})$.

设函数 F 的输入变量取自样本子集 $X\subset Z_2^n$, 记 $a_{ij}=\#\{x\in X|(F(x))_j\neq(F(x^{(i)}))_j\}$ $(i=1,2,\cdots,n;\ j=1,2,\cdots,m)$, 表示 X 中的列 x 和 $x^{(i)}$ 对应的输出序列之间第 j 比特不同的个数.

$b_{ij}=\#\{x\in X|W_H(F(x)\oplus(F(x^{(i)}))=j\}$

$(i=1,2,\cdots,n; j=1,2,\cdots,m)$, 表示 X 中的输入序列 x 和 $x^{(i)}$ 对应的输出序列之间的差分汉明重量为 j 的个数.

文献 [15] 中定义 d_1, d_2, d_3, d_4 四个统计量来表示算法非线性扩散程度的度量.

$$d_1 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{\#X}\sum_{x\in X} W_H\left(F(x)\oplus F(x^{(i)})\right)\right)$$

是雪崩效应程度的度量;

$d_2 = 1 - \#\{(i,j)|a_{ij} = 0,\ i = 1,\cdots,n;\ j=1,\cdots,m\}/(mn)$ 是完全性程度的度量;

$$d_3 = 1 - \frac{1}{n}\sum_{i=1}^{n}\left|\frac{1}{\#X\times m}\sum_{j=1}^{m} 2jb_{ij} - 1\right|$$

是雪崩效应程度的另一种度量;

$$d_4 = 1 - \left(\sum_{i=1}^{n}\sum_{j=1}^{m}\left|\frac{2a_{ij}}{\#X} - 1\right|\right)\Big/(mn)$$

是严格雪崩程度的度量.

文献 [15] 中指出: 若 $d_1 \approx m/2$, $d_2 = 1$, $d_3 \approx 1$, $d_4 \approx 1$, 则说明算法满足非线性扩散的基本要求, 具有很好的完全性和雪崩效应, 满足严格雪崩准则. 选取 N 个随机消息样本, 计算 Hash 函数的非线性扩散程度, 结果如表 4.

表 4 本文 Hash 函数的扩散性质测试结果

试验次数 N	d_1	d_2	d_3	d_4
10	63.27431	0.998427	0.976671	0.747238
100	63.30115	1.000000	0.99191	0.914139
1000	63.30841	1.000000	0.985102	0.966254
10000	63.50271	1.000000	0.99527	0.98191

4.3.4 抗碰撞性分析

碰撞是指虽然消息不同但其 Hash 值却相同, 即多对一映射. 随机选择一段明文, 将其 Hash 值保存为 ASCII 字符的形式. 然后随机改变明文消息中的一位, 保存其 Hash 值为 ASCII 字符的形式. 对两个 Hash 值进行比较, 计算它们在相同位置上 ASCII 字符相同的个数. 统计其中相同的次数. 碰撞的次数可以作为衡量抗碰撞性的指标 [16].

测试 10000 次, 同时计算出样本空间为 10000 的理论数值和测试值, 如图 11 所示.

绝对距离也是碰撞性测试的一个重要指标, 定义两个 Hash 值之间的绝对距离为

$$d = \sum_{i=1}^{N}|t(e_i) - t(e_i')|, \tag{11}$$

其中, e_i 和 e_i' 分别为两个 Hash 值中的第 i 个 ASCII 字符, 函数 $t(\cdot)$ 表示将 ASCII 字符转换为对应的数值. 重复上述过程 128 次, 最大、最小和绝对距离如表 5 所示.

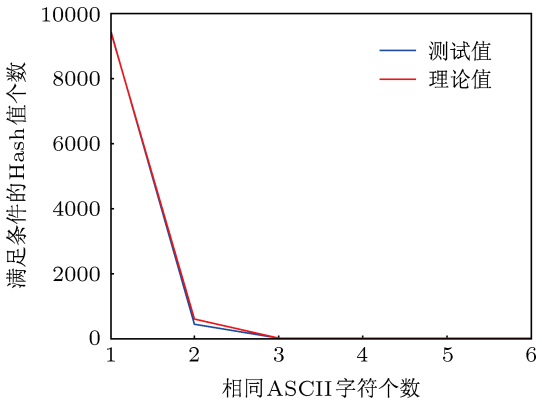

图 11 在相同位置上具有相同 ASCII 字符的 Hash 值个数分布

表 5 两个 Hash 值的绝对距离

	最大值	最小值	平均值	每字符平均
绝对距离	2457	446	1346.48550	84.1553438

平均值是 84.1553438, 比较接近理论值 85.33. 因此可以得出结论: 仅有一位不同的两段明文产生的 Hash 值是独立且随机分布的序列. 也就是说, 从 Hash 值的角度而言, 即使有 1 bit 不同, 那两段明文也是相互独立的.

4.3.5 算法效率分析

执行效率是 Hash 算法是否具有实际应用价值的重要标志之一. 基于混沌的 Hash 算法往往执行效率较低, 因而其实际应用受到制约. 本文算法基于耦合映像格子时空混沌模型, 与简单的低维混沌系统相比, 更容易达到较高的安全性. 此外, 该算法将明文并行注入每个空间格点, 明文之间通过密钥相互联系, 使整个系统易于并行实现, 从而降低了系统复杂性带来的时空开销, 达到了较高的执行效率.

5 结 论

本文提出了一种基于空间伸缩结构的参数可

控的单向 Hash 函数, 该方案借鉴时空混沌的特性, 结合空间结构的并行迭代函数对消息进行扩散, 从而在效率和安全性上完成对以往 Hash 函数的增强. 仿真结果和理论分析也表明: 1) 新算法具有良好的单向性和抗碰撞性; 2) 算法有效地避免了混沌短周期和有限精度的影响; 3) 新算法通过密钥控制字动态调制系统内部的计算模式, 结构高效安全, 产生的扩展消息具有良好的扩散和混淆特性; 4) 算法结构适合并行计算, 有效弥补了混沌映射计算效率的缺陷, 减少了 Hash 值计算时间.

[1] Wang X Y, Lai X J, Feng D G 2005 *Proceedings of Advances in Cryptology- Eurocrypt'05*, LNCS 3494, Springer-Verlag, May, 2005 p1

[2] Wang X Y, Yu H B 2005 *Proceedings of Advances in Cryptology-Eurocrypt'05* Springer-Verlag, Aartheis, Denmarh May, 2005 p19

[3] Guo W, Cao Y, Wang X M, He D K 2008 *Journal on Communications* **29** 93 (in Chinese) [郭伟, 曹杨, 王小敏, 何大可 2008 通信学报 **29** 93]

[4] Liu G J, Shan L, Dai Y W, Sun J S, Wang Z Q 2006 *Acta Phys. Sin.* **55** 5688 (in Chinese) [刘光杰, 单梁, 戴跃伟, 孙金生, 王执铨 2006 物理学报 **55** 5688]

[5] Wang X M, Zhang J S, Zhang W F 2005 *Acta Phys. Sin.* **54** 5566 (in Chinese) [王小敏, 张家树, 张文芳 2005 物理学报 **54** 5566]

[6] Wang X M, Zhang J S, Zhang W F 2003 *Acta Phys. Sin.* **52** 2736 (in Chinese) [王小敏, 张家树, 张文芳 2003 物理学报 **52** 2736]

[7] Xiao D, Liao X F, Deng S J 2008 *Phys. Lett.* A **372** 4682

[8] Xiao D, Liao X F, Wang Y 2009 *Neurocomputing* **72** 2288

[9] Wang X M, Guo W, Zhang W F, Khan M K, Alghathbar K 2011 *Telecommun. Syst.* DOI 10.1007/s11235-011-9457-9

[10] Guo W, Wang X M, He D K, Cao Y 2009 *Phys. Lett.* A **373** 3201

[11] Li P, Li Z, Halang W A, Chen G R 2007 *Chaos, Solitons and Fractals* **32** 1867

[12] Yi X 2005 *IEEE Trans. on Circuits and Systems* II **52** 354

[13] Chen G R, Mao Y B, Chui C K 2004 *Chaos, Solitons and Fractals* **21** 749

[14] Wang S H, Hu G 2012 *Information Sciences* **195** 266

[15] Zhu M F, Zhang B D, Lü S W 2002 *Journal on Communications* **23** 122 (in Chinese) [朱明富, 张宝东, 吕述望 2002 通信学报 **23** 122]

[16] Zhang J S, Wang X M, Zhang W F 2007 *Phys. Lett.* A **362** 439

The chaotic hash function based on spatial expansion construction with controllable parameters*

Liao Dong[1)] Wang Xiao-Min[1)†] Zhang Jia-Shu[2)] Zhang Wen-Fang[1)]

1) (*Institute of Information Science and Technology, Southwest Jiaotong University, Chengdu* 610031, *China*)

2) (*Key Laboratory of Signal and Information Processing of Sichuan Province, Southwest Jiaotong University, Chengdu* 610031, *China*)

(Received 26 April 2012; revised manuscript received 13 June 2012)

Abstract

A novel chaotic one-way hash function based on spatial expansion construction with controllable parameter is presented which combines with the advantages of both chaotic system and parallel hash function. In the proposed approach, the hash model of message block is determined by chaotic dynamic parameter. The new method improves the security of hash function and avoids degrading the system performance at the same time. Theoretical and experimental results show that the proposed method has high performance in parallel algorithm, nearly uniform distribution and desired diffusion and confusion properties.

Keywords: hash function, chaotic system, spatial expansion, controllable-parameter

PACS: 05.45.Gg, 05.50.+q

* Project supported by the National Natural Science Foundation of China (Grant Nos. 60903202, 61003245), the Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20090184120024), the Excellent Youth Foundation of Sichuan Province (Grant No. 2011JQ0027), the Fundamental Research Funds for the Central Universities (Grant No. SWJTU11CX041, SWJTU12CX099), the Major Science Technology and Development Program of Ministry of Railways of China (Grant No. 2012X004-A) and the Basic Science Foundation of Southwest Jiaotong University (Grant No. 2008B08).

† E-mail: xmwang@home.swjtu.edu.cn