Spinning Top:一种基于元胞自动机的新型序列密码算法

董有恒[1],张艳硕[1],鲁小娟[1],王彩冰[1],赵 耿[1],黄雅婷[2*]
(1. 北京电子科技学院密码科学与技术系,北京 100071;2. 兴唐通信科技有限公司,北京 100089)

摘 要: 针对现有序列密码算法存在非线性度不足、硬件实现效率受限等问题,本文基于初等元胞自动机(Elementary Cellular Automata,ECA)提出了一种在硬件平台上性能优于ZUC(ZU Chongzhi)算法,且安全可靠的“旋转陀螺”(Spinning Top)序列密码算法。研究首次将弹性混沌规则引入序列密码构造,有效兼顾了均衡性、相关免疫性与混沌性,从而确保输出序列具备优良的统计特性与高度不可预测性,并利用该类规则设计了三ECA循环嵌合结构,该结构在不采用S盒的情况下能利用有限的硬件资源保证算法的非线性度。相关分析和实验表明,算法能够通过NIST SP800-22与Test U01的相关测试,随机性达标;可抵御区分攻击与猜测—确定攻击,且具备前向/后向安全性。性能上,FPGA平台(Zynq-7000/200 MHz)吞吐率理论上能够达到25~40 Gbps,相同时钟周期下是ZUC算法能够达到的最大吞吐率7.1 Gpbs的3~5倍以上,且资源占用还比ZUC略少。该研究提供了利用ECA进行高非线性序列密码设计的范式,实践中可适配工业互联网、6G算力网络等场景,为资源受限设备提供“低资源-高安全”的加密方案。

关键词: 序列密码算法;元胞自动机;弹性函数;相关免疫性;伪随机数
基金项目: 中央高校基本科研业务费(No.3282024020,No.3282024019);国家重点研发计划(No.2024YFB3108105)
中图分类号: TN918 **文献标识码:** A **文章编号:** 0372-2112(2026)03-1263-17
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20250820

Spinning Top: A Novel Stream Cipher Constructed from Cellular Automata

DONG Youheng[1], ZHANG Yanshuo[1], LU Xiaojuan[1], WANG Caibing[1], ZHAO Geng[1], HUANG Yating[2*]
(1. *Department of Cryptography Science and Technology*, *Beijing Electronic Science and Technology Institute*, *Beijing* 100071, *China*;
2. *Xingtang Telecommunications Technology Co.*, *Ltd.*, *Beijing* 100089, *China*)

Abstract: To address the limitations of existing stream cipher algorithms in terms of insufficient nonlinearity and constrained hardware efficiency, this paper proposes a novel “Spinning Top” stream cipher algorithm based on elementary cellular automata (ECA). The proposed scheme demonstrates significantly higher performance than the standardized ZUC (ZU Chongzhi) algorithm on hardware platforms while ensuring strong security. For the first time, the concept of resilient chaotic rules is introduced into stream cipher design, jointly achieving balance, correlation immunity, and chaotic behavior. This ensures that the generated key-stream exhibits excellent statistical properties and high unpredictability. Furthermore, a cyclically embedded three-ECA structure is constructed, which ensures a high non-linearity without relying on S-boxes and with only limited hardware resources. Analytical evaluation and experimental results show that the proposed algorithm successfully passes the NIST SP800-22 and Test U01 suites, confirming statistical randomness. It also resists distinguishing attacks and guess-and-determine attacks, while providing both forward and backward security. In terms of performance, on the FPGA platform (Zynq-7000/200 MHz), the algorithm achieves a theoretical throughput of 25~40 Gbps, which is 3~5 times higher than the maximum throughput of 7.1 Gbps achieved by ZUC, with slightly lower resource consumption. This study establishes a new paradigm for high non-linearity stream cipher design using ECA, and in practice, it offers a low-resource yet high-security encryption solution suitable for application scenarios such as the Industrial Internet and 6G computing power networks.

Keywords: stream cipher algorithms; cellular automata; resilient function; correlation immunity; pseudo-random number
Foundation Item(s): The Fundamental Research Funds for the Central Universities (No.3282024020, No.3282024019); National Key Research and Development Program of China (No.2024YFB3108105)

收稿日期:2025-09-19;录用日期:2026-03-02;责任编辑:覃怀银
*通信作者:黄雅婷

0 引言

序列密码作为对称加密的重要分支,以其加解密速度快、错误传播有限、易于硬件实现等优点备受关注[1]。然而,要使序列密码既能满足严格的安全性要求又能在资源受限硬件平台上实现高吞吐率,一直是设计上的根本矛盾[2],提高安全性常伴随查表、S-盒或复杂代数操作,从而增加逻辑/存储开销;而减轻硬件开销又往往损失可证明或可测的抗攻击强度。

当前主流的序列密码算法各有特点:ChaCha20采用加法-循环移位-异或(Addition-Rotation-XOR,ARX)结构,具有256位密钥长、简单的轮函数和优秀的软件性能,但硬件实现需要较多的算术单元和流水线寄存器[3];Grain/Trivium等则基于线性反馈移位寄存器(Linear Feedback Shift Register,LFSR)结构,无S盒设计、仅用异或和移位,硬件面积极小且并行度高,但在软件上的每周期比特输出限制使得吞吐率相对较低[4]。ACORN作为轻量级认证加密算法,同样采用位运算结构(无S盒)并集成了消息认证功能,其硬件资源和能耗远低于等效的AES-GCM实现,且在软件中可通过并行运算获得较高速度[5-6]。综合来看,这些流密码的安全性、硬件复杂度、功耗和吞吐率各有侧重。ChaCha20由于轮函数设计合理,在传统差分/线性分析下安全性高,但其32位加法器和移位器在高并行硬件中代价较大[3];Grain/Trivium结构简单、功耗低,适用于资源受限环境,但极易受到立方攻击[7]。总体而言,现有流密码在不同应用场景中存在权衡:有的高吞吐但成本大,有的结构紧凑但速度有限,有的功能丰富但实现复杂。如何在保持高安全性的前提下进一步提高流水线并行度、降低硬件实现复杂度和功耗,是当前研究亟待解决的问题。面对未来6G、算力网络与工业互联网中对"低资源—高安全—高吞吐"并存的迫切需求,探索替代的构造范式具有重要意义。

初等元胞自动机(Elementary Cellular Automata,ECA)以其局部规则简单、并行性强、特定规则下能够呈现出混沌、随机等特性,为伪随机数发生器与序列密码设计提供了天然的候选结构[8-9]。目前,已有相关工作利用ECA构造伪随机数发生器或轻量级密码构件,但通常面临两类问题:一是单一ECA规则或简单并行结构难以实现足够的全局非线性与高阶相关免疫性,易被线性/相关分析弱化[10];二是在保证统计随机性与构建可分析安全性之间缺乏可扩展的设计手段,导致基于ECA的序列密码算法发展遇到了瓶颈。

为克服上述问题,本文提出"旋转陀螺"(Spinning Top)序列密码算法。其核心思想如下:其一,在规则层面选取具有均衡性与相关免疫性的特定ECA局部迭代规则,本文将此类规则命名为"弹性混沌规则";其二,在结构层面采用三重ECA循环嵌合+规则置乱+动态输出整合的设计,使得整体迭代过程在无需传统S盒的情况下即可产生高度非线性、接近理想线性复杂度且统计上不可区分于真随机序列的密钥流。同时该设计天然适配位并行硬件实现,在FPGA上实现将具备显著的单位资源吞吐率优势。具体的贡献如下。

(1)识别并证明了一些ECA的Class III规则[11]所指代的布尔函数同时满足均衡性与至少1阶的相关免疫性,并将这类规则定义为"弹性混沌规则"。以这些局部规则为基础,可以保证每一轮输出具有良好的统计均衡性与抗相关分析的特性。

(2)三层嵌合控制结构以产生全局非线性:提出一种循环控制机制——ECA1的输出控制ECA2的规则选择,ECA2控制ECA3,ECA3再返回控制ECA1;每个ECA在每轮由两种候选规则动态选择。本文形式上证明了即便单个局部规则为线性映射,三层互控结构在迭代后必然引入线性函数的乘积项,从而产生真正的多次可增长非线性项,非线性度的计算也表明算法的非线性度较高,提高了对线性攻击的抗性。

(3)动态规则置乱与整合输出以增强不确定性:在初始化与迭代过程中引入基于密钥驱动的规则表置乱、周期性循环移位与基于远距三个元胞状态的输出规则索引。该动态过程增加了由输出推导内部状态或规则排列的难度。

1 理论基础

1.1 初等元胞自动机

元胞自动机(Cellular Automata,CA)最早是由Von Neumann在20世纪40年代提出,起初是用来模拟自然界中生物自我复制现象的动力学系统[12]。随后,数学家Stanislaw Ulam与Von Neumann共同推动了CA的形式化研究,所提出的二维正方格网格和局部邻域的概念成为后续CA研究的经典框架。

CA是一种在时空和状态上都离散的动力学系统,该系统是由多个元胞通过一定方式排列组合而成,每个元胞在某一时刻处于有限状态之一,状态的演化和更新遵循一定的规则。一个标准的元胞自动机系统A由五元组构成:

$$A=(L,d,S,N,r) \tag{1}$$

其中,L为元胞空间;d为元胞空间的维数;S为元胞的有限状态集合;N为元胞邻居;r为局部迭代规则(下

文统称为迭代规则),此外CA若有边界则会存在边界条件,包括:固定边界,即元胞空间边界处的元胞,其邻居的状态值固定,并不随自动机的演化而变化;周期边界,即元胞空间的边界之间互为邻居;随机或伪随机边界,即边界处元胞的邻居状态跟随自动机的演化随机或伪随机地进行变换。

初等元胞自动机(Elementary Cellular Automata,ECA)是结构最简单的一类CA。ECA的元胞空间为一维等间距格点,可近似看作一个向量。元胞的状态仅有两种,可表示为{0,1}。元胞邻居为与当前元胞紧密相邻的两个元胞。ECA的结构如图1所示。

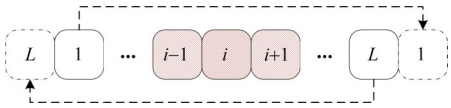

图1　ECA结构图

Figure 1　Structure of the ECA

图1为拥有L个元胞的ECA,元胞$i-1$和元胞$i+1$为元胞i的邻居。该ECA采用的是周期边界条件,即元胞空间边界处的第1个元胞和最后一个元胞L互为邻居。

ECA的局部迭代规则可表示为一个布尔函数:

$$x_i^{t+1}=f_r(x_{i-1}^t,x_i^t,x_{i+1}^t) \tag{2}$$

其中,x_i^t代表当前ECA中元胞空间的第i个元胞在第t个时刻的状态;x_{i-1}^t和x_{i+1}^t代表第i个元胞的元胞邻居在第t个时刻的状态;函数f_r为布尔函数,其输出结果仅为"0"或"1",而该结果不仅与当前元胞和其元胞邻居的状态相关,还与迭代规则r相关。例如当迭代规则r为90时,该布尔函数的真值表如表1所示。

表1　布尔函数f_{90}的真值表

Table 1　Truth table of the Boolean function f_{90}

x_{i+1}^t	1	1	1	1	0	0	0	0
x_i^t	1	1	0	0	1	1	0	0
x_{i-1}^t	1	0	1	0	1	0	1	0
x_i^{t+1}	0	1	0	1	1	0	1	0

表1的最后一行,即对于t时刻8种不同的输入$(x_{i-1}^t,x_i^t,x_{i+1}^t)$,该布尔函数在$t+1$时刻的输出刚好组成十进制数90的八位二进制表达形式:01011010b,"90"为该迭代规则下的编号,这种编号方式由计算机科学家Wolfram所引入[11]。显然对于ECA,共有$2^8=256$种迭代规则,不同的迭代规则使得ECA的演化呈现出不同的状态,Wolfram将这些规则分为四类:收敛到均匀状态(Class I),如规则0、255等;进入周期模式(Class II),如规则4、12等;表现出混沌与伪随机性(Class III),如规则30、90等;展现复杂局部结构的演化(Class IV),如规则110、124等。研究者们发现,某些Class III中的迭代规则具有较强的伪随机性和不可预测性,能够作为伪随机数发生器应用于流密码设计[13]。与此同时,ECA的并行性和结构简单性使其在硬件实现中具有天然优势。

1.2　t阶相关免疫的均衡布尔函数(t阶弹性布尔函数)

布尔函数是密码学中构造序列密码算法的重要工具之一。设:

$$f:\{0,1\}^n\to\{0,1\} \tag{3}$$

其中,f是n维输入的布尔函数,输出仅为"0"或"1"。对于序列密码设计,布尔函数的均衡性和非线性度是重点关注的两个核心性质。

均衡性:对所有可能的输入,布尔函数f的输出为"0"和"1"的次数相等,即

$$\left|\{x\in\{0,1\}^n:f(x)=0\}\right|=\left|\{x\in\{0,1\}^n:f(x)=1\}\right| \tag{4}$$

满足该等式的函数f为均衡布尔函数。此类布尔函数能够确保输出的比特流为等概分布,可以有效提高序列密码算法抗统计攻击的能力。

非线性度:布尔函数与所有仿射函数之间的最小汉明距离。较高的非线性度意味着攻击者难以用线性或仿射函数逼近该布尔函数,从而增强算法抗线性攻击的能力。

除上述两个性质外,在序列密码中布尔函数通常用于将多个线性反馈移位寄存器的输出组合为密钥序列。若布尔函数与部分输入变量之间存在线性相关性,则攻击者可利用该相关性实施相关攻击。为抵抗此类攻击,提出了相关免疫的概念。

t阶相关免疫性:设n维布尔函数$f(x_1,x_2,\cdots,x_n)$,若对输入向量$\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$中任意不超过$t$比特的子集,其分布与$f(x_1,x_2,\cdots,x_n)$的输出相互独立,则称该布尔函数$f$具有$t$阶相关免疫性。换言之,对于攻击者或密码分析者来说,当其获取不超过t比特的输入时,对于预测布尔函数的输出没有任何帮助。此外,相关免疫性和非线性度两者之间存在相互制约的关系,即对于一个布尔函数,其相关免疫的阶数t越大,则相对应的非线性度会越低。

一个布尔函数的相关免疫阶数t可通过计算Walsh变换来确定,定理如下。

定理1[14]　设有n维布尔函数f,则f是$t(1\leqslant t\leqslant n-1)$阶相关免疫布尔函数,当且仅当对所有使$1\leqslant W_H(\boldsymbol{w})\leqslant t$的$\boldsymbol{w}$,$F(\boldsymbol{w})=0$。

定理中$W_H(\boldsymbol{w})$代表二进制向量$\boldsymbol{w}$的汉明重量,且向量$\boldsymbol{w}$的长度和布尔函数f的输入$\boldsymbol{x}$相同,$F(\boldsymbol{w})$为

Walsh变换，具体表示如下：

$$F(\boldsymbol{w})=\sum_{\boldsymbol{x}} f(\boldsymbol{x})(-1)^{\langle \boldsymbol{x},\boldsymbol{w}\rangle} \tag{5}$$

其中，$f(\boldsymbol{x})$为上述布尔函数；$\langle \boldsymbol{x},\boldsymbol{w}\rangle$为向量$\boldsymbol{x}$和$\boldsymbol{w}$在二元域下的内积。

当一个布尔函数同时满足均衡性和t阶相关免疫性时，就称其为t阶相关免疫的均衡布尔函数，也称t阶弹性布尔函数[15]，这类布尔函数若能保证一定的非线性度，则在序列密码算法设计中将具有重要的应用价值，它能够确保设计出的序列密码具有抵抗统计分析，相关攻击以及线性分析攻击等的能力。

2 “旋转陀螺”(Spinning Top)序列密码算法

2.1 算法结构

算法核心由三个拥有相同元胞空间，但迭代规则不同的ECA组成。每个ECA由两个不同的迭代规则控制进行状态更新，每个元胞更新状态时所选用的迭代规则是由另一个ECA对应位置上的元胞状态所决定，同时三个ECA的迭代规则也会随着自动机的演化不断更新，这样三个ECA互相控制，相互影响，并不断输出统计特性良好的伪随机序列，由于其结构和演化形似“陀螺”在旋转，因此将其命名为“旋转陀螺”序列密码算法，下文简称“旋转陀螺”算法，结构如图2所示。

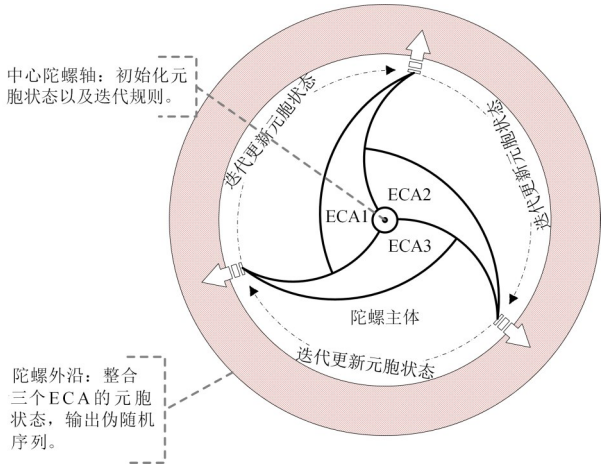

图2 “旋转陀螺”算法结构图

Figure 2 Structure of the “Spinning Top” algorithm

由图2可知，整个算法可分为静态和动态两个层面。

静态层面主要包含三个部分：“中心陀螺轴”即整个算法的控制和初始化中心，用于输入种子密钥和初始向量，并通过变换扩散到整个“陀螺主体”来初始化三个ECA中的元胞状态，同时控制陀螺的旋转形式，即初始化ECA的迭代规则；“陀螺主体”则由三个元胞空间相同的ECA构成，三个ECA之间互相作用，嵌合形成一个有机的主体；“陀螺外沿”用于保持整个陀螺的“平衡”，该部分通过整合三个ECA当前的状态，使输出序列具备更好的统计平衡、不可预测等随机性。

动态层面相当于陀螺的“旋转”，即整个算法运行的过程，如图3所示，其中白色方格代表当前元胞状态为“0”，阴影方格代表当前元胞状态为“1”，以ECA1控制ECA2的第i个元胞为例，迭代过程可用布尔函数表示如下：

$$y_i^{t+1}=\overline{x_i^{t+1}}\cdot f_{r1}(y_{i-1}^t,y_i^t,y_{i+1}^t)\oplus x_i^{t+1}\cdot f_{r2}(y_{i-1}^t,y_i^t,y_{i+1}^t) \tag{6}$$

其中，x_i^t、y_i^t分别代表ECA1和ECA2中第i个元胞在t时刻的状态；f_{r1}和f_{r2}为迭代规则$r1$与$r2$指代的布尔函数，含义同式(2)。由式(6)可知，$t+1$时刻ECA1中第i个元胞状态x_i^{t+1}为“0”时，仅会输出f_{r1}的结果，即令ECA2的第i个元胞由规则$r1$控制迭代，反之，仅会输出f_{r2}的结果，即令ECA2的第i个元胞由规则$r2$控制迭代，从而完成对ECA2迭代规则的控制。由此，“旋转方式”由迭代规则所决定，“陀螺旋转一周”意味着整个算法运行一轮，主体中的三个ECA完成元胞状态的一次更新，同时通过“陀螺外沿”整合输出本轮计算出的伪随机序列。

2.2 算法描述

本算法的输入为256 bit的种子密钥**Key**和128 bit初始向量**IV**，相应的三个ECA的元胞空间各含有256个元胞。具体的算法运行过程如下。

步骤1 初始化迭代规则表和ECA的元胞状态。首先是对迭代规则表进行置乱，本算法选择固定的八个迭代规则构成迭代规则表，如表2所示。

表2中所选的八个迭代规则按照十进制中从小到大的顺序排列，这些规则均属于类别Class III，具备混沌和随机性[11,16]，且指代的布尔函数均是具有相关免疫性的均衡函数，这是本算法的输出序列具备良好统计特性的基础，这一点将会在3.3节中进行详细证明。下面是迭代规则表置乱的算法，即算法1。该算法参考了RC4算法中所使用的Fisher-Yates置乱。

该算法建立了置乱后的迭代规则表$\boldsymbol{R}^*$和种子密钥**Key**之间的相关性，同时也使得后续迭代规则的使用具备一定的伪随机性。需要注意的是置乱后的**r_index**为本算法的全局变量，在步骤3中还会用到。

置乱完成后，利用$\boldsymbol{R}^*$中的前三个迭代规则和**Key**建立三个ECA的初始状态。设三个ECA分别为ECA1、ECA2、ECA3，三者的元胞状态分别用向量$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$表示。以ECA1为例，其某一时刻的迭代过程用

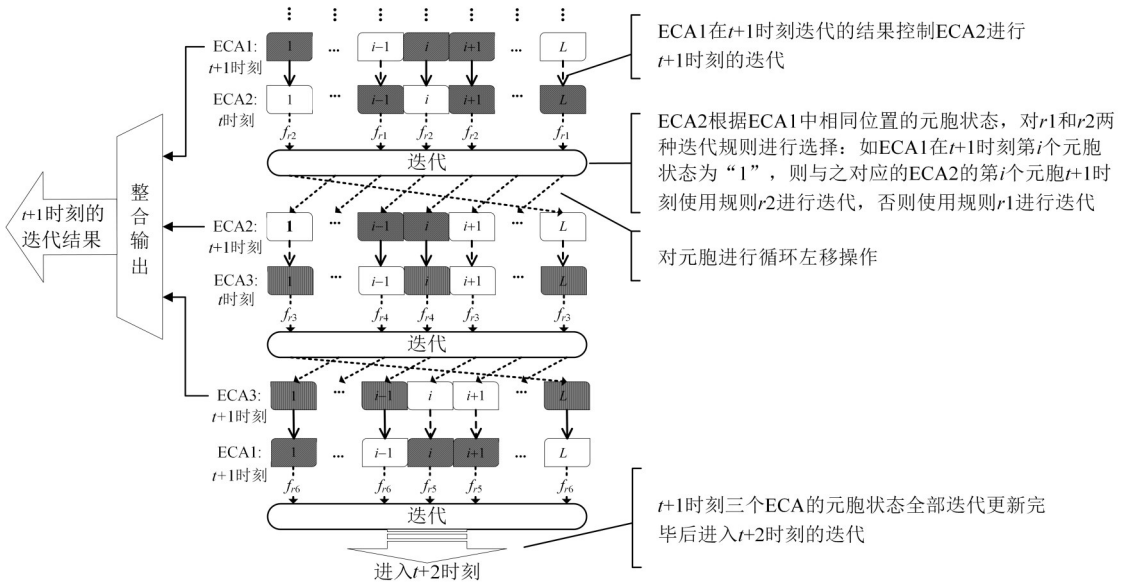

图3　“旋转陀螺”算法运行图

Figure 3　Operation of the “Spinning Top” algorithm

表2　迭代规则表

Table 2　Iteration rules

序号	1	2	3	4	5	6	7	8
迭代规则	60	90	102	105	150	153	165	195

算法1　迭代规则表置乱算法

输入：种子密钥**Key**（256 bit二进制数）、迭代规则表$\boldsymbol{R}\leftarrow$[60,90,102,105,150,153,165,195]

输出：置乱后的迭代规则$\boldsymbol{R}^*$

算法步骤：

1. 划分种子密钥得到$\boldsymbol{K}$,并初始化索引向量**r_index**:

$\boldsymbol{K}\leftarrow$将**Key**按高至低位划分为8个32位二进制数,并转化为8个十进制整数。

r_index $\leftarrow$[1, 2, ⋯, 8]

2. 利用得到的$\boldsymbol{K}$对**r_index**循环置乱：

for i = 8递减至2，步长为−1，执行：

$r\leftarrow \boldsymbol{K}(9-i)$

$j\leftarrow(r \bmod i)+1$

交换序号向量**r_index**中的元**r_index**(i)与**r_index**(j)

循环结束

3. 按照置乱后的**r_index**对$\boldsymbol{R}$执行置乱:$\boldsymbol{R}^*=\boldsymbol{R}$(**r_index**)

函数F表示如下：

$$\boldsymbol{A}_{t+1}=F(\boldsymbol{A}_t,R_i^*) \tag{7}$$

其中，$\boldsymbol{A}_t$为ECA1在第t个时刻的元胞状态；$\boldsymbol{R}_i^*$表示本次迭代所使用的规则为$\boldsymbol{R}^*$中的第i个迭代规则。将**Key**的低128位与**IV**进行异或并作为元胞状态向量，三个ECA的初始化过程表示如下：

$$\begin{cases}\text{ECA1:}\\ \quad \boldsymbol{A}=F(\mathbf{Key}\oplus\mathbf{IV},R_1^*)\\ \quad \boldsymbol{A}_0=F^{256}\left(\left[\boldsymbol{A}(3\colon 256),\boldsymbol{A}(1\colon 2)\right],R_1^*\right)\\ \text{ECA2:}\\ \quad \boldsymbol{B}=F(\boldsymbol{A}_0\oplus\mathbf{IV},R_2^*)\\ \quad \boldsymbol{B}_0=F^{256}\left(\left[\boldsymbol{B}(3\colon 256),\boldsymbol{B}(1\colon 2)\right],R_2^*\right)\\ \text{ECA3:}\\ \quad \boldsymbol{C}=F(\boldsymbol{B}_0\oplus\mathbf{IV},R_3^*)\\ \quad \boldsymbol{C}_0=F^{256}\left(\left[\boldsymbol{C}(3\colon 256),\boldsymbol{C}(1\colon 2)\right],R_3^*\right)\end{cases} \tag{8}$$

其中，F^{256}表示在该条件下循环迭代256次；$\boldsymbol{A}_0$、$\boldsymbol{B}_0$和$\boldsymbol{C}_0$分别为初始化完成后ECA1、ECA2和ECA3的元胞状态，具体每个元胞的迭代过程可参考式(2)。使用3个不同的迭代规则对初始状态**Key**⊕**IV**进行迭代，以此得到各个ECA的初始状态可以较为有效地掩盖种子密钥的信息，利用迭代过程中的循环左移操作可以有效提升整个算法的扩散性，使得**Key**和**IV**的每一位都能够影响到所有元胞的状态。

步骤2　迭代更新ECA状态。每个ECA将由两不同的迭代规则控制，由图3可知，三个ECA的迭代和状态更新是相互影响的。具体迭代的算法如算法2所示。

算法2是ECAi控制ECAj进行一次迭代的过程，其中循环左移的目的是在后续循环迭代过程中增强整个算法的扩散性。同时，边界条件采用的是伪随机的，边界状态是由ECAj当前时刻两端的元胞状态和

算法 2 ECAi (i= 1, 2, 3)控制 ECAj (j = 2, 3, 4)进行迭代

输入：$t+1$ 时刻 ECAi 的元胞状态 $\boldsymbol{I}_{t+1}$，t 时刻 ECAj 的元胞状态 $\boldsymbol{J}_t$，$t+1$ 时刻 ECAj 迭代所使用的迭代规则 $\{r0, r1\} \in \boldsymbol{R}^*$。

输出：$t+1$ 时刻 ECAj 的迭代结果 $\boldsymbol{J}_{t+1}$

算法步骤：

1. 初始化迭代规则，并对 $\boldsymbol{J}_t$ 进行一次循环左移：

$\{\boldsymbol{R0}, \boldsymbol{R1}\}$←将 $\{r0, r1\}$ 转换为八位二进制数，构成两个八元素比特向量(左为低位)

$\boldsymbol{J}_t \leftarrow \boldsymbol{J}_t(2, 3, 4, \cdots, 255, 256, 1)$

2. 计算 ECAj 的边界状态：

index_1 ← 取 $\boldsymbol{J}_t$ 的前 3 个元素构成一个 3 位二进制数，并转化为十进制整数 + 1

index_end ← 取 $\boldsymbol{J}_t$ 的最后 3 个元素构成一个 3 位二进制数，并转化为十进制整数 + 1

$j_1^* \leftarrow \boldsymbol{R0}(\text{index\_1})$

$j_{\text{end}}^* \leftarrow \boldsymbol{R1}(\text{index\_end})$

3. 构造邻居序列：

$\boldsymbol{J}_1 \leftarrow \left[j_1^*, \boldsymbol{J}_t(1, 2, \cdots, 255)\right]$

$\boldsymbol{J}_{\text{end}} \leftarrow \left[\boldsymbol{J}_t(2, 3, \cdots, 256), j_{\text{end}}^*\right]$

4. 计算索引：

$\mathbf{INDEX} \leftarrow 1 + 4 \times \boldsymbol{J}_1 + 2 \times \boldsymbol{J}_t + \boldsymbol{J}_{\text{end}}$(十进制下的矩阵运算)

5. 对第 $k(k = 1, 2, 3, \cdots, 256)$ 个元胞进行如下计算：

若 $\boldsymbol{I}_{t+1}(k) = 0$，则 $\boldsymbol{J}_{t+1}(k) \leftarrow \boldsymbol{R0}\left(\mathbf{INDEX}(k)\right)$(即取 $\boldsymbol{R0}$ 中的第 $\mathbf{INDEX}(k)$ 个元素)

若 $\boldsymbol{I}_{t+1}(k) = 1$，则 $\boldsymbol{J}_{t+1}(k) \leftarrow \boldsymbol{R1}\left(\mathbf{INDEX}(k)\right)$(即取 $\boldsymbol{R1}$ 中的第 $\mathbf{INDEX}(k)$ 个元素)

6. 当 ECAj 的所有元胞计算完成后输出 $\boldsymbol{J}_{t+1}$

两个迭代规则共同决定，增强了整个算法的随机性。通过算法 2 可将“旋转陀螺”中的三个 ECA 有机结合在一起，从而增强整个算法的非线性度。整个序列密码算法的一轮运算通过三次调用算法 2 来实现，具体过程如算法 3 所示。

步骤 3 整合输出一轮运算的伪随机序列，并更新迭代规则表。这一过程需要用到步骤 2 中算法 3 的输出，即 ECA1、ECA2 和 ECA3 更新后的元胞状态 $\boldsymbol{A}_{t+1}$、$\boldsymbol{B}_{t+1}$ 和 $\boldsymbol{C}_{t+1}$，以及当前的迭代规则表 $\boldsymbol{R}^*$。具体过程如算法 4 所示。

算法 4 对三个 ECA 在 $t+1$ 时刻的元胞状态进行整合，并利用 ECA 的迭代方式得到序列 $\mathbf{KS}_{t+1}$，且迭代规则 $\boldsymbol{r}$ 由元胞位置上距离较远的三个 ECA 的三个元胞共同决定，这种整合方式可以有效掩盖当前时刻三个 ECA 的内部状态。每轮运算的最后均要更新 $\boldsymbol{R}^*$，从而使迭代规则表中所有规则均能够被使用到，且保证每轮各个 ECA 所使用的两个迭代规则均在变化，提高整个算法的不可预测性。

步骤 3 完成后，返回步骤 2 进行下一轮迭代，以此循环，直到输出所需长度的密钥流。

图 4 为算法运行过程中截取的时空演化结果，包括同一时间段三个 ECA 以及最终输出的演化结果。图中横轴表示空间(256 个元胞)，纵轴表示时间变化(迭代 300 次)，黑(白)色点表示当前元胞状态为“0”(“1”)，该图呈现出无规律的雪花噪点状态，直观展现了算法良好的随机性。

算法 3 三个 ECA 的元胞状态更新算法

输入：ECA1、ECA2 和 ECA3 在 t 时刻的元胞状态 $\boldsymbol{A}_t$、$\boldsymbol{B}_t$ 和 $\boldsymbol{C}_t$，迭代规则表 $\boldsymbol{R}^*$

输出：ECA1、ECA2 和 ECA3 在 $t+1$ 时刻的元胞状态 $\boldsymbol{A}_{t+1}$、$\boldsymbol{B}_{t+1}$ 和 $\boldsymbol{C}_{t+1}$

算法步骤：

1. 更新 ECA1 的元胞状态：

$\{r0, r1\} \leftarrow \left\{\boldsymbol{R}^*(1), \boldsymbol{R}^*(2)\right\}$

$\{\boldsymbol{I}_{t+1}, \boldsymbol{J}_t\} \leftarrow \{\boldsymbol{C}_t, \boldsymbol{A}_t\}$

$\boldsymbol{A}_{t+1}$←输入 $(\boldsymbol{I}_{t+1}, \boldsymbol{J}_t, r0, r1)$ 调用算法 2 得到下一时刻元胞状态

2. 更新 ECA2 的元胞状态：

$\{r0, r1\} \leftarrow \left\{\boldsymbol{R}^*(3), \boldsymbol{R}^*(4)\right\}$

$\{\boldsymbol{I}_{t+1}, \boldsymbol{J}_t\} \leftarrow \{\boldsymbol{A}_{t+1}, \boldsymbol{B}_t\}$

$\boldsymbol{B}_{t+1}$←输入 $(\boldsymbol{I}_{t+1}, \boldsymbol{J}_t, r0, r1)$ 调用算法 2 得到下一时刻元胞状态

3. 更新 ECA3 的元胞状态：

$\{r0, r1\} \leftarrow \left\{\boldsymbol{R}^*(5), \boldsymbol{R}^*(6)\right\}$

$\{\boldsymbol{I}_{t+1}, \boldsymbol{J}_t\} \leftarrow \{\boldsymbol{B}_{t+1}, \boldsymbol{C}_t\}$

$\boldsymbol{C}_{t+1}$←输入 $(\boldsymbol{I}_{t+1}, \boldsymbol{J}_t, r0, r1)$ 调用算法 2 得到下一时刻元胞状态

算法 4 整合输出算法

输入：元胞状态 $\boldsymbol{A}_{t+1}$、$\boldsymbol{B}_{t+1}$ 和 $\boldsymbol{C}_{t+1}$，迭代规则表 $\boldsymbol{R}^*$

输出：$t+1$ 时刻的伪随机比特向量 $\mathbf{KS}_{t+1}$，置乱后的迭代规则表 $\boldsymbol{R}^*$

算法步骤：

1. 将三个元胞状态进行整合，计算索引矩阵 **INDEX**，得到输出迭代规则 $\boldsymbol{r}$：

$\mathbf{INDEX} \leftarrow 1 + 4 \times \boldsymbol{A}_{t+1} + 2 \times \boldsymbol{B}_{t+1} + \boldsymbol{C}_{t+1}$(十进制下的矩阵运算)

$\text{index} \leftarrow 1 + 4 \times \boldsymbol{A}_{t+1}(1) + 2 \times \boldsymbol{B}_{t+1}(86) + \boldsymbol{C}_{t+1}(171)$

$\boldsymbol{r}$←取 $\boldsymbol{R}^*$ 的第 index 个元素转换为八位二进制数，构成八元素比特向量(左为低位)

2. 对于 $\mathbf{KS}_{t+1}$ 的第 $k(k = 1, 2, 3, \cdots, 256)$ 个比特：

$\mathbf{KS}_{t+1}(k) \leftarrow \boldsymbol{r}\left(\mathbf{INDEX}(k)\right)$

3. 更新迭代规则表 $\boldsymbol{R}^*$：

若 $6 \times (t+1) \bmod 8 = 0$，则 $\mathbf{temp} = \boldsymbol{R}^*(\mathbf{r\_index})$(注：**r_index** 由步骤 1 中置乱得到)

若 $6 \times (t+1) \bmod 8 \neq 0$，则 $\mathbf{temp} = \left[\boldsymbol{R}^*(7), \boldsymbol{R}^*(8), \boldsymbol{R}^*(1, 2, \cdots, 6)\right]$

$\boldsymbol{R}^* \leftarrow \mathbf{temp}$

2.3 算法特性

2.3.1 均衡性和相关免疫性

“旋转陀螺”算法的均衡性和相关免疫性由所选的八个迭代规则(详见表 2)所保证。

根据 2.1 节对 ECA 的描述，在 ECA 的 256 个迭代规则中，具有均衡性的迭代规则共有 $C_8^4 = 70$ 个，这些

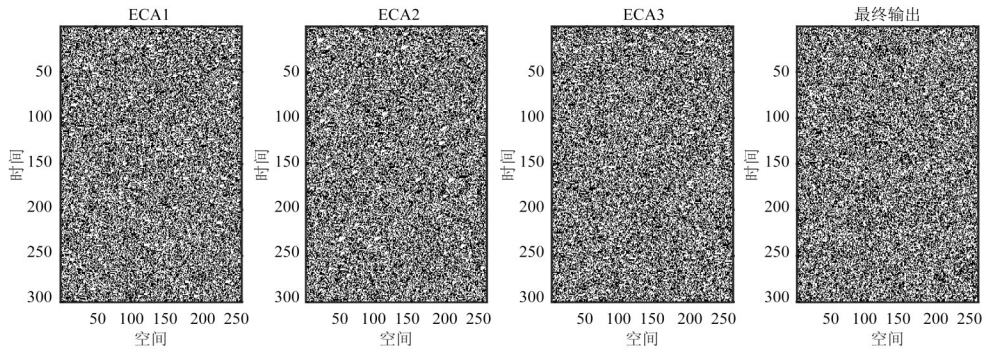

图4　“旋转陀螺”序列密码中元胞自动机的时空演化图

Figure 4　Spatiotemporal evolution of the cellular automata in the “Spinning Top” stream cipher

规则对八种不同的输入,输出为“1”和“0”的数量相等,且这些规则在转化为二进制后,汉明重量均为4,本文称这些规则为均衡规则。在均衡规则中,并非所有的规则都具备良好的混沌和随机性。本算法中所选的八个规则都属于均衡规则,同时这八个规则还属于Class III,具备良好的随机性[17]。

在具备均衡性的同时,这些迭代规则对应的布尔函数还具备相关免疫性,证明过程如下。

证明　ECA利用八个不同迭代规则的迭代过程可表示为式(2)的布尔函数运算,这些布尔函数的真值表如表3所示。

由表3和式(5)计算这八个布尔函数的Walsh变换结果。根据定理1,相关免疫阶数t应小于布尔函数输入向量的维数$1\leqslant t\leqslant n-1$,此处$n=3$,对于二进制向量$\boldsymbol{w}$,其汉明重量应满足$1\leqslant W_H(\boldsymbol{w})\leqslant 2$。计算结果如表4所示。

由表4可知,八个布尔函数在$W_H(\boldsymbol{w})=1$时Walsh变换的结果均为0,在$W_H(\boldsymbol{w})=2$时,迭代规则105和150对应布尔函数的Walsh变换结果依旧为0。由定理1可知,这八个布尔函数相关免疫阶数至少为1。故“旋转陀螺”算法使用的迭代规则所对应的布尔函数均具备相关免疫性。证毕

综上所述,本算法采用的八个迭代规则对应的布尔函数运算均同时具备:均衡性、相关免疫性、混沌和随机性(属于Class III)。相关研究[16-18]已证实对于ECA的256个迭代规则,有且仅有这八个规则满足上述条件,参考密码学中对弹性布尔函数的定义,本研究对这八个迭代规则单独定义为一类。定义如下。

表3　八个迭代规则对应的布尔函数真值表

Table 3　Truth tables of the Boolean functions corresponding to the eight iteration rules

输入	x_{i+1}^t	1	1	1	1	0	0	0	0
	x_i^t	1	1	0	0	1	1	0	0
	x_{i-1}^t	1	0	1	0	1	0	1	0
输出 $f_r(x_{i-1}^t,x_i^t,x_{i+1}^t)$	$r=60$	0	0	1	1	1	1	0	0
	$r=90$	0	1	0	1	1	0	1	0
	$r=102$	0	1	1	0	0	1	1	0
	$r=105$	0	1	1	0	1	0	0	1
	$r=150$	1	0	0	1	0	1	1	0
	$r=153$	1	0	0	1	1	0	0	1
	$r=165$	1	0	1	0	0	1	0	1
	$r=195$	1	1	0	0	0	0	1	1

定义1　对于初等元胞自动机的局部迭代规则,所对应的布尔函数同时具备均衡性和相关免疫性,且属于Class III类的迭代规则,统称为弹性混沌规则。

弹性混沌规则的三个性质使得“旋转陀螺”算法

表4　八个迭代规则对应的布尔函数Walsh变换的结果

Table 4　Walsh transform results of the Boolean functions corresponding to the eight iteration rules

汉明重量		1			2		
$\boldsymbol{w}$		001	010	100	011	101	110
布尔函数f_r的Walsh变换$F(\boldsymbol{w})$	$r=60$	0			0	0	−4
	$r=90$	0			0	−4	0
	$r=102$	0			−4	0	0
	$r=105$	0			0	0	0
	$r=150$	0			0	0	0
	$r=153$	0			4	0	0
	$r=165$	0			0	4	0
	$r=195$	0			0	0	4

具备抵抗相关分析和统计分析的能力，使得算法输出的序列拥有良好的伪随机性和不可预测性。

2.3.2 非线性

八个迭代规则所对应的虽为弹性布尔函数，然而，通过化简可发现这些布尔函数均为线性函数，如单独用来生成密钥流，很容易通过线性分析进行破解。所以在本算法的设计中，采用了ECA相互控制的嵌合结构，该结构可以保证算法的非线性。根据算法步骤2，不考虑边界条件和移位操作，三个ECA中单个元胞迭代的过程如式(9)~(11)。

$$a_i^{t+1}=\overline{c_i^t}\cdot f_{r1}(a_{i-1}^t,a_i^t,a_{i+1}^t)\oplus c_i^t\cdot f_{r2}(a_{i-1}^t,a_i^t,a_{i+1}^t) \tag{9}$$

$$b_i^{t+1}=\overline{a_i^{t+1}}\cdot f_{r3}(b_{i-1}^t,b_i^t,b_{i+1}^t)\oplus a_i^{t+1}\cdot f_{r4}(b_{i-1}^t,b_i^t,b_{i+1}^t) \tag{10}$$

$$c_i^{t+1}=\overline{b_i^{t+1}}\cdot f_{r5}(c_{i-1}^t,c_i^t,c_{i+1}^t)\oplus b_i^{t+1}\cdot f_{r6}(c_{i-1}^t,c_i^t,c_{i+1}^t) \tag{11}$$

其中，a_i^t、b_i^t和c_i^t分别代表ECA1、ECA2和ECA3中第i个元胞在t时刻的状态。函数$f_{ri}(i=1,2,\cdots,6)$表示迭代规则$ri(ri\in\boldsymbol{R}^*)$对应的布尔函数，详见式(2)。以式(9)为例，其含义是指若ECA3的第i个元胞在时刻t的状态c_i^t为“0”(“1”)，则当前时刻ECA1的第i个元胞迭代时使用规则$r1(r2)$，元胞状态经过迭代由状态a_i^t变为a_i^{t+1}，式(10)和(11)含义类似。若将式(10)带入到式(11)中可以得到式(12)。

$$\begin{aligned}c_i^{t+1}=&\left[\overline{a_i^{t+1}}\cdot f_{r3}(b_{i-1}^t,b_i^t,b_{i+1}^t)\oplus a_i^{t+1}\cdot f_{r4}(b_{i-1}^t,b_i^t,b_{i+1}^t)\right]\cdot f_{r5}(c_{i-1}^t,c_i^t,c_{i+1}^t)\\&\oplus\left[\overline{a_i^{t+1}}\cdot f_{r3}(b_{i-1}^t,b_i^t,b_{i+1}^t)\oplus a_i^{t+1}\cdot f_{r4}(b_{i-1}^t,b_i^t,b_{i+1}^t)\right]\cdot f_{r6}(c_{i-1}^t,c_i^t,c_{i+1}^t)\end{aligned} \tag{12}$$

显然，尽管f_{ri}均为线性函数，但由于$r5\neq r6$，式(12)简化后一定含有两个线性函数相乘的形式，故此运算必定为非线性运算。若将式(9)带入至式(12)，该非线性运算的次数又会增加。得到的c_i^{t+1}后可控制下一个时刻即$t+2$时刻ECA1的迭代，随着迭代的进行，计算某个元胞状态的运算式次数会不断增加。此外，在迭代过程中还进行了移位和伪随机边界的运算，迭代规则表也在不断更新，再加上步骤3的整合输出，整个算法的非线性能够得到较好的保证。

要计算所有输出位的非线性度需要的计算复杂性达到了2^{256}，计算上是不可接受的。因此为了验证所提序列密码算法的非线性强度，本文在每一轮迭代中，选取种子密钥的中间8位作为输入，最终输出的中间8位作为输出字节，并将其视为一个8元布尔函数的真值表(通过遍历种子密钥对应8位的所有256种可能，其他密钥位和初始向量不变)。随后，计算不同迭代次数下该布尔函数的非线性度，同时取8位非线性度的均值，计算结果如图5所示。

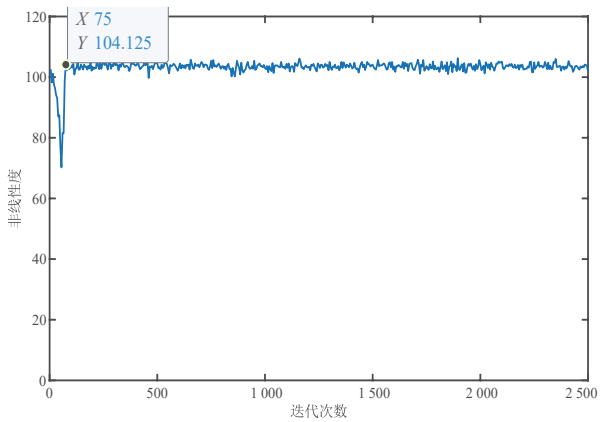

图5 中间八位输出的非线性度随迭代次数的变化

Figure 5 Nonlinearity of the middle 8-bit output versus the number of iterations

由图5可知，随着迭代次数的增加，输出函数的非线性度迅速趋于稳定，自第75次迭代起，所有测试实例的非线性度均值基本稳定在100以上，表明该算法能够有效混淆线性关系，具备抵抗线性分析的能力。在实际使用中，可在正式输出密钥流之前使算法预迭代多次，抛弃输出结果，该次数甚至可作为密钥的一部分，这样可获得较高且稳定的非线性度，从而提高算法安全性。

2.3.3 前向安全和后向安全性

在序列密码设计中，前向安全性与后向安全性是衡量密钥流抗攻击能力的两个重要指标。前向安全性要求攻击者即使掌握了生成序列中的某一时刻的内部状态，也无法有效推导出后续的序列输出；后向安全性则要求攻击者即便完全获取了某一时刻的状态，也不能恢复出先前已生成的密钥流，从而保证历史数据的机密性。

(1)前向安全性

本文提出的“旋转陀螺”算法主体由三个ECA嵌合而成，三者采用规则置乱与控制依赖机制。具体而言，算法在每一轮迭代中，通过如下方式保障前向安全性。

弹性混沌规则的选择：算法选取的迭代规则均对应于至少1阶相关免疫的均衡布尔函数，具备基本的相关免疫性和平衡性。均衡性保证每一比特输出是均匀分布的(0、1概率相等)，避免攻击者通过统计偏差预测未来序列。相关免疫性保证输出与任意少量输入比特之间没有可利用的相关性。这样即使攻击者知道当前部分状态，也难以预测下一轮的输出。规则的这些性质直接提高了未来序列的不可预测性，从而强化前向安全。

规则置乱：初始迭代规则表在种子密钥驱动下进行置乱，确保每一次初始化均与密钥高度相关。攻击者若仅获得某t时刻三个ECA的中间状态，并不能推

导出下一轮的规则选择,因此后续比特生成过程存在不确定性。

多层交互:第一层ECA的输出控制第二层,第二层控制第三层,第三层再反馈控制第一层,形成循环依赖关系,再加上不断变化的迭代规则表。这一结构在每次迭代中都引入新的非线性组合,使得状态到输出的映射为高度非线性函数,攻击者很难通过线性分析等方式得到等价线性函数,所以无法准确预测后续序列。

整合输出与动态规则索引:输出比特由三层ECA状态拼接后,并结合动态滚动的迭代规则表确定。即使攻击者完全掌握当前输出比特,由于规则索引的循环移位与重新排列,未来比特的分布仍难以预测。同时由于最后整合输出采用的是八个弹性混沌规则之一,具备相关免疫性,加之对应的布尔函数必定涉及异或运算,而异或运算是不可逆的,故攻击者得到了当前时刻的输出比特也无法通过相关攻击等手段获得三个ECA当前的内部状态。

因此,若攻击者试图通过当前状态推导后续序列,其必须同时解决上述四个算法特有的性质。这些性质保证了系统的前向安全性。

(2)后向安全性

算法的后向安全性同样可以得到保证,原因主要如下。

迭代的不可逆性:迭代过程中,ECA之间的嵌合结构会形成高度非线性的映射,该映射属于不可逆操作,迭代一次后,给定当前三个ECA的状态,攻击者无法唯一确定上一时刻的状态。迭代过程中边界填充的两个比特不是直接从前一时刻的序列推导出来的,而是通过当前时刻使用的规则$r0$、$r1$和边界比特共同确定。因此即使给定当前ECA的状态,也无法唯一确定上一时刻边界的真实比特。由算法步骤2可知,当前ECA中的元胞采用的迭代规则是由另一个ECA的状态所决定,攻击者如果只掌握了某一ECA的状态,而并不知道其他ECA的状态,就无法回溯到底是通过规则$r0$还是$r1$产生的该比特。

弹性规则的相关免疫性:由于弹性混沌规则所对应的布尔函数至少具备1阶相关免疫性,不同的输入组合可能映射到同一个输出,攻击者无法通过低阶相关性重建历史状态。

规则表的动态性:迭代规则表在每轮迭代中会发生循环或重新排序,不同的历史路径可能导致相同的当前状态,从而加大了逆向推导的难度。

依赖反馈机制:三层ECA相互控制,当前状态不仅取决于前一轮的状态,还依赖于规则组合与种子密钥对算法的初始化。因此,攻击者即便完全获取某一时刻的内部状态,也无法准确反推出前一时刻的输出比特。

3　随机性测试

3.1　NIST SP800-22测试

NIST SP800-22随机性测试套件是由美国国家标准与技术研究院(National Institute of Standards and Technology, NIST)提出的一套统计检验方法,用于评估二进制序列的随机性特征[19]。该套件包含15项基础测试,这些测试能够从不同角度检验序列是否符合理想随机比特流的统计分布特性。测试依赖通过率综合评估序列的随机性,若整体通过率处于统计容忍区间内,则该序列被认为在该项测试中表现合格。

对“旋转陀螺”算法生成的密钥流进行测试,测试显著性水平设定为$\alpha=0.01$。实验中共测试256组子序列,每组子序列含10^6个比特,对每组子序列执行15项测试。根据NIST的测试标准,通过率大于等于248/256则通过该项测试。需要注意的是“随机游走测试”与“随机游走变体测试”的组数依赖于序列对应随机游走的零穿越次数,因此不同于其他固定划分子序列的测试,其有效样本数量随输入比特流特性而变化,故基准通过率也会有所变化。为避免实验结果的偶然性,实验选取随机生成的不同种子密钥,共进行了五次实验,测试结果如表5所示。

表5中标注有“*”的测试表示该测试下包含多个子测试,表中这些测试项目的通过率取其子测试的最小值。由表可知,对于随机生成的不同密钥种子,“旋转陀螺”算法生成的密钥流均能够通过SP800-22的15项测试,表明该算法生成密钥流的随机性在统计意义上达到了可接受水平。

3.2　Test U01测试

Test U01是由L'Ecuyer等人[20]开发的伪随机数序列测试库,为序列密码算法随机性分析提供科学依据。该测试库提供多层次的测试套件,可对序列的均匀性、独立性及相关性进行严格检验,其输出包括p值与测试结果。

为快速获取不同种子密钥下的测试结果,本研究采用SmallCrush和Rabbit两个测试套件。SmallCrush是轻量级快速测试套件,包含10项基础测试(共15项子测试),用于初步筛查随机数生成器的明显缺陷。Rabbit是Test U01中用于测试二进制随机比特流的严苛测试套件,包含26项(共40项子测试)统计测试,通过分析比特序列的统计特性来检测缺陷。在Test U01中,若检验的p值落在区间$[0.001, 0.999]$则认为该检验通过;若p值接近0或1(如小于10^{-10}),则认为该检验不通过或序列存在显著偏差。本研究对“旋转陀螺”算法在五个不同随机种子密钥下生成的密钥流进行了测试,测试结果如表6所示。

表5 NIST SP800-22 随机性测试结果

Table 5 NIST SP800-22 randomness test results

测试项目	实验1	实验2	实验3	实验4	实验5
频数测试	253/256	254/256	250/256	249/256	253/256
块频数测试	253/256	254/256	255/256	253/256	253/256
累积和测试*	251/256	253/256	251/256	249/256	254/256
游程测试	256/256	256/256	250/256	250/256	253/256
最长游程测试	253/256	254/256	253/256	254/256	252/256
秩测试	254/256	256/256	251/256	253/256	254/256
快速傅里叶变换测试	253/256	254/256	252/256	254/256	252/256
非重叠模板匹配测试*	248/256	248/256	248/256	249/256	249/256
重叠模板匹配测试	253/256	255/256	254/256	249/256	251/256
通用统计测试	254/256	256/256	253/256	252/256	253/256
近似熵测试	253/256	255/256	255/256	256/256	249/256
随机游走测试*	137/140(≥135/140)	155/161(≥155/161)	140/143(≥138/143)	148/151(≥145/151)	145/151(≥145/151)
随机游走变体测试*	136/140(≥135/140)	158/161(≥155/161)	141/143(≥138/143)	147/151(≥145/151)	148/151(≥145/151)
串行测试*	249/256	251/256	250/256	254/256	254/256
线性复杂度测试	254/256	254/256	250/256	252/256	254/256

表6 Test U01 随机性测试结果

Table 6 Test U01 randomness test results

测试套件	密钥流1	密钥流2	密钥流3	密钥流4	密钥流5	测试结果
SmallCrush(p值:最小~最大)	0.10~0.995 2	0.01~0.87	0.02~0.97	0.10~0.990 2	0.004 7~0.996 5	通过
Rabbit(p值:最小~最大)	0.04~0.88	0.03~0.995 0	0.007 8~0.998 8	0.007 1~0.995 0	0.001 4~0.996 5	通过

表6中列出的是所有测试项目下得到的最小和最大p值。其中最小值为0.001 4,最大值为0.998 8,所有测试所得p值均在通过区间[0.001,0.999]内,测试结果表明“旋转陀螺”算法所生成的密钥流在统计意义上未表现出显著的非随机性偏差,这进一步说明该算法在随机性方面具有较好的可靠性。

4 算法分析

4.1 安全性分析

在前文的算法特性分析中,已经验证了“旋转陀螺”算法的高度非线性和相关免疫性,因此基本的线性分析攻击比如B-M算法和相关分析攻击对于该算法是无效的,而在随机性测试中又验证了算法输出序列具备良好的随机性,一般的统计分析无法对该算法构成威胁。本算法的内部状态大小达到了$2^{768}\times 8!$,远超种子密钥长度的两倍,因此基于时间存储数据折中的分析方法对该算法无效[21]。立方攻击则主要针对具备非线性反馈移位寄存器结构的算法[7],对本算法并不适用。本节主要使用区分攻击和猜测—确定攻击的方法来进一步验证该算法的安全性。

分析中主要考虑被动攻击者模型,即攻击者仅通过观测算法输出的密钥流或其统计特性实施攻击,不具备对算法内部结构的主动篡改能力。具体假设如下。

攻击者完全了解算法的公开结构,包括元胞自动机的更新规则形式、三ECA循环嵌合结构、规则置乱与输出整合机制等,符合Kerckhoffs原则;但攻击者未知种子密钥**Key**,且无法控制其取值。

攻击者可以获取由算法生成的任意长度的密钥流序列,并可在不同**Key**条件下重复观测输出;但攻击者无法直接访问或精确获知任意时刻三个ECA的完整内部状态,也无法获得每轮迭代中具体采用的局部迭代规则序列。

攻击者的主要目标包括。

(1)通过统计分析或判别方法,将算法输出序列与理想随机序列区分开来,即区分攻击;

(2)利用输出比特与内部状态或规则之间的潜在关系,逐步猜测部分内部状态或迭代规则,并最终恢复完整状态或密钥,即猜测—确定攻击。

在上述攻击模型下,以下小节将分别从区分攻击和猜测—确定攻击两个方面,对“旋转陀螺”算法的安全性进行分析,论证其在合理攻击假设下的抗攻击能力。

4.1.1 区分攻击

在上述攻击模型下,攻击者仅能被动获取算法输出的密钥流序列,而无法直接访问内部状态或迭代规

则。在该威胁假设下,区分攻击[21]主要通过判断输出序列是否与真正随机序列存在统计差异来寻找算法漏洞,进而破解算法。

本文主要从统计的角度评估生成序列的周期性、熵和相关性等是否与真随机数有显著差距。周期性方面,理论上该算法的状态空间极大,算上迭代规则表共8!种排列方式,共有$2^{768}\times 8!$种不同的状态,因此理论周期是极大的,很难与真随机数区分。本文通过输出足够长的序列求自相关系数的方式来判断算法是否存在短周期,令算法迭代10^6次,取不同的种子密钥和不同元胞位置的输出密钥流进行10次实验,得到的自相关系数结果如图6所示。

图6种各个子图的横坐标表示延迟,纵坐标表示不同延迟下序列的自相关系数,由图可知,在不同种子密钥下,取不同元胞位置生成的密钥流求得的自相关系数均类似,实验结果显示,其自相关系数仅在延迟为零时是1,其他延迟下均接近于0,这一点与真随机数相同,亦可知序列至少不存在10^6以下的短周期。

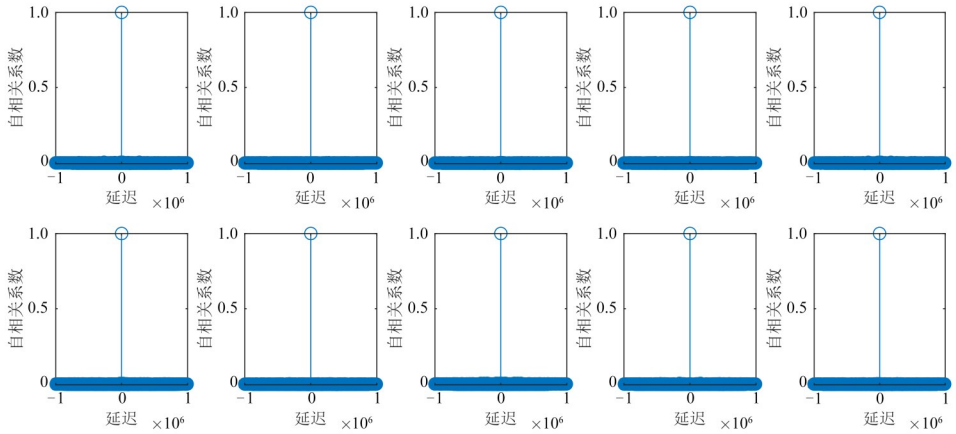

图6　不同种子密钥生成序列的自相关系数

Figure 6　Autocorrelation coefficients for sequences with different seed keys

对于真随机序列,其输出比特应独立且均匀,因此熵值应接近于1。利用“旋转陀螺”算法迭代10^6次,同样计算不同种子密钥下,不同元胞位置生成密钥流的熵。进行10次实验得到的结果如表7所示。

表7　不同密钥流信息熵的计算结果

Table 7　Information entropy of different key streams

实验序号	第1次	第2次	第3次	第4次	第5次	第6次	第7次	第8次	第9次	第10次
熵	1	0.999 9	1	1	1	1	1	0.999 9	0.999 9	1

由表7可知,算法得到的不同密钥流,信息熵均接近或等于1,实验结果说明输出的密钥流比特分布十分均匀,与真随机数相似。这是由于算法采用的是弹性混沌规则,其指代的布尔函数都为均衡函数。

对于同一真随机数源,其产生的不同序列之间应该是相互无关的。利用本文所提算法生成足够长度的密钥流,进行以下三组实验:计算左右相邻元胞位置生成密钥流之间的相关系数,每次实验取不同位置上的相邻元胞;随机改变种子密钥的某一位,计算同一元胞位置新生成的密钥流和原始密钥流之间的相关系数,每次实验改变种子密钥的不同比特位;对种子密钥整体取反,计算同一元胞位置新生成的密钥流和原始密钥流之间的相关系数,每次实验取不同的种子密钥,每组实验进行10次,得到的实验结果如表8所示。

表8　相关性分析

Table 8　Correlation analysis

实验序号		第1次	第2次	第3次	第4次	第5次	第6次	第7次	第8次	第9次	第10次
相邻元胞位置	左邻居	0.000 5	0.000 8	0.004 1	0.000 8	0.000 9	0.002 3	0.004 0	0.003 3	0.002 2	0.008 0
	右邻居	0.001 7	0.003 2	0.005 3	0.007 0	0.006 5	0.000 1	0.003 2	0.001 1	0.002 7	0.008 1
改变种子密钥某一位		0.003 4	0.002 6	0.001 1	0.002 6	0.001 8	0.000 8	0.000 6	0.000 3	0.001 5	0.000 6
种子密钥取逆		0.000 6	0.004 4	0.002 6	0.004 3	0.002 8	0.002 9	0.005 5	0.001 4	0.001 8	0.002 3

表8中相关系数均保留小数点后四位数,由表可知三组实验得到的相关系数均小于0.01,最大仅为0.008 1,工程上对于二进制序列,当相关性低于0.05时则认为两组数据几乎无关,因此可以得出如下结论:

相同种子密钥下，相邻元胞位置输出的序列之间相关性极弱，敌手很难通过某一元胞位置输出的密钥流推理出其他密钥流；种子密钥取逆或者改变种子密钥的某一比特位，生成的密钥流和原始密钥流几乎无关，这意味着算法具备良好雪崩特性与高度密钥敏感性，可以有效抵抗相关攻击与差分分析。

综上所述，“旋转陀螺”算法在周期性、熵和相关性等方面均与真随机数类似，敌手很难通过区分攻击的方法破解算法。相关实验表明算法具备极大周期，随机性良好，同时能够抵抗相关分析和差分分析。

4.1.2 猜测—确定攻击

在前述攻击模型中，假设攻击者可在选择明文条件下获取任意长度的密钥流，但无法获知算法任意时刻的完整内部状态或迭代规则。在该条件下，猜测—确定攻击（Guess-and-Determine Attack，GDA）[21-22]是对序列密码常用的攻击策略：攻击者能够通过大量明密文对获取足够长的密钥流，之后利用算法本身存在的约束条件确定其余内部状态或验证猜测的正确性。若最终能够一致地恢复出内部状态或密钥，即完成攻击。

对于“旋转陀螺”算法，攻击者在选择明文模型下可获取任意长度的密钥流。如果采用暴力破解的方式，破解初始种子密钥，则有2^{256}种可能，若要破解算法内部状态，加上迭代规则表则有$2^{768}\times 8!$种可能，故暴力破解的时间复杂度对敌手来说均是不可接受的，本文提供两种可能的猜测策略以期降低破解的时间复杂度：

策略1 静态层面的攻击，从较好猜测的迭代规则表和当前时刻输出迭代规则出发，迭代规则表共8!种排列方式，计算时间复杂度在2^{15}~2^{16}之间，然后猜测输出迭代规则即从八个规则中选一个，此时时间复杂度达到2^{18}~2^{19}，对于敌手来说是可接受的，接着取当前时刻输出密钥流的256 bit对三个ECA的元胞状态进行确定。已知密钥流某一比特的值为0(1)，则与之对应的当前元胞位置上三个ECA的元胞状态缩减至4种可能，这是由于规则指代的布尔函数为均衡函数，对于输出为0或1有四种可能的输入，图7以猜测规则$r=60$为例展示了这一过程。若要猜测所有元胞位置的可能性共有4^{256}种可能，时间复杂度达到了2^{512}种，对于敌手来说依旧是不可接受的。

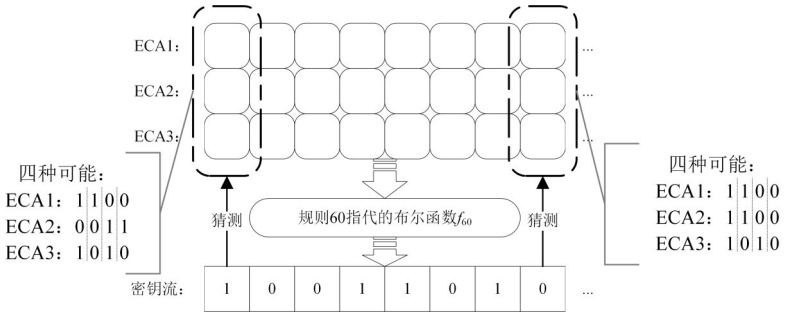

图7 猜测当前时刻整合输出时使用规则为60的GDA过程

Figure 7 GDA process with rule-60 for real-time output integration

图7中各个圆角方格代表ECA中的元胞，方格中的值为当前时刻对应元胞位置上的输出，进一步观察该图GDA过程，ECA1、ECA2、ECA3三者同一位置的元胞之间是存在一定约束关系的，图中若当前位置输出的密钥流为“1”，则ECA1和ECA2在该位置的元胞状态必然不同，若输出为“0”，则两者的状态必然相同，然而由于混沌弹性规则具备相关免疫性，ECA3的状态和输出无关，依旧无法确定。不考虑前面猜测迭代规则的过程，假设敌手能够猜测出其中ECA1的状态，这就需要2^{256}的时间复杂度，虽然能进一步确定ECA2的状态，但ECA3的状态依旧需要穷举，总的时间复杂度依旧为2^{512}。由于使用的迭代规则均具备相关免疫性，即使是其他输出迭代规则，依旧无法在可接受时间内确定三个ECA的内部状态。综上所述，该策略下总的破解时间复杂度为$2^{512}\times 8!$，大于直接暴力破解密钥，因此攻击无效。

策略2 动态层面的攻击，在策略1的基础上，参考MS攻击方法[9-10]，综合考虑不同时刻的密钥流和元胞之间的约束，尝试得到所有元胞在某一时刻的状态。如果仅猜测一个时刻某一ECA的状态，即使能够由输出迭代规则确定另一个ECA的状态，但由于第三个ECA未知，又因为三个ECA在迭代时相互依赖，因此无法进行下次迭代的推导和验证。故假设已猜测出迭代规则表以及t和t+1时刻ECA1的8个元胞状态以及输出迭代规则，时间复杂度为$2^{16}\times 8!\times 8^2$，约为$2^{37}$，对敌手来说是可接受的。在迭代中GDA的过程如图8所示。

图8中带有阴影和数字的圆方格代表状态被猜

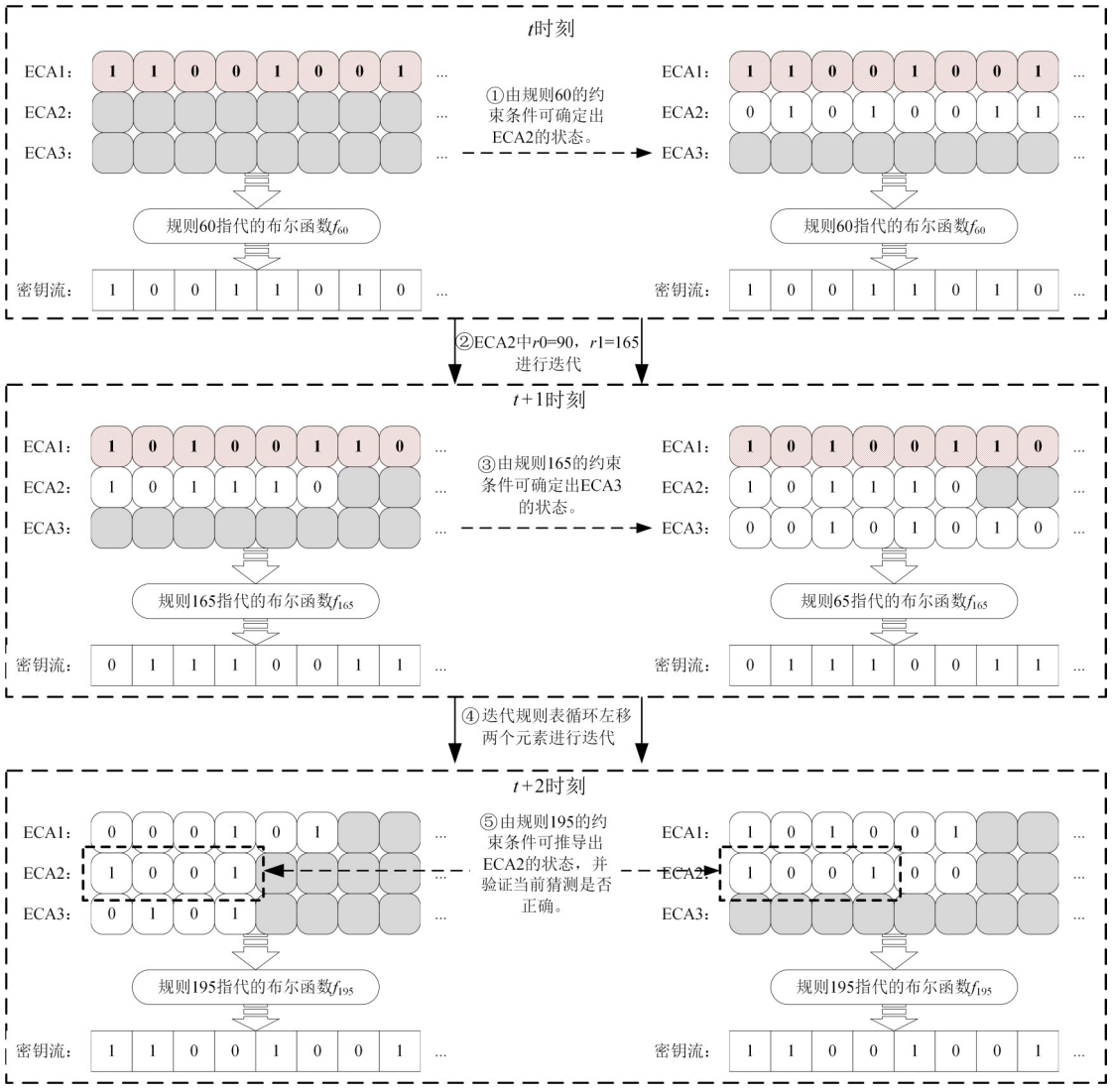

图8　在迭代过程中使用GDA

Figure 8　Employing GDA during the iterative process

测出的元胞,白色圆方格则是由约束条件或迭代规则唯一确定状态的元胞,灰色圆方格代表当前无法确定状态的元胞,GDA过程中假设使用的所有迭代规则已经通过猜测获得。图8描述的GDA过程如下。

①在时刻t敌手利用当前输出迭代规则60的约束,由密钥流和ECA1的8个元胞状态可唯一确定ECA2相同位置上8个元胞的状态。

②由$t+1$时刻猜测出的ECA1的8个元胞状态可实现对ECA2的控制迭代,得到$t+1$时刻ECA2的部分元胞状态,这里只能确定前6个元胞的状态,原因是在迭代时包含左移一位的操作,故最右侧两个元胞的状态无法确定。

③在$t+1$时刻由已知的密钥流和ECA1、ECA2的元胞状态进行推导,会有三种可能:其一,如果输出迭代规则指代的布尔函数与ECA1的元胞状态无关,如规则102、153,那么到此仅能推出ECA3的前6个元胞状态;其二,如果输出迭代规则仅与ECA3无关,如规则60、195,则可根据输出规则约束下推导的和迭代推导的ECA2的元胞状态是否相同,来判断本次猜测是否正确,对于ECA3的元胞状态依旧未知;其三,输出迭代规则如果仅与ECA2无关,那么可以由该约束直接得到ECA3的三个状态,此时对于敌手来说是最为有利的,图中展示的也是这种情况,即输出迭代规则为165,ECA3的状态可以直接由密钥流和ECA1的状态共同确定。

④根据算法描述,首先对迭代规则表进行置换或循环移位,接着根据$t+1$时刻已经获得的三个ECA的状态进行迭代,由于移位的存在,仅能得到ECA1

的6个元胞状态和ECA2、ECA3的4个元胞状态。

⑤根据$t+2$时刻的输出迭代规则和密钥流，仅能够验证猜测的正确与否。已经通过迭代获取到了ECA2在$t+1$时刻的部分元胞状态，同时根据当前输出迭代规则为195，可由$t+2$时刻ECA1状态和密钥流确定ECA2的另一个元胞状态结果，两个结果如果相等则证明猜测可能正确，否则猜测一定不正确。如果是其他输出迭代规则情况类似，仅仅是用于验证猜测正确与否的ECA不同。

根据上述GDA过程，如果猜测正确，迭代可继续下去，同时由于移位的存在，不确定的元胞数必然会增加，当某一ECA的可确定状态的元胞数小于等于3时，迭代便无法继续，需要重新再次猜测元胞状态。假设敌手算力很强，2^{128}的时间复杂度依旧可以接受，设敌手能够猜测的最大元胞状态数为L，猜测迭代规则表需要的时间复杂度约为2^{15}。每次迭代都要猜测输出迭代规则，但并不是每次猜测都要从8个迭代规则中进行选择，这是因为输出迭代规则的选取与三个位置相距较远的元胞状态相关，如果猜测的元胞状态能够涵盖这三个元胞位置则可以减少猜测次数。令敌手通过GDA能够进行的迭代次数为n，且GDA过程中遇到的都是有利的状况，如图8中的步骤③。综上，可得到如下方程：

$$\begin{cases} n=L/2-1 \\ 4^n\times 2^{15}\times 2^{2L}=2^{128}, & 2<L<86 \\ 2^n\times 2^{15}\times 2^{2L}=2^{128}, & 86<L<171 \\ 2^{15}\times 2^{2L}=2^{128}, & 171<L<256 \end{cases} \tag{13}$$

求解方程(13)可知仅有$2<L<86$时有解，解得$L\approx 38.33$，$n\approx 18.17$，向上取整后可知敌手在时间复杂度为2^{128}可接受的情况下最多能够正确猜测出39个元胞的状态，并且在最好的情况下最多可迭代19次。从密钥流的末尾开始，减去用于验证猜测是否正确的序列，即两次迭代，最多仅能准确预测后续的289个非连续比特。进一步假设敌手算力为2^N，且N足够大，L一定可涵盖决定输出迭代规则的三个元胞，此时建立N、L和后续能够准确猜测的比特数M之间的关系如下：

$$\begin{cases} L=\dfrac{N-15}{2} \\ M=\displaystyle\sum_{i=3}^{L/2-1}(L-2\times i)=\frac{L^2}{4}-\frac{5L}{2}+6 \end{cases} \tag{14}$$

$$M=\frac{N^2-50N+621}{16} \tag{15}$$

由式(15)可知预测一个比特位消耗的时间复杂度为$O(2^N/N^2)$，依旧为指数级，而且此结果是对敌手最为有利的情况下得到的，故策略2的攻击无效。

策略3 在同一个ECA中，由猜测的元胞状态，扩散推导以确定其他未知的元胞状态。该策略在相关文献中已经做过分析，要得到相邻元胞的未知状态，需要知道该ECA上一时刻的元胞状态以及控制该ECA的另一个ECA的元胞状态，这从原理上是不可行的，等价于解决3-SAT问题[8,23]。同时，前文的相关性分析也从统计的角度验证了同一个ECA相邻元胞进行推导的困难性。

4.2 对比分析

将“旋转陀螺”算法与一些较新的基于CA的序列密码算法进行对比，得到的对比结果如表9所示。

表9中，效率是指在理想条件下(使用流水线技术)，各算法在每个时钟周期能够输出的密钥流长度。由表可知，相较于其他基于CA的算法，“旋转陀螺”算法具备较高的效率。安全性方面，由于采用了弹性混沌规则，对应布尔函数具备相关免疫性，同时ECA之间的控制迭代又显著增强了算法非线性度，使得“旋转陀螺”算法的安全性相较于其他算法也更为可靠。

4.3 性能分析

为进一步评估所提出的“旋转陀螺”算法在不同计算平台上的适用性，本节对比了该算法与国际标准算法ZUC、Grain和Trivium在吞吐率、存储占用及实现复杂度等方面的差异。

在通用CPU环境下，由于ZUC采用字级运算和线性反馈移位寄存器结构，其软件实现能够充分利用编译器优化和现代处理器的字级并行特性，单核吞吐率可达到4.22 Gbps[25]。而“旋转陀螺”算法在CPU平台上需执行多层迭代和动态规则选择，导致单线程吞吐率约为0.1~0.5 Gbps，整体性能低于ZUC。

在FPGA平台上，本文将文献[26]的实验结果量化至200 MHz的相同时钟频率环境下进行对比，对比结果如表10所示。其中，ZUC的S盒和线性变换结构需要额外的查表与逻辑资源，尽管可实现7.1 Gbps的吞吐率[27-28]，但往往伴随较高的LUT与逻辑单元的消耗。Trivium能够达到6.4 Gbps的吞吐率，但是通过32倍展开实现的，增加了硬件消耗，同样的方法，Grain-128也可以通过32倍展开实现并行运行，从而达到6.41 Gbps[26]。“旋转陀螺”算法的局部更新特性使其可以在硬件中构建完全位并行的流水线结构，理论上可实现单时钟周期更新256位状态，相同时钟频率下吞吐率可超过25 Gbps，且单位资源的吞吐效率优于其他算法。这表明所提出算法在可编程逻辑器件上具有较强的实现优势。

表9 “旋转陀螺”算法与基于ECA的其他序列密码算法对比

Table 9 Comparison of the “Spinning Top” with other ECA-based stream ciphers

方案	算法结构	算法特点	效率/每时钟周期	说明
文献[9]（2020年）	属于Grain Family，包括线性模块、非线性模块以及输出函数等结构。	改进Rule30为Rule30+，修改依赖关系解决线性相关性，可抵抗MS攻击。	输出1 bit	依赖Grain形结构，存在被立方体攻破的可能。ECA虽做出了改进，但所使用的ECA规则固定，一旦内部状态泄露，则后续密钥能够完全得到。优化后的Rule30+，有效规避了MS攻击，然而对应的布尔函数并不存在相关免疫性，就单个ECA而言依旧存在被猜测确定攻击攻破的风险。
文献[8]（2024年）	主体由两个ECA嵌合构成，隐藏部分输出。	通过一个简单ECA控制一个双规则混杂ECA进行迭代，优化了单一ECA输出序列的统计特性。	输出128 bit	安全性依赖于3-SAT问题的等价性。通过隐藏一半的输出有效规避了猜测确定攻击，但这降低了算法的输出效率。每次加密使用的规则为随机选取，加密开始后便固定下来，并不能保证相关免疫性，若敌手能够查看内存，则无法很好地抵抗猜测确定攻击。
文献[24]（2024年）	结合混沌映射和二阶CA实现图像加密。	密钥流主要由混沌映射生成，CA则主要用于明文图像的演化和置乱。	输出<8 bit	该算法生成的密钥流主要依赖混沌映射，涉及实数运算，效率上相较于其他位运算算法较弱。CA则使用了二阶元胞自动机，元胞邻居涉及四个，因此运算消耗的资源较多。此外该算法的安全性主要依赖统计特性上的分析，与CA对应的布尔函数关系不大，安全性方面无法很好地衡量和判断。
“旋转陀螺”算法	主体由三个ECA循环嵌合构成。	三个ECA均为双规则控制，互相影响，互相控制，最终整合输出。	输出256 bit	算法使用弹性混沌规则，其相关免疫性使算法能够有效抵抗猜测确定攻击，同时迭代规则表的变化以及循环嵌合结构，使得算法在有限资源下拥有高度的非线性，且具备较为可靠的前向安全和后向安全性。效率方面，由于运算的并行性，且所有输出均被使用，不存在隐藏部分，显著提高了算法效率，在流水线技术下效率能够优于其他算法。

表10 “旋转陀螺”算法与其他标准算法性能对比（相同时钟频率）

Table 10 Comparison of the “Spinning Top” algorithm with standard algorithms (at identical clock frequencies)

平台	算法	典型吞吐率/Gbps（数值越大越好）	资源占用（数值越小越好）	说明
CPU: Intel i7-2700H（单核C实现）	ZUC[25]	4.22	~70字节	ZUC更适合字级并行，软件效率高。
	“旋转陀螺”算法	0.1~0.5	~96字节	需要大量位运算/索引，分支/查表影响流水线。
FPGA: Zynq-7000/200 MHz	ZUC[27-28]	7.1	575 slices（每个slices含4寄存器+4个LUT）	ZUC有S盒运算，可用BRAM/ROM实现并行S盒运算。
	Trivium(×32)[26]	6.4	621个寄存器+503个LUT	通过32倍循环展开，可在单时钟周期内并行计算32轮内部状态，从而构建高吞吐率的流水线。
	Grain-128(×32)[26]	6.41	384个寄存器+609个LUT	局部依赖特性使得在硬件中可以高效地进行多轮（如32轮）组合逻辑复制与展开，构建并行的流水线，在不显著增加寄存器资源的情况下大幅提升吞吐率。
	“旋转陀螺”算法	25~40	768个寄存器+8个LUT	ECA局部规则、位级独立，可做极高并行度流水线，在同等FPGA资源下通常能超越其他算法。

5 结束语

本研究利用三个ECA的有机嵌合构成了新型的序列密码算法——“旋转陀螺”算法。该算法选取具有均衡性和相关免疫性的混沌迭代规则，并命名为“弹性混沌迭代规则”，设计出了新型的ECA非线性嵌合方式，三个ECA互为表里，相互控制，并最终整合输出密钥序列。相关分析和实验表明该算法具备以下诸多优点：良好的伪随机性，能够通过NIST SP800-22以及Test U01的相关随机性测试；高度的非线性，随着迭代的进行，非线性度会不断加深；可靠的安全性，除具备基本的前向和后向安全性外，能够抵抗区分攻击和猜测确定攻击。在安全性和效率方面相比于同类型的其他算法均具有明显优势，相比于

国际标准算法ZUC、Trivium和Grain-128，相同时钟频率下该算法在FPGA中的吞吐率能够达到25 Gbps以上，是其他算法的3倍以上，且占用的资源数还略低。

应用方面，该算法实现简单，易于推广。其一，针对工业互联网、数字孪生及6G算力网络等场景对“高安全—高吞吐”需求，算法可满足实时数据传输加密、终端设备身份认证等核心安全需求，为关键信息基础设施的安全防护提供技术保障；其二，解决了传统序列密码安全性提升依赖复杂度增加，进而导致硬件资源消耗过高的矛盾，该算法通过ECA的局部迭代特性和非线性结合实现了“低资源占用—高安全性能”的平衡，适配边缘终端、工业控制器等资源受限设备；其三，推动了元胞自动机这一经典动力学模型在密码学领域的实用化进程，为后续基于复杂系统理论的密码算法研究提供了新思路，有望激发“元胞自动机—密码学—硬件”实现交叉领域的进一步创新。

未来研究可进一步优化算法的软件实现效率（如通过SIMD指令加速位运算），并进行硬件实现进一步提高吞吐率，以期为更广泛的安全场景提供更具适应性的加密方案。

参考文献

[1] Wang H Y, Hsu C, Harn L. A lightweight and robust stream cipher based on PI for intelligent transportation systems[J]. Wireless Personal Communications, 2023, 130(3): 1661-1675.

[2] Ding L, Liao Z Y, Li Z T, et al. A practical key recovery attack on the lightweight WG-5 stream cipher[J]. Heliyon, 2024, 10(2): e24197.

[3] Rashidi B. High-performance hardware structure of ChaCha20 stream cipher based on sparse parallel prefix adder[J]. International Journal of Circuit Theory and Applications, 2025, 53(5): 2947-2957.

[4] 刘晨, 田甜. 关于Trivium-型序列密码代数次数估计的研究[J]. 密码学报, 2021, 8(1): 110-123.
Liu Chen, Tian Tian. On degree evaluation of trivium-like stream ciphers[J]. Journal of Cryptologic Reseatch, 2021, 8(1): 110-123. (in Chinese)

[5] 马成栋, 蒋梓龙, 魏鹏. 基于SMT的ACORN v3算法的差分分析[J]. 智能安全, 2024, 3(3): 1-11.
Ma Chengdong, Jiang Zilong, Wei Peng. Differential cryptanalisis of ACORN v3 based on SMT[J]. Artificial Intelligence Security, 2024, 3(3): 1-11. (in Chinese)

[6] Li Y Q, Cui T. Linear forgery attacks on the authenticated encryption cipher ACORN-like[J]. Chinese Journal of Electronics, 2025, 34(1): 257-265.

[7] 刘晨, 田甜, 戚文峰. 针对立方攻击中大规模超多项式恢复技术的改进(英文)[J]. 密码学报(中英文), 2024,11(5): 1179-1198.
Liu Chen, Tian Tian, Qi Wenfeng. Improvement on large-scale super-polynomial recovery technology in cube attacks[J]. Journal of Cryptologic Research, 2024, 11(5): 1179-1198. (in Chinese)

[8] Du P, Dong Y H, Cui Q, et al. A novel hybrid elementary cellular automata and its application in a stream cipher[J]. Applied Sciences, 2024, 14(21): 9719.

[9] 郭晓威, 郭亚军. 一种基于Rule30+细胞自动机的流密码设计方法[J]. 密码学报, 2020, 7(4): 439-452.
Guo Xiaowei, Guo Yajun. An efficient stream cipher design based on Rule30+ cellular automaton[J]. Journal of Cryptologic Research, 2020, 7(4): 439-452. (in Chinese)

[10] Meier W, Staffelbach O. Analysis of pseudo random sequences generated by cellular automata[C]//Advances in Cryptology-EUROCRYPT'91. Berlin: Springer, 1991: 186-199.

[11] Wolfram S. Cellular automata as models of complexity[J]. Nature, 1984, 311(5985): 419-424.

[12] Neumann J V B A. Theory of self-reproducing automata[J]. University of Illinois Press, 1966.

[13] Stănică G C, Anghelescu P. Reversible cellular automata based cryptosystem[J]. Electronics, 2024, 13(13): 2515.

[14] 周宇,胡予濮,董新锋. 布尔函数的设计与分析: Design and analysis of boolean functions[M]. 北京: 国防工业出版社, 2015.
Zhou Yu, Hu Yupu, Dong Xinfeng. Design and Analysis of Boolean Functions[M]. Beijing: National Defense Industry Press, 2015. (in Chinese)

[15] 胡予濮, 杨波, 张玉清. 均衡弹性函数的结构与弹性阶[J]. 电子学报, 2002, 30(7): 1035-1037.
Hu Yupu, Yang Bo, Zhang Yuqing. Structures and resilient orders of balanced resilient functions[J]. Acta Electronica Sinica, 2002, 30(7): 1035-1037. (in Chinese)

[16] Li W, Packard N. The structure of the elementary cellular automata rule space[J]. Complex Systems, 2000, 4(3): 281-297.

[17] Wolfram S. Universality and complexity in cellular automata[J]. Physica D: Nonlinear Phenomena, 1984, 10(1/2): 1-35.

[18] Burrieza J E, Del Rey A M, Pérez Iglesias J L, et al. Cryptographic properties of Boolean functions defining elementary cellular automata[J]. International Journal of Computer Mathematics, 2011, 88(2): 239-248.

[19] Rukhin A, Soto J, Nechvatal J, et al. A statistical test suite for random and pseudorandom number generators for cryptographic applications[J]. Nist Special Publication, 2001.

[20] L'Ecuyer P, Simard R. TestU01: A C library for empirical testing of random number generators[J]. ACM Transactions on Mathematical Software, 2007, 33(4): 1-40.

[21] 周照存, 冯登国. 流密码分析方法研究综述[J]. 通信学报, 2022, 43(11): 183-198.
Zhou Zhaocun, Feng Dengguo. Survey on approaches of stream cipher cryptanalysis[J]. Journal on Communications, 2022, 43(11): 183-198. (in Chinese)

[22] Knudsen L R, Meier W, Preneel B, et al. Analysis methods for (alleged) RC4[M]//Advances in Cryptology-ASIACRYPT'98. Berlin, HeidelbergSpringer, 1998: 327-341.

[23] 董有恒. 基于时空混沌系统和元胞自动机的序列密码研究[D]. 北京: 北京邮电大学, 2023.
Dong Youheng. Research on Stream Cipher Based on Spatiotemporal Chaotic System and Cellular Automata[D]. Beijing: Beijing University of Posts and Telecommunications, 2023. (in Chinese)

[24] Kumar K, Roy S, Rawat U, et al. SOCIET: Second-order cellular automata and chaotic map-based hybrid image encryption technique[J]. Multimedia Tools and Applications, 2024, 83(10): 29455-29484.

[25] 张宇鹏, 高莹, 严宇, 等. ZUC算法软件快速实现[J]. 密码学报, 2021, 8(3): 388-401.
Zhang Yupeng, Gao Ying, Yan Yu, et al. Fast software implementation of ZUC algorithm[J]. Journal of Cryptologic Research, 2021, 8(3): 388-401. (in Chinese)

[26] Alharbi F, Hameed M K, Chowdhury A, et al. Analysis of area-efficiency vs. unrolling for eSTREAM hardware portfolio stream ciphers[J]. Electronics, 2020, 9(11): 1935.

[27] 刘云涛, 申泽生, 方硕, 等. 高吞吐率流水线结构的ZUC-256流密码硬件设计[J]. 电子学报, 2023, 51(2): 438-445.
Liu Yuntao, Shen Zesheng, Fang Shuo, et al. A hardware design of ZUC-256 stream cipher of pipelining structure with high throughput[J]. Acta Electronica Sinica, 2023, 51(2): 438-445. (in Chinese)

[28] Wang L, Jing J W, Liu Z B, et al. Evaluating optimized implementations of stream cipher ZUC algorithm on FPGA[M]//Information and Communications Security. Berlin, Heidelberg: Springer, 2011: 202-215.

作者简介

董有恒　男,1995年5月出生于山东省济宁市。2023年毕业于北京邮电大学网络空间安全学院。现为北京电子科技学院密码科学与技术系讲师。主要研究方向为序列密码设计及分析、混沌密码理论及其应用。
E-mail: Dyh_CHAOS@163.com

王彩冰　女,1996年4月生于山东省德州市。2023年毕业于中国科学院信息工程研究所。现为北京电子科技学院密码科学与技术系讲师。主要研究方向为密码设计与分析。
E-mail: wangcaibing@besti.edu.cn

张艳硕　男,1979年出生于陕西省宝鸡市。现为北京电子科技学院密码科学与技术系副教授、博士生导师。主要研究方向为密码理论及其应用。
E-mail: zhang_yanshuo@163.com

赵　耿　男,1964年出生于四川省广元市。北京电子科技学院网络空间安全系教授、博士生导师。主要研究方向为混沌密码理论及应用、信息安全等。
E-mail: zhaogeng@besti.edu.cn

鲁小娟　女,1992年11月出生于江苏省。2022年毕业于中科院信工所。现为北京电子科技学院密码科学与技术系讲师。主要研究方向为对称密码设计与分析。
E-mail: xjlu92@163.com

黄雅婷　女,1997年2月出生于山东省东营市。2022年毕业于北京电子科技学院。现为兴唐通信科技有限公司工程师。主要研究方向为密码设计与实现。
E-mail: HYT_SJS@163.com