密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research, 2021, 8(6): 1039–1057
©《密码学报》编辑部版权所有.
E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

流密码算法、架构与硬件实现研究*

赵石磊, 刘 玲, 黄 海, 徐 江, 刘志伟, 于 斌

哈尔滨理工大学 计算机科学与技术学院, 哈尔滨 150080
通信作者: 黄海, E-mail: ic@hrbust.edu.cn

摘 要: 流密码算法的设计与实现一直是密码学领域的研究热点之一. 随着应用环境的多样化以及安全需求的提高, 流密码算法的设计结构和高效实现也面临着新的挑战. 本文首先回顾流密码的发展历史, 介绍最新的研究成果, 整理流密码发展过程中部分主要国家和地区发起的流密码相关项目以及相关标准; 随后, 归纳流密码设计中常见的构建块, 并对流密码的设计作分类, 跟踪不同分类中典型算法的发展脉络, 同时对主流的设计分类进行分析、举例; 然后, 从可重构计算的角度对典型流密码的算法结构进行分解, 提取计算算子, 总结运算种类并分析基本运算单元的使用情况, 基于这些研究设计一个面向流密码算法的通用的可重构计算密码处理架构. 最后, 在分析当前流密码算法需要解决的一些关键问题后, 从不同角度展望流密码的发展趋势.

关键词: 流密码; 流密码设计; 流密码实现; 可重构

中图分类号: TP309.7 **文献标识码:** A DOI: 10.13868/j.cnki.jcr.000495

中文引用格式: 赵石磊, 刘玲, 黄海, 徐江, 刘志伟, 于斌. 流密码算法、架构与硬件实现研究[J]. 密码学报, 2021, 8(6): 1039–1057. [DOI: 10.13868/j.cnki.jcr.000495]

英文引用格式: ZHAO S L, LIU L, HUANG H, XU J, LIU Z W, YU B. Algorithm, architecture and hardware implementation of stream cipher[J]. Journal of Cryptologic Research, 2021, 8(6): 1039–1057. [DOI: 10.13868/j.cnki.jcr.000495]

Algorithm, Architecture and Hardware Implementation of Stream Cipher

ZHAO Shi-Lei, LIU Ling, HUANG Hai, XU Jiang, LIU Zhi-Wei, YU Bin

School of Computer Science and Technology, Harbin University of Science and Technology, Harbin 150080, China

Corresponding author: HUANG Hai, E-mail: ic@hrbust.edu.cn

Abstract: The design and implementation of stream cipher algorithms have always been research hotspots in the field of cryptography. Different application environments and the improvement of

* 基金项目: 黑龙江省自然科学基金优秀青年项目 (YQ2019F010); 国家重点研发计划 “光电子与微电子器件及集成” 重点专项子课题 (2018YFB2202100); 中央引导地方科技发展专项 (ZY20B11); 黑龙江省普通高校基本科研业务费专项资金 (LGYC2018JC016, 2019KYYWF0214)

Foundation: Excellent Youth Project of Heilongjiang Natural Science Foundation (YQ2019F010); National Key R&D Program “Optical Electronics and Microelectronic Devices and Integration” Key Special Sub-Project (2018YFB2202100); Special Program of Central Government Guiding Localities for Scientific and Technological Development (ZY20B11); Fundamental Research Funds for Higher Educations of Heilongjiang (LGYC2018JC016, 2019KYYWF0214)

收稿日期: 2020-11-18 定稿日期: 2021-11-06

security levels have also launched new challenges to the design structure and efficient implementation of stream cipher algorithms. This paper first reviews the development history of stream ciphers, introduces the latest research results, and lists some stream cipher related projects and related standards in the process of stream cipher development. Subsequently, the common building blocks in stream cipher design are summarized, and the design of stream cipher is classified. The development of typical stream cipher algorithms is classified in different categories. Moreover, the mainstream design classifications are analyzed and demonstrated with examples. Then, from the perspective of reconfigurable computing, the algorithm structure of typical stream ciphers is decomposed, the computing operators are extracted, the types of operations are summarized and the use of basic operation units is analyzed. Based on these, this paper designs a general reconfigurable cryptographic processing architecture for stream cipher algorithms. Finally, after analyzing some key problems that need to be solved in current stream cipher algorithms, the development trend of stream ciphers is presented from different angles.

Key words: stream cipher; design of stream ciphers; implementation of stream ciphers; reconfigurable

1 引言

网络的发展带来海量的机密信息传输, 密码学在日常生活中发挥着越来越重要的作用, 现代密码学发展的核心是保护数据信息的机密性、完整性、认证性和不可否认性等 [1]. 现代密码体制一般分为对称密码体制和非对称密码体制两大类, 其中对称密码又分为分组密码和流密码. 流密码因其具有良好保密性、轻量化和加密速度快等特点得到了广泛的应用.

流密码通常用于安全网络通信中保护通信数据, 尤其在军事、政府部门等领域. 应用比较广泛的流密码算法有: GSM 系统的 A5/1 算法 [2], 3GPP 标准中用于移动通信的 SNOW 3G [3] 和 ZUC 算法 [4], IEEE 802.11 中规定的安全机制 WEP、SSL/TLS 等标准协议采用的 RC4 算法 [5], 用于蓝牙加密系统的 E0 算法 [6], 还有目前应用极为广泛的 ChaCha20 算法 [7] 等. 谷歌选择 ChaCha20 算法和 Bernstein 的 Poly1305 消息验证码来代替 TLS 中的 RC4, ChaCha20-Poly1305 AEAD 密码套件已经在 TLS 1.3 中标准化 [8], ChaCha20 还被用于 FreeBSD、OpenBSD、NetBSD、Linux 内核等各种操作系统 [9]. 近年来, 随着通信和网络飞速发展, 流密码算法的应用越来越广泛. 为了满足现代应用的需求, 具有加密功能和消息验证功能的认证加密算法和适用于资源受限环境 (如射频识别 (radio frequency identification, RFID) 标签、传感器网络等) 的轻量级加密算法得到了人们的关注, 基于流密码设计的认证加密算法和轻量级流密码算法也成为流密码研究的重点对象, 一系列此类流密码算法被提出, 如 ACORN、Grain-128AEAD、Fruit-80、WAGE 等.

本文对传统流密码算法和近年来提出的基于流密码算法的认证加密算法和轻量级加密算法进行了分析, 对部分流密码算法的计算算子进行归纳, 设计一个面向流密码算法的通用的可重构计算密码处理架构. 本文组织结构如下: 第 2 节对流密码进行梳理, 包括对流密码相关项目及标准的总结、流密码设计构建块的分析、流密码设计的分类及典型流密码算法的结构分析等; 第 3 节分析流密码算法的实现思想及其可重构特性, 总结基本运算操作的使用情况, 设计面向流密码算法的可重构计算密码处理架构; 第 4 节尝试从不同角度展望流密码的发展趋势.

2 流密码概述

1917 年, Vernam 密码体制被提出, 后来 Mauborgne 基于该体制提出了改进方案, 也就是 “一次一密” 密码体制, 这可以看作流密码的最早起源 [10]. 1949 年, Shannon 证明 “一次一密” 密码体制是理论安全加密体制 [11], 但是, 其密钥流必须完全随机生成, 长度至少与明文相同, 不能多次使用, 这在实际应用中存在很大困难, 而且, 密钥的生成、分配和管理是一个不容忽视的问题. 因此, 人们设计了各种流密码算法来代替 “一次一密” 体制 [12].

2.1 流密码相关项目及标准

自现代密码发展开始, 为了保护信息安全, 各个国家对密码理论与技术的研究都十分重视, 许多国家和地区发起了很多大大小小的密码相关的项目、竞赛等, 其中相当一部分涉及流密码算法, 如图 1 所示.

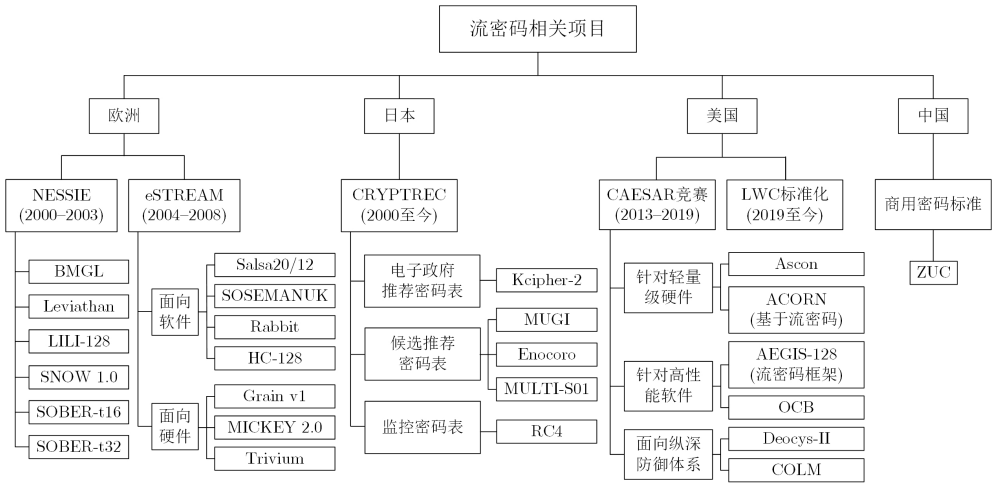


图 1 流密码相关项目

Figure 1 Projects related to stream ciphers

2000 年, 欧盟启动了 NESSIE 项目 [13], 该项目为期 3 年, 旨在促进美国 NIST 组织的分组密码 AES 标准化的最后阶段, 同时发起了一项独立的公开呼吁, 通过评估所提交的加密原语, 提出一个各种类型的强大的密码原语组合. NESSIE 共评估了 6 个流密码算法, 分别是 BMGL、Leviathan、LILI-128、SNOW 1.0、SOBER-t16 和 SOBER-t32, 但这 6 个算法在安全性方面存在一些问题, 最终均未入选 [13].

2004 年, 由欧盟资助的 ECRYPT 宣布了 eSTREAM 项目 [14], 其发起建立在 2000 年提交给 NESSIE 的 6 个流密码算法都失败的基础上, 目的是促进设计更适合广泛应用的高效、紧凑的流密码. eSTREAM 持续到 2008 年, 共征集到 34 个算法, 经过多次安全和性能评估, 最终选择了其中 7 种. 具体来讲, 征集的流密码算法分为更适合于具有高吞吐量要求的软件应用的流密码算法和适用于如有限的存储、门数、功耗等资源受限的硬件应用的流密码算法, 最终面向软件的胜选算法为 Salsa20/12、SOSEMANUK、Rabbit、HC-128, 面向硬件的胜选算法为 Grain v1、Trivium、MICKEY v2.

2000 年, 日本政府提出了 CRYPTREC 项目 [15], 其目的是评估和推荐一些密码算法用于日本电子政务系统. CRYPTREC 密码表包括三部分: 电子政府推荐密码表、候选推荐密码表、监控密码表, 评估推荐的流密码算法有 KCipher-2、MUGI、Enocoro、MULTI-S01 等. 2013 年, CRYPTREC 成立了轻量级密码学工作组, 旨在为日本电子政务系统和某些应用程序评估轻量级密码解决方案. 2015 年, 工作组发布了轻量级密码学及应用现状的评估报告, 报告中评估的轻量级密码算法多数为分组密码.

2013 年, 由 NIST 资助, Bernstein 等人发起了 CAESAR 竞赛 [16], 该竞赛主要征集优于 AES-GCM 且适合广泛应用的认证加密算法, 认证加密算法可以同时实现数据机密性、完整性、数据源认证 [17]. CAESAR 竞赛持续到 2019 年, 最终有 6 个算法胜出, 分别是针对轻量级硬件应用的 Ascon 和 ACORN、针对高性能软件应用的 AEGIS-128 和 OCB 以及面向纵深防御体系、健壮性良好的 Deoxys-II 和 COLM. 其中 AEGIS-128 整体采用了流密码的框架, ACORN 是基于流密码设计的认证加密算法 [17].

2013 年, NIST 发起了一项轻量级密码 (lightweight cryptography, LWC) 标准化的项目, 以征集、评估和标准化适用于当前 NIST 密码标准性能不可接受的受限环境的轻量级密码算法, 从而确保受限环境中的机密性、真实性和完整性 [18]. 2015 年, NIST 举办了首届轻量级密码学研讨会. 2018 年, NIST 宣布了轻量级密码标准化的最终提交要求和评估标准, 并呼吁提供相关数据认证加密和可选散列功能的密码算法. 2019 年, LWC 标准化项目共征集到 57 种算法, 第一轮共评估了 56 种算法, 但是只有 32 种被选中继续进行第二轮. 2021 年 3 月, NIST 宣布 ASCON、Elephant、GIFT-COFB、Grain128-AEAD、ISAP、

Photon-Beetle、Romulus、Sparkle、TinyJambu、Xoodyak 等 10 个轻量级密码算法入围决赛, 预计最后一轮持续大约 12 个月. 其中入围决赛的 Grain-128 AEAD 是基于流密码设计的轻量级认证加密算法.

在国内, 2011 年, 我国自主研发的流密码算法 ZUC 正式被 3GPP SA 全会通过, 成为 3GPP LTE 机密性和完整性算法第三套加密标准核心算法, 被采纳为国际加密标准, 即第四代移动通信加密标准 [4]. 2018 年, ZUC 算法研制组提出了与 ZUC-128 高度兼容的 ZUC-256 流密码算法, 提供 5G 应用环境下 256 bit 密钥的安全性 [19].

表 1 列出了上述几个项目及涉及到流密码算法的 ISO/IEC 标准的相关信息.

表 **1** 流密码相关项目及标准化

Table 1 Projects and standardization related to stream ciphers

项目/竞赛/标准	时间	类型	相关流密码算法
NESSIE [13]	2000–2003	评估密码算法	BMGL、Leviathan、LILI-128、SNOW 1.0、SOBER-t16、SOBER-t32
CRYPTREC [15]	2000 至今	评估、推荐密码算法	KCipher-2、MUGI、Enocoro、MULTI-S01、RC4
eSTREAM [14]	2004–2008	征集、评估流密码算法	Salsa20、SOSEMANUK、Rabbit、HC-128、Grain、MICKEY、Trivium 等
CAESAR [16]	2013–2019	征集、评估认证加密算法	ACORN、AEGIS-128 等 (基于流密码的认证加密算法)
LWC 标准化 [18]	2019 至今	标准化轻量级密码算法	Grian-128AEAD、WAGE、Triad、Bleep64、Quartet 等
ISO/IEC 18033-4 [20]	2011	流密码标准化	MUGI、SNOW 2.0、Rabbit、Decim v2、KCipher-2
ISO/IEC 29192-3 [21]	2012	轻量级流密码标准化	Enocoro、Trivium
ISO/IEC 29167-13 [22]	2015	流密码标准化	Grain-128a
ISO/IEC 18033-4/AMD1[23]	2020	流密码标准化	ZUC

2.2 流密码的设计结构

一般认为, 一个安全的流密码所生成的密钥流序列具备长周期、高线性复杂度、良好统计特性等特点 [1], 在设计流密码时也是以这些特点作为基础的, 核心部件的安全程度对算法整体安全性产生重要影响. 下面介绍流密码设计中的一些构建块.

2.2.1 流密码构建块

从 NESSIE 等标准化项目征集到的流密码以及公开的各种流密码来看, 流密码设计中的构建块大致包括反馈移位寄存器 (feedback shift register, FSR)、布尔函数、状态表、ARX 结构、细胞自动机 (cellular automata, CA)、混沌映射、T 函数、PANAMA 结构、置换滤波生成器、海绵结构、非线性滤波函数、非线性组合函数、钟控、S 盒、有限状态机 (finite state machine, FSM) 等. 流密码的构建块有的用来产生状态序列, 有的用作非线性部分. 图 2 给出了常见的流密码构建块.

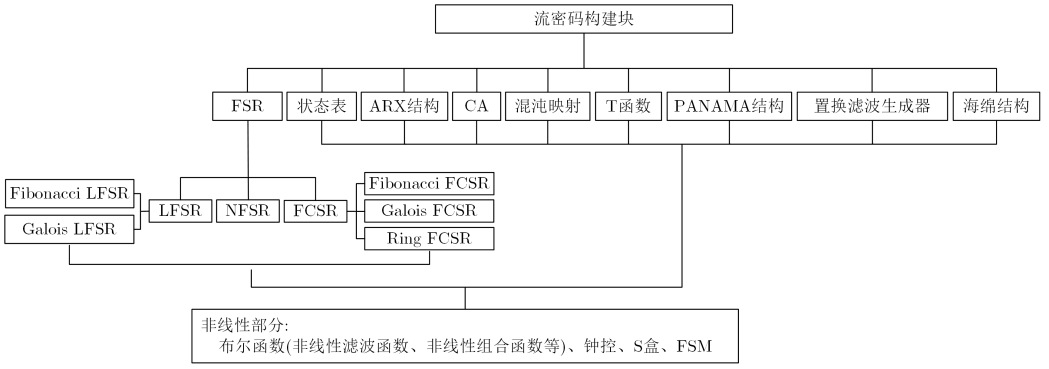


图 2 流密码构建块

Figure 2 Building blocks of stream ciphers

FSR. FSR 是流密码设计中非常常用的构建块, 可以生成具有良好特性的状态序列, 根据反馈函数的不同可以分为线性反馈移位寄存器 (linear feedback shift register, LFSR)、非线性反馈移位寄存器 (nonlinear feedback shift register, NFSR)、带进位的反馈移位寄存器 (feedback with carry shift register, FCSR). LFSR 有两种表示形式: Fibonacci LFSR 和 Galois LFSR, LFSR 生成的序列是线性的, 不能直接作为密钥流序列输出, 需要其他构建块提高其线性复杂度; NFSR 生成的序列是非线性的, 所以 NFSR 本身也可作为设计流密码的非线性构建块; FCSR 有三种表示形式: Fibonacci FCSR、Galois FCSR 和 Ring FCSR [24], FCSR 类似于 LFSR, 可以生成具有良好伪随机性的状态序列, 另外, FCSR 的状态转移函数是非线性的, 生成的状态序列的线性复杂度较高, 但是 FCSR 生成的序列具有可预测性, 所以也不能直接作为密钥流使用. FSR 的硬件结构简单, 可以很容易地使用基本逻辑门来构造, 很适合于硬件实现.

布尔函数. 布尔函数是流密码中非常重要的构建块, 1917 年, Vernam 密码中就出现了布尔逻辑结构, 布尔函数在 LFSR 中的应用是其在流密码中真正作为研究对象的始源 [25]. Berlekamp-Massey 算法的提出使人们意识到 LFSR 产生的状态序列的线性性质对密码分析没有任何免疫力, 需要提高 LFSR 状态序列的线性复杂度才能满足流密码的安全性. 布尔函数的性质使其在流密码的设计与分析中起着关键作用, 其主要性质归纳起来有平衡性、对称性、高代数次数、高非线性、相关免疫性、弹性阶、自相关性、扩散性等 [26]. 布尔函数包括单输出布尔函数和多输出布尔函数, 流密码中用于提高 LFSR 状态序列线性复杂度的非线性滤波函数和非线性组合函数, 对其研究可以归结为对布尔函数的研究; 另外, 在流密码中对用于提高非线性的 S 盒的研究可归结为对多输出布尔函数的研究 [27]. 不同密钥流生成器中布尔函数应该满足不同的性质才能保证生成的密钥流具有更高的安全性, 在进行流密码设计时, 可以选择目前公认的各方面性质较为平衡的布尔函数作为构建块 [28]. 对 NFSR 中非线性反馈函数的研究是布尔函数的一个很重要的研究方向 [25].

状态表. 将流密码的状态序列存储到状态表中, 由非线性函数更新状态表中的状态序列. 状态表作为设计流密码的构建块, 其软件实现速度快、硬件存储消耗较高.

ARX 结构. ARX 是指模加 (modular add)、循环移位 (rotation) 和逻辑异或 (xor) 三种运算, 三种运算的实现速度较快, 可以将 ARX 三种混合运算应用到流密码的设计中.

CA. 1985 年, Wolfram 将 CA 应用在密码学领域 [29], 自此 CA 在密码学方面的研究逐渐开展 [30–32]. CA 是自动机理论中研究的一种离散计算模型, 由规则的细胞网格组成, 每个细胞都处于有限数量的状态之一, 通过某种固定规则, 根据细胞的当前状态和其邻域中的细胞的状态来确定每个细胞的新状态, 二维细胞自动机可以用来构造伪随机数发生器 [33].

混沌映射. 1982 年, OISHI 等将混沌系统应用到流密码中 [34], 混沌映射具有明显的伪随机性和不规则状态, 可以使用混沌映射的控制参数和初始条件作为流密码的密钥, 从更广泛的角度来看, 混沌映射与密码系统之间的相似性是设计基于混沌映射的流密码算法的主要动机, 混沌理论可以很好地对对称密码中的扩散和混淆进行建模 [35].

T 函数. T 函数是 Klimov 和 Shamir [36] 提出的一种能够在任意长的字上混合布尔运算和算术运算的密码构建块, 属于一类可逆映射, T 函数因其双射特性可以用于流密码中产生状态序列 [37], 可以在软件上快速实现, 利用 T 函数生成的状态序列具有长周期、良好的非线性和非代数性, Klimov 和 Shamir 指出 T 函数可以作为 LFSR 的替代, 作为基于软件的密钥流生成器的构建块 [36].

PANAMA 结构. 1998 年, Daemen 提出了流密码算法 PANAMA [38], PANAMA 采用了一种新的设计结构, PANAMA 结构的内部状态机由内部状态 S (状态 a 和缓冲区 b) 及其更新函数 F (状态 a 和缓冲区 b 的更新函数分别为 ρ 和 λ) 组成 [38]. 由 (a,b)、(ρ,λ) 和从 S 提取输出序列的输出滤波器 f 组成的密钥流生成器如果满足以下条件, 则称为 PANAMA 式密钥流生成器: ρ 包括 SPN 变换, 类似于分组密码的轮函数, 并使用缓冲区 b 的一部分作为参数 $a^{(t+1)}=\rho(a^{(t)},b^{(t)})$; λ 是线性变换, 并使用状态 a 的一部分作为参数 $b^{(t+1)}=\lambda(b^{(t)},a^{(t)})$; f 每轮提取输出状态 a 的一部分 (通常不超过 1/2) [39].

置换滤波生成器. 置换滤波生成器是在 Eurocrypt 2016 上提出的一种新的流密码结构, 它由三部分组成: 用于存储密钥的寄存器、由伪随机数生成器 (由公共向量初始化) 参数化的置换生成器, 生成密钥流的滤波函数. 与滤波生成器相比, 置换滤波生成器中 LFSR 被密钥寄存器替换, 换句话说, 寄存器不再通过 LFSR 更新, 而是使用伪随机位置换进行更新, 也就是在每个周期, 即每次滤波函数输出一位时, 伪随机

置换被应用于寄存器, 并且非线性滤波函数作用于被置换的密钥寄存器的输出. 这种结构的优点是始终将非线性滤波函数直接作用于密钥位, 使得输出噪声始终是一个常数, 并且具有很强的同态能力 [40]. 从目前的分析来看, 这类结构对滤波函数的选择要求很高, 否则很容易被分析而导致大量密钥信息泄漏 [41].

海绵结构. 海绵结构是 Bertoni 和 Daemen 在 ECRYPT 2007 上提出的一种模型 [42], 是一种基于定长置换和填充规则的操作模式, 海绵结构构建将可变长度的输入映射到可变长度输出的函数, 这样的函数称为海绵函数. 海绵函数是具有固定输出长度的散列函数和具有固定输入长度的流密码的泛化, 它通过迭代地应用内部置换来对有限状态进行操作, 该内部置换与输入数据或输出数据进行交错 [43]. 海绵函数使用随机置换, 接受可变长度的输入和输出, 这一点使其适合作为设计流密码的构建块 [44]. 用作流密码构建块时, 输入是初始密钥和初始向量, 然后在压缩阶段输出得到密钥流. 海绵结构的优点是设计相对简单, 灵活性较高 [43].

非线性滤波函数、非线性组合函数、钟控、S 盒、FSM 一般用作流密码非线性部分的构建块, 非线性部分主要用来引入或提高状态序列的非线性. 非线性滤波函数、非线性组合函数、钟控、FSM 主要是针对基于 LFSR 设计的流密码而引入的, 目的是为了降低 LFSR 序列线性的弱点, 引入非线性. 非线性滤波函数和非线性组合函数主要通过作用于 LFSR 序列来破坏其线性特性; 钟控是通过对 LFSR 进行不规则的时钟控制来提高其非线性; FSM 是计算的数学模型, 在任何给定时刻都可以处于有限状态之一, FSM 由其状态列表、初始状态和触发每次转换的输入来定义, 可以根据某些输入从一种状态转换为另外一种状态 [45]; 对称密码算法中最常见的混淆方式就是利用 S 盒, 因为 S 盒输入的微小变化就会导致输出的复杂变化, 许多流密码也使用 S 盒来引入非线性.

2.2.2 流密码设计分类

流密码由状态更新函数和输出函数组成, 流密码的状态序列在加密期间不断更新, 使得被加密消息在不同位置的比特以不同的状态加密, 输出函数根据状态序列生成密钥流序列, 并执行加密和解密 [27]. 状态更新的基本要求是应以足够大的周期生成状态, 利用流密码的构建块可以设计状态更新函数生成状态序列, 然后利用非线性构建块提高状态序列的线性复杂度, 以生成高安全级别的密钥流序列 [46]. 目前流密码主流的设计结构有基于 LFSR 的设计、基于 NFSR 的设计、基于状态表驱动的设计、基于分组密码的设计几类. 图 3 给出了流密码的设计分类.

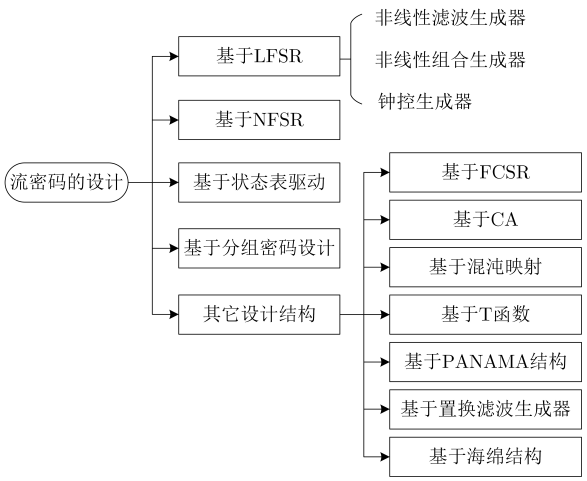


图 3 流密码的设计分类

Figure 3 Design classification of stream ciphers

基于 LFSR 的设计. 20 世纪 40 年代电子计算机被发明, 计算技术的进步使密码学家可以使用比计数器更安全的操作来设计流密码, 基于 LFSR 设计的流密码就出现在此时期 [46]. Rueppel [27] 将密钥流生成器分为驱动部分和非线性部分, 驱动部分主要用来将初始密钥扩散成具有良好周期性和统计特性的状态序列, 非线性部分对驱动部分的输出序列进行非线性运算, 提高其线性复杂度和不可预测性, 从而生成

最终密钥流. 驱动部分一般利用 LFSR 产生状态序列, LFSR 序列为线性, 则非线性部分采用各种非线性构建块 (如非线性滤波函数、非线性组合函数等布尔函数或时钟控制等) 来掩盖其线性, 以生成高度非线性的密钥流. 根据所采用的非线性构建块的不同, 出现了非线性滤波生成器、非线性组合生成器、钟控生成器等各种密钥流生成器 [47]. 如何利用非线性布尔函数掩盖 LFSR 序列的线性是保证这类流密码安全性的首要目标, 这类流密码的结构代表了传统流密码的设计思想, 对其理论分析也比较成熟. 图 4 所示为常见的基于 LFSR 的流密码算法.

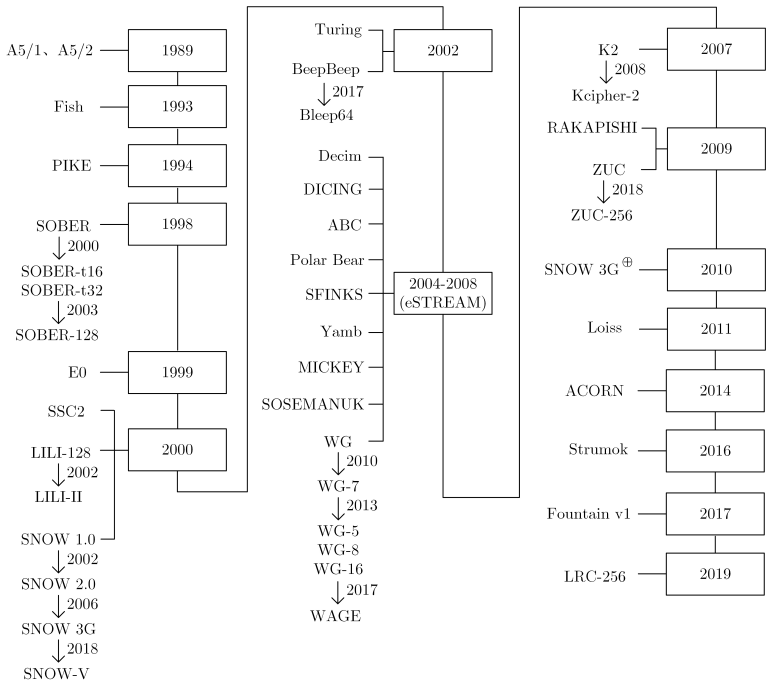


图 4 基于 LFSR 的流密码算法

Figure 4 Stream cipher algorithms based on LFSR

基于 NFSR 的设计. 自 eSTREAM 项目以来, 出现了很多流密码的设计理念与方法, 其中比较典型的一类是基于 NFSR 设计的流密码. 这类设计是把 NFSR 的非线性反馈与非线性输出相结合来提供良好的序列特性和安全性 [12]. 在硬件上, NFSR 比 LFSR 稍复杂, NFSR 对密码分析的抵抗能力较强, 基于 NFSR 设计的流密码的安全级别取决于分析设计本身的难度, 由于目前还没有简单且精确的准则来构建强大的非线性布尔函数, 所以较难设计出性能良好的 NFSR [46]. 目前关于 NFSR 的理论研究还比较少, 还没有统一的模式对这类流密码算法进行安全性分析 [25]. 对 NFSR 的非线性反馈函数的研究是一个值得深入的研究方向. 图 5 所示为常见的基于 NFSR 的流密码算法.

基于状态表驱动的设计. 基于状态表驱动设计流密码的思想来源于 1987 年 Rivest 设计的 RC4 流密码算法, 该类设计是利用状态表的转换和选择来构造流密码, 换句话说, 就是将很多操作进行预计算, 产生随机排列, 存储在状态表中, 便于在计算中使用. 基于状态表驱动设计的流密码算法包含较大的状态表且状态随时间持续变化, 所以硬件实现代价较高, 但是软件实现速度较快. 图 6 所示为常见的基于状态表驱动的流密码算法.

基于分组密码的设计. 基于分组密码的设计可以利用成熟的分组密码部件、结构, 或者利用已有的成熟的分组密码理论来构造流密码 [12]. 可以利用分组密码的某些操作模式生成密钥流序列, 比如当分组密码以输出反馈模式、密文反馈模式、计数器模式运行时可以用作流密码使用; 可以利用分组密码的设计结构, 比如采用分组密码的 Feistel 结构、SPN 结构等; 可以利用分组密码的非线性部件 S 盒; 可以利用分组密码中行移位、列混淆的设计思想; 可以直接输出分组密码的某些中间状态, 通过简单运算作为密钥流

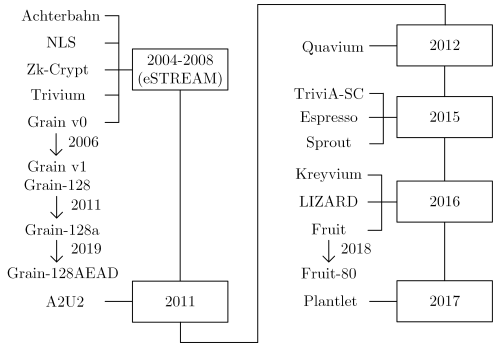


图 5 基于 NFSR 的流密码算法

Figure 5 Stream cipher algorithms based on NFSR

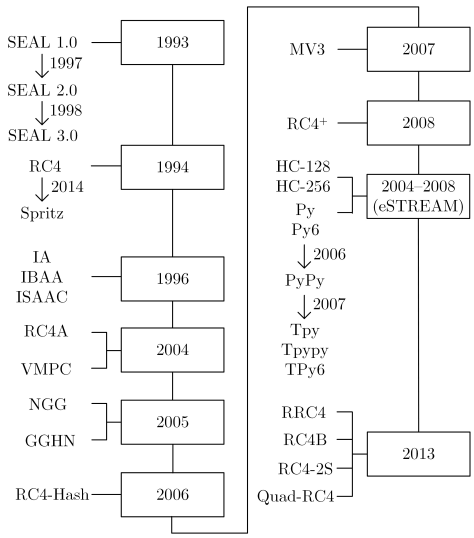


图 6 基于状态表驱动的流密码算法

Figure 6 Stream cipher algorithms based on state table driver

输出 [27, 41]. 基于分组密码设计的流密码算法软件实现速度很快, 这类流密码算法的内部状态较大, 而且不存在移位寄存器, 很难用传统的分析方法对其进行评估 [12]. 图 7 所示为常见的基于分组密码设计的流密码算法.

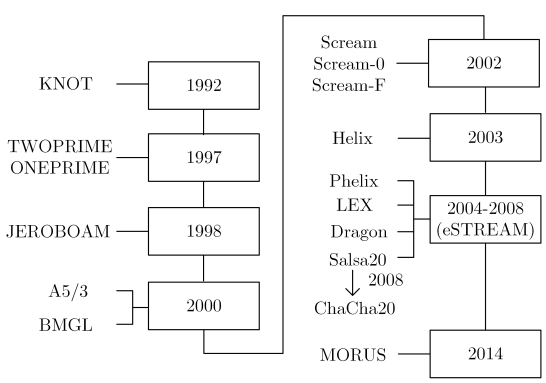


图 7 基于分组密码设计的流密码算法

Figure 7 Stream cipher algorithms based on block cipher

流密码的设计方法灵活多变, 设计结构也趋于多样化, 除了上述主流设计结构外, 还有一些基于其他结构设计的流密码. 如基于 FCSR 的流密码、基于 CA 的流密码、基于混沌映射的流密码、基于 T 函数的流密码、基于 PANAMA 结构的流密码、基于置换滤波生成器的流密码、基于海绵结构的流密码等.

按时间顺序对基于上述不同设计结构的典型流密码算法进行举例, 包括算法的结构、初始密钥 Key 及初始向量 IV 的长度等. 如表 2–6 所示.

流密码通过不断变化的内部状态进行采样来生成密钥流, 该状态通常是在初始密钥 Key 和 IV 的作用下进行初始化. 大多数流密码算法的初始密钥长度集中在 80 bit、128 bit、256 bit. 传统加密主要用于高端设备领域, 如服务器、台式计算机、平板电脑、智能手机等 [48], 传统流密码算法的初始密钥长度一般为 128 bit 或 256 bit. 现在, 随着物联网的发展, 像 RFID 标签、嵌入式系统、工业控制器、无线传感器网络、智能卡等小型计算设备变得越来越普遍, 传统加密很难在这类资源受限的设备中实现, 所以对轻量级加密的需求明显增加, 轻量级加密是专门为资源受限的设备量身定制的解决方案, 由于许多受限设备的

性质, 比如有限的存储、门数等, 在轻量级应用中, 功耗和能耗是相关的性能指标, 而且高吞吐量可能不是主要的设计目标, 大多数应用程序需要中等吞吐量. 轻量级流密码算法的初始密钥长度相对较短, 通常为 80 bit. eSTREAM 项目中面向硬件的胜选算法 Grain v1、Trivium、MICKEY 2.0 可以看作是出现较早的适用于受限硬件设备的轻量级流密码算法.

表 **2** 基于 LFSR 的流密码算法

Table 2 Stream cipher algorithms based on LFSR

流密码算法	时间	Key/bit	IV/bit	结构	来源、应用
A5/1、A5/2 [2]	1989	64	22	LFSR	GSM 通信
E0 [6]	1999	128	–	LFSR	蓝牙网络规范
LILI-128 [49]	2000	128	128	LFSR	NESSIE
SNOW 2.0 [50]	2002	128, 256	128	LFSR	ISO/IEC 18033-4
SNOW 3G [3]	2006	128	128	LFSR	3GPP 标准算法
SNOW-V [51]	2018	256	128	LFSR	5G
Decim v2 [52]	2005	80	64	LFSR	eSTREAM; ISO/IEC 18033-4
SOSEMANUK [53]	2005	128–256	64	LFSR	eSTREAM
MICKEY 2.0 [54]	2006	80	80	LFSR	eSTREAM
KCipher-2.0 [15]	2008	128	128	LFSR	CRYPTREC; ISO/IEC 18033-4
ZUC [4]	2009	128	128	LFSR	3GPP 标准算法
ZUC-256 [19]	2018	256	184	LFSR	5G
ACORN [55]	2014	128	128	LFSR	CAESAR
Strumok [56]	2016	256, 512	256	LFSR	乌克兰对称加密标准候选

表 **3** 基于 NFSR 的流密码算法

Table 3 Stream cipher algorithms based on NFSR

流密码算法	时间	Key/bit	IV/bit	结构	来源、应用
Trivium [57]	2005	80	80	NFSR	eSTREAM
Grain v1 [58]	2006	80	64	NFSR+LFSR	eSTREAM
Grain-128a [59]	2011	128	96	NFSR+LFSR	ISO/IEC 29167-13
Grain-128AEAD [60]	2019	128	96	NFSR+LFSR	LWC 标准化
Espresso [61]	2015	128	96	NFSR	5G
Sprout [62]	2015	80	80	NFSR+LFSR	小状态流密码
LIZARD [63]	2016	120	64	NFSR	小状态流密码
Plantlet [64]	2016	80	80	NFSR+LFSR	小状态流密码
Fruit-80 [65]	2018	80	70	NFSR+LFSR	小状态流密码

表 **4** 基于状态表驱动的流密码算法

Table 4 Stream cipher algorithms based on state table driver

流密码算法	时间	Key/bit	IV/bit	结构	来源、应用
RC4 [5]	1987	8–2048	–	状态表驱动	1994 年公布
SEAL 1.0 [66]	1993	16	–	状态表驱动	FSE
HC-128 [67]	2004	128	128	状态表驱动	eSTREAM
Py [68, 69]	2005	2048	2048	状态表驱动	eSTREAM
Spritz [70]	2014	2048	–	状态表驱动	RC4 变体

表 **5** 基于分组密码设计的流密码算法
Table 5 Stream cipher algorithms based on block cipher

流密码算法	时间	Key/bit	IV/bit	结构	来源、应用
A5/3 [71]	2000	64	22	基于分组密码	3G
Scream [72]	2002	128	128	基于分组密码	FSE
Helix [73]	2003	256	128	基于分组密码	FSE
Dragon [74]	2005	128, 256	128, 256	基于分组密码	eSTREAM
Phelix [75]	2005	256	128	基于分组密码	eSTREAM
LEX [76]	2005	128, 256	128, 256	基于分组密码	eSTREAM
Salsa20 [77]	2005	128, 256	64	基于分组密码	eSTREAM
ChaCha20 [7]	2008	128, 256	64	基于分组密码	TLS 1.3 标准化; RFC7539 标准化
MORUS [78]	2014	128, 256	–	基于分组密码	CAESAR

表 **6** 基于其他结构设计的流密码算法
Table 6 Stream cipher algorithms based on other structures

流密码算法	时间	Key/bit	IV/bit	结构	来源、应用
Chameleon [79]	1997	–	–	T 函数	FSE
PANAMA [38]	1998	256	–	PANAMA 结构	FSE
MUGI [39]	2002	128	128	PANAMA 结构	CRYPTREC; ISO/IEC 18033-4
Enocoro-80 [80]	2007	80	64	PANAMA 结构	CRYPTREC; ISO/IEC 29192-3
Enocoro-128 v2 [81]	2010	128	64	PANAMA 结构	CRYPTREC; ISO/IEC 29192-3
Rabbit [35]	2003	128	64	混沌映射 +ARX 结构	eSTREAM; ISO/IEC 18033-4
Hiji-bij-bij (HBB) [82]	2003	128, 256	–	CA	INDOCRYPT; eSTREAM
CAR30 [83]	2013	128	120	CA	Cryptography and Communications
F-FCSR [84]	2009	80, 128	80, 128	FCSR	eSTREAM
ICEPOLE [85]	2014	128, 256	96	海绵结构	CHES
FLIP [86]	2016	80, 128	–	置换滤波生成器	EUROCRYPT

3 流密码算法的实现

密码处理器作为密码算法的实现载体, 为信息安全提供基础设施服务 [87]. 流密码算法的实现包括硬件实现和软件实现. 对于嵌入到系统软件的加密程序而言, 软件实现效率更高; 对于高速网络中数据流的传输而言, 硬件实现效率更高. 流密码算法的实现主要利用实现密码应用的各类集成电路载体实现, 包括通用处理器 (general purpose processor, GPP)、专用集成电路 (application specific integrated circuit, ASIC)、可编程逻辑门阵列 (field programmable gate array, FPGA)、可重构计算密码处理器几种形式. 利用 GPP 可以实现多种流密码算法, 灵活性较高, 但 GPP 基于指令流驱动, 实现速度比较慢. ASIC 主要针对特定目标流密码算法设计专用的硬件电路来实现, 速度快、功耗低、面积小, 但一旦流片, 将无法改变电路的功能, 灵活性较差, 安全性低. 利用 FPGA 可以实现各种流密码算法, 灵活性很高, 速度比较折中, 介于 GPP 和 ASIC 之间, 但 FPGA 的配置信息比较繁琐, 不能实时改变, 而且资源有限, 资源丰富的 FPGA 成本又比较高. 随着网络信息技术的发展, 一个常用的信息安全解决方案可能会用到很多种密码算法, 为了支持应用或协议中尽量多的流密码算法, 需要密码处理器有足够的灵活性, 此外, 芯片设计的两个核心指标是性能和功耗, 性能功耗比, 也就是能量效率, 自然成为比纯粹的性能更为合理严格的指标, 安全性也是密码处理器中一项很重要的指标. 可重构计算密码处理器是一种性能、功耗、速度、灵活性合理折中的密码算法实现方式, 主要面向领域, 面积有时会冗余, 但现代工艺允许不关心面积冗余这一弊端 [88].

所以从应用和协议的角度来看, 在保证一定处理速度的情况下又能够满足一个信息安全解决方案对多种流密码算法的需求, 可重构计算密码处理器是一种很好的选择. 表 7 总结了上述流密码实现方式的优缺点.

表 7 GPP、ASIC、FPGA、可重构计算密码处理器比较

Table 7 Comparison of GPP, ASIC, FPGA, and reconfigurable computing cryptographic processor

硬件实现方式	优点	缺点
GPP	灵活性高	速度慢
ASIC	速度快、功耗低、面积小	灵活性差、安全性低
FPGA	灵活性高、速度相对折中	配置信息繁琐、成本高
可重构计算密码处理器	性能、功耗、速度、灵活性折中	面积有时冗余

3.1 流密码算法特征分析

虽然根据设计方法和结构可以将流密码细分为很多类, 但很多流密码算法有相似的密码处理结构和操作类型, 所以不同的流密码算法存在大量的共性逻辑. 如前所述, 主要的设计结构有基于 LFSR 的流密码算法、基于 NFSR 的流密码算法、基于状态表驱动的流密码算法和基于分组密码设计的流密码算法, 对主流的四类设计中典型的流密码算法如 A5/1、E0、Grain、Trivium、ZUC、SNOW、RC4、ChaCha20 等进行研究, 分析算法状态更新和密钥生成所采用的构建块, 从利用可重构计算思想实现流密码算法的角度出发, 对算法的结构特征、操作特征和计算算子进行总结归纳, 发现它们有多种类型的共性逻辑, 包括 LFSR、NFSR、逻辑运算 (包括异或、与、或、非)、算术运算 (包括加法、模加、模乘)、移位 (包括逻辑左移、逻辑右移、循环左移、循环右移)、S 盒、有限域 GF (2^n) $(n = 8, 16, 32, 64)$ 乘法运算等. 对有限域 GF (2^n) 乘法运算进一步分解, 可以得到更简单、重用程度更高的基本操作, 如 S 盒、异或、加法、乘法、移位等 [87]. 如表 8 所示对典型流密码算法特征进行了分析总结, 其中操作位宽和输出位宽以 bit 为单位; nXOR 表示 n-bit 按位逻辑异或; nAND 表示 n-bit 按位逻辑与; nOR 表示 n-bit 按位逻辑或; nNAND 表示 n-bit 按位与非; $\ggg$ 表示循环右移; $\lll$ 表示循环左移; $\ll$ 表示逻辑右移, S 盒一列中 $n \times m$ 表示 n-bit 输入, m-bit 输出的查找表运算.

通过分析典型流密码算法, 对其状态更新部分和密钥生成部分的结构进行分解, 可以分为有限域 GF(2) 和 GF (2^n) 两种情况进行分析: GF(2) 上流密码算法的数据操作位宽一般为 1 bit, 一次产生 1 bit 密钥, 主要计算算子为逻辑运算, 包括逻辑异或、逻辑与等. GF (2^n) 上流密码算法的数据操作位宽一般为字节或字节的整数倍, 这些流密码算法的非线性部分一般由多种非线性变换相互结合, 涉及到的计算算子较复杂, 包括逻辑运算、算术运算、移位、S 盒等. 整体来看, 基于 LFSR 和基于 NFSR 的流密码算法用到了 FSR 构建块, 不同流密码算法 FSR 的个数和级数不同, FSR 个数大多不超过 4, 级数大多不超过 160 bit, FSR 总容量不超过 1024 bit; 流密码算法中的逻辑运算和算术运算的操作粒度大多为单比特、字节或字节的整数倍, 一般为 1~32 bit; 很多流密码算法使用了逻辑移位和循环移位, 移位的位数一般不超过 32 bit; 不同流密码算法中的 S 盒的输入输出位宽不同, 输入位宽以 8 bit 居多, 输出位宽差别较大, 主要包括 4 bit、8 bit、16 bit、32 bit 四种.

3.2 可重构计算流密码处理架构

根据对不同流密码算法的结构特征、操作特征和计算算子的统计, 分析数据存储、重构粒度、运算单元 (process element, PE)、互连网络、运算单元阵列、配置信息等关键技术, 设计一种通用的面向四类主流设计结构的流密码算法的可重构计算密码处理架构, 包括可重构数据通路和可重构控制器两部分. 可重构数据通路负责流密码数据流的处理, 可重构控制器负责可重构数据通路的配置切换和调度 [87]. 所设计的可重构计算流密码处理架构结构框图如图 8 所示.

可重构数据通路包括可重构反馈移位寄存器阵列、抽头抽取结构、可重构运算单元阵列、配置接口、反馈数据选择、密钥流数据选择、输入数据存储器、密钥流存储器几个主要模块. 各模块设计及功能具体描述如下:

可重构反馈移位寄存器阵列主要用来存储流密码的状态向量, 共包含 32 个 32 bit 的移位寄存器单元阵列, 可以根据配置信息实现寄存器单元之间不同方式的级联以及不同方向、不同粒度的移位, 支持 1 bit、

表 **8** 流密码算法特征分析

Table 8 Analysis of algorithm characteristics of stream ciphers

流密码算法	操作位宽	**FSR**	**FSR** 级数	逻辑运算	模加	乘法	移位	**S** 盒	输出
A5/1 [2]	1	3LFSR	19, 22, 23	1XOR	–	–	–	–	1
LILI-128 [49]	1	2LFSR	128, 127	1XOR	–	–	–	–	1
Decim v2 [52]	1	1LFSR	192	1XOR	–	–	–	–	1
Grain v1 [58]	1	1LFSR+1NFSR	80, 80	1XOR, 1AND	–	–	–	–	1
Grain-128 [89]	1	1LFSR+1NFSR	128, 128	1XOR, 1AND	–	–	–	–	1
Trivium [57]	1	3NFSR	93, 84, 111	1XOR, 1AND	–	–	–	–	1
MICKEY 2.0 [54]	1	2FSR	100, 100	1XOR, 1AND	–	–	–	–	1
MICKEY-128 [90]	1	2FSR	160, 160	1XOR, 1AND	–	–	–	–	1
A2U2 [91]	1	1LFSR+2NFSR	17, 9, 7	1XOR, 1NAND	–	–	–	–	1
Espresso [61]	1	1NFSR	256	1XOR, 1AND	–	–	–	–	1
Sprout [62]	1	1LFSR+1NFSR	40, 40	1XOR, 1AND	–	–	–	–	1
LIZARD [63]	1	2NFSR	90, 31	1XOR, 1AND	–	–	–	–	1
Plantlet [64]	1	1LFSR+1NFSR	61, 40	1XOR, 1AND	–	–	–	–	1
Fruit-80 [65]	1	1LFSR+1NFSR	43, 37	1XOR, 1AND	–	–	–	–	1
E0 [6]	1	4LFSR	25, 31, 33, 39	1XOR	2^4	–	–	–	1
WG-16 [92]	1	1LFSR	32	16XOR, 16AND	–	GF (2^{16})	–	–	1
SOBER [93]	8	1LFSR	17	8XOR, 8AND	2^8	GF (2^8)	–	–	8
SOBER-t16 [94]	16	1LFSR	17	16XOR, 8XOR	2^{16}	GF (2^{16})	–	8×16	16
SOBER-128 [95]	32	1LFSR	17	32XOR, 24XOR	2^{32}	GF (2^{32})	$\ggg$	8×32	32
SOSEMANUK [53]	32	1LFSR	10	32XOR	2^{32}	GF (2^{32})	$\lll$	4×4	32
Loiss [96]	8	1LFSR	32	32XOR, 8XOR	–	GF (2^8)	$\lll$,$\gg$	8×8	8
Strumok [56]	64	1LFSR	16	64XOR	2^{64}	GF (2^{64})	–	8×8	64
SNOW 2.0 [50]	32	1LFSR	16	32XOR	2^{32}	GF (2^{32})	–	8×8	32
SNOW 3G [3]	32	1LFSR	16	32XOR	2^{32}	GF (2^{32})	–	8×32	32
SNOW-V [51]	128	2LFSR	16, 16	128XOR, 16XOR	2^{32}	GF (2^{16})	–	8×8	128
ZUC [4]	32	1LFSR	16	32XOR	$2^{32}, 2^{31}-1$	–	$\gg$,$\lll$	8×8	32
ZUC-256 [19]	32	1LFSR	16	32XOR, 7OR	$2^{32}, 2^{31}-1$	–	$\gg$,$\lll$	8×8	32
RC4 [5]	8	–	–	–	2^8	–	–	–	8
ChaCha20 [7]	32	–	–	32XOR	2^{32}	–	$\lll$	–	32

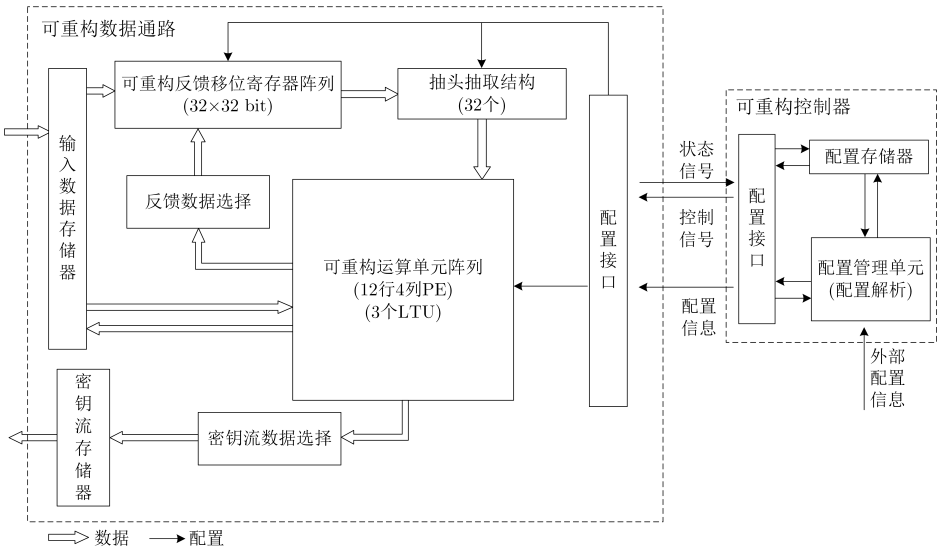


图 8 可重构计算流密码处理架构结构框图

Figure 8 Block diagram of reconfigurable computing stream cipher processing architecture

8 bit、16 bit、32 bit 一共 4 种操作粒度. 可重构反馈移位寄存器阵列的反馈输入为可重构运算单元阵列的输出数据.

抽头抽取结构负责从可重构反馈移位寄存器阵列中抽取数据并送入可重构运算单元阵列进行运算, 共 32 个. 可重构反馈移位寄存器阵列中每个 32 bit 的移位寄存器单元阵列为一组, 共 32 组, 每个抽头抽取结构先从 32 组 32 bit 数据中抽取出一组, 然后根据具体流密码算法的操作粒度, 对抽取出的 32 bit 数据进行再一次抽取选择, 比如操作粒度为 1 bit 时, 用一个 32 选 1 的多路选择器抽取出任意 1 bit, 高位补 0 拼为 32 bit. 通过这种结构实现任意不同位置抽头的抽取.

可重构运算单元阵列是可重构计算流密码处理架构中的核心模块, 主要负责流密码中的数据运算. 其结构框图如图 9 所示.

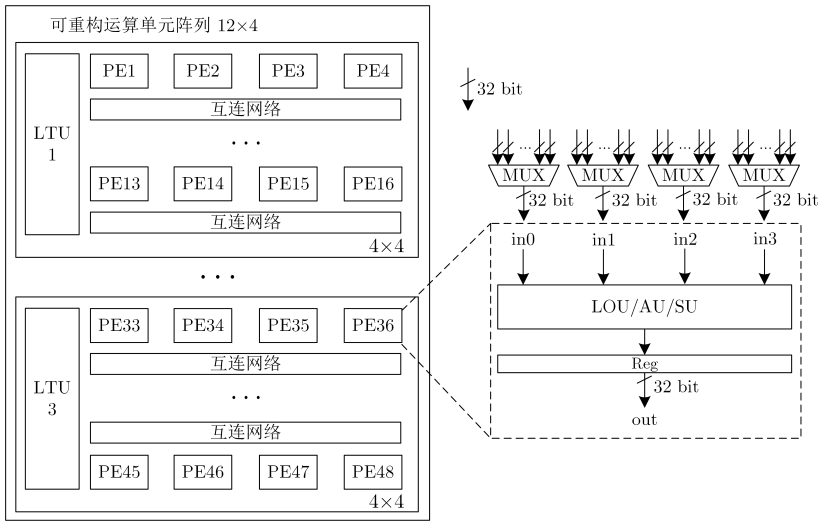


图 9 可重构运算单元阵列结构框图

Figure 9 Structure diagram of reconfigurable operation unit array

由 12 行 4 列 PE 和 PE 行之间的互连网络组成, 根据对流密码计算粒度的统计分析将 PE 的重构粒度设计为 32 bit, 以更好的满足不同操作位宽的流密码. 运算单元的设计包括逻辑运算单元 (logical operation unit, LOU)、算术运算单元 (arithmetic operation unit, AU)、移位单元 (shift unit, SU)、查找表单元 (lookup table unit, LTU) 4 种不同的功能单元. 每个 PE 都包含 LOU、AU、SU, 因为在流密码中 S 盒使用频率较低而且面积较大, 所以每 4 行 PE 作为一个 PE 组共享一个 LTU. LOU 有四个输入, 可以实现四输入中任意两个输入操作数最多三级的任意基本逻辑操作, 包括异或 (XOR)、与 (AND)、或 (OR) 等; AU 有四个输入, 由一个 32 位的加法器组成, 可以实现模 2^8、模 2^{16}、模 2^{32}、模 $2^{31}-1$ 几种基数的加法运算; SU 有三个输入, 由两个 32 位的桶形移位器组成, 可以实现最多 64 bit 输入操作数的 32 bit 以内任意位数的逻辑左移、逻辑右移、循环左移、循环右移; LTU 有三个输入, 由 4 个 8 bit 进 8 bit 出的 S 盒运算单元组成, S 盒运算单元用 SRAM 实现, 支持最大 8 bit 输入, 32 bit 输出的查找表运算. 因为异或逻辑在流密码中使用非常频繁, 所以为了加快计算效率, 在 AU、SU、LTU 的输入和输出添加了异或操作, 从而在进行算术运算、移位或查找表运算之前或之后可以与另一个输入操作数进行异或操作. 每个 PE 有 4 个 32 bit 的输入, 1 个 32 bit 的输出, PE 的输入可以来自于抽头抽取结构的输出数据、与之相邻上一行 4 个 PE 的输出数据、LTU 的输出数据. 另外为了在 PE 之间可以实现流水操作, 在每个 PE 的输出端设置一个 32 bit 的寄存器用于缓存每个 PE 的计算结果, 如果需要流水处理, 就寄存一级, 如果不需要就对寄存器进行旁路. PE 行之间的互连网络采用 CrossBar 方式进行互连, 实现相邻行与行之间的全互连.

该可重构数据通路的缓存结构包括输入数据存储器和密钥流存储器, 均连接外部数据接口, 负责与外部数据的交互. 输入数据存储器用来存储从外部数据接口读取到的要参与运算的数据, 可重构运算单元阵列完成运算后, 如果在流密码的状态更新阶段, 反馈数据选择模块从 PE 输出端选择要反馈数据, 反馈回可重构反馈移位寄存器继续参与状态的更新; 如果在流密码的密钥流生成阶段, 密钥流数据选择模块从 PE 输出端选择密钥流数据存储到密钥流存储器并将其输出到外部接口. 可重构运算单元阵列在进行流密码算法的数据流运算操作时产生的中间结果数据也可以反馈回可重构运算单元阵列进行缓存, 需要时再通过抽头抽取结构抽出.

可重构控制器中配置管理单元接收并解析外部配置信息, 生成内部配置信息和顶层控制信号. 配置存储器用于保存内部配置信息. 配置接口访问配置存储器并将内部配置信息输出到可重构数据通路 [87], 完成对可重构反馈移位寄存器阵列、抽头抽取结构、可重构运算单元阵列的具体配置. 同时, 配置接口输出可重构数据通路的状态信号, 根据状态信号产生控制码, 控制可重构运算单元阵列的资源调度.

对于流密码的数据相关性而言, 从整个算法加解密的层次来看, 流密码算法是串行执行的, 因此不能将任务分解成多个子任务并行执行; 从算法的内部结构来看, 流密码算法各个构建块之间存在读后写相关, 比如两个反馈移位寄存器级联时, 下一级寄存器要先读取到上一级寄存器的值, 并更新本身状态后, 上一级寄存器才能更新自己的状态. 对于流密码的并行性而言, 流密码算法各个构建块之间存在并行性, 像基于 LFSR 的流密码算法的反馈移位寄存器和非线性部分之间是并发执行的, 各个构建块都需要同时执行, 并更新本身的状态, 因此在利用可重构计算密码处理器实现流密码时可以开发其并行性 [87].

4 发展趋势

从算法设计的角度来看, 传统面向比特的流密码已经很难满足当今通信应用的需求, 特别是在软件实现方面, 为了提高处理器软件中密钥流的生成速度, 流密码算法逐渐倾向于面向字设计. 设计结构也逐渐多样化, 越来越多的流密码算法不仅基于某一种结构, 而是将不同结构结合起来进行设计, 以便提高安全性, 比如基于 LFSR 的流密码, 也会在非线性部分利用一些分组密码的组件, 像 S 盒等. 流密码算法设计的关注点主要集中在所采用的密码构建块和基本操作上, 本质上遵循 Shannon 的混淆和扩散范式, 在设计流密码时既要保证能够对现有攻击具有免疫性, 又要保证易于评估其他密码属性, 还要使设计在安全性、性能和资源需求之间达到良好的平衡 [48].

从算法实现的角度来看, 在实现流密码算法时, 要在所需的性能和资源之间进行权衡, 性能主要指功耗、能耗、延迟、吞吐量等方面; 硬件实现所需的资源通常指面积、等效门或逻辑块, 软件实现所需的资

源通常体现在寄存器的数量、RAM 和 ROM 的规模上, 资源需求即成本, 因为增加更多的门或者内存会增加设备的生产成本. 从流密码算法加解密的角度来看, 算法是串行的, 从其内部结构来看, 算法的各个组件之间存在并行性, 利用可重构计算密码处理器实现时可以开发流密码算法的并行性.

从算法发展的角度来看, 基于流密码算法的轻量级认证加密算法的设计、分析与实现是一个热点. 近年来对称密码界将注意力集中在更有效地提供消息加密和认证的专用认证加密方案上 [17]. 另外, 如何在资源受限的设备上使用安全有效的密码算法是一个具有挑战性的问题, 目前关于轻量级密码算法还没有严格的标准, 但轻量级密码算法的常见表现是对目标设备的资源要求很低 [48]. 所以, 为了满足物联网时代广泛应用的资源受限设备, 轻量级密码算法的设计和分析也是一个研究热点. 由 NIST 发起并正在进行的轻量级密码标准化项目征集的目标是带有认证功能的轻量级认证加密算法, 一定程度上反映出, 设计和分析基于流密码的轻量级认证加密算法也是目前及未来非常重要的研究方向.

参考文献

[1] MENEZES A J, VAN OORSCHOT P C, VANSTONE S A. Handbook of applied cryptography[M]. NW Boca Raton, FL, USA: CRC press, 2018: 191–122.

[2] ANDERSON D P, HERRTWICH R G. Internet communication with end-to-end performance guarantees[C]. In: Telekommunikation und multimediale Anwendungen der Informatik. Springer Berlin Heidelberg, 1991: 246–258. [DOI: 10.1007/978-3-642-77060-9_23]

[3] ETSI/SAGE. Specification of the 3GPP confidentiality and integrity algorithms UEA2 & UIA2. Document 2: SNOW 3G specification[EB/OL]. Version 1.1. September 2006. http://www.3gpp.org/ftp/

[4] FENG X T. 3GPP LTE international encryption standard ZUC algorithm[J]. Information Security and Communications Privacy, 2011, 2011(12): 45–46. [DOI: 10.3969/j.issn.1009-8054.2011.12.033]
冯秀涛. 3GPP LTE 国际加密标准 ZUC 算法 [J]. 信息安全与通信保密, 2011, 2011(12): 45–46. [DOI: 10.3969/j.issn.1009-8054.2011.12.033]

[5] RIVEST R. The RC4 encryption algorithm[R]. RSA Data Security Inc. Doc No 20, 1992: 86–96.

[6] Bluetooth S: Specification of Bluetooth System. Version 4.0[S/OL]. http://www.Bluetooth.com, 2020.

[7] BERNSTEIN D J. ChaCha, a variant of Salsa20[C/OL]. In: Workshop Record of SASC. 2008. https://cr.yp.to/chacha/chacha-20080120.pdf

[8] LANGLEY A, CHANG W T, MAVROGIANNOPOULOS N, et al. ChaCha20—Poly1305 Cipher Suites for Transport Layer Security (TLS)[S/OL]. RFC 7905, 2016. https://deva.amsl.com/doc/rfc7905/

[9] ChaCha usage & deployment[EB/OL]. https://ianix.com/pub/chacha-deployment.html, 2020.

[10] STALLINGS W. Cryptography and Network Security[M]. 4th ed. Pearson Education India, 2006: 218–239.

[11] SHANNON C E. Communication theory of secrecy systems[J]. The Bell System Technical Journal, 1949, 28(4): 656–715.

[12] ZHANG B, XU C, FENG D G. Design and analysis of stream ciphers: Past, present and future directions[J]. Journal of Cryptologic Research, 2016, 3(6): 527–545. [DOI: 10.13868/j.cnki.jcr.000149]
张斌, 徐超, 冯登国. 流密码的设计与分析: 回顾、现状与展望 [J]. 密码学报, 2016, 3(6): 527–545. [DOI: 10.13868/j.cnki.jcr.000149]

[13] NESSIE: New European schemes for signatures, integrity, and encryption[EB/OL]. https://cosic.esat.kuleuven.be/nessie/, 2020.

[14] eSTREAM: The ECRYPT stream cipher project[EB/OL]. https://www.ecrypt.eu.org/stream/project.html, 2020.

[15] CRYPTREC: Cryptography Research and Evaluation Committees[EB/OL]. https://www.cryptrec.go.jp/en/about.html, 2020.

[16] CAESAR: Competition for Authenticated encryption: security, applicability, and robustness[EB/OL]. http://competitions.cr.yp.to/caesar.html, , 2020.

[17] WU W L. Research advances on authenticated encryption algorithms[J]. Journal of Cryptologic Research, 2018, 5(1): 70–82. [DOI: 10.13868/j.cnki.jcr.000220]
吴文玲. 认证加密算法研究进展 [J]. 密码学报, 2018, 5(01): 70–82. [DOI: 10.13868/j.cnki.jcr.000220]

[18] Computer Securty Resoource Center (NIST): Lightweight cryptography[EB/OL]. https://csrc.nist.gov/projects/lightweight-cryptography.html, 2020.

[19] DESIGN TEAM. ZUC-256 stream cipher[J]. Journal of Cryptologic Research, 2018, 5(2): 167–179. [DOI: 10.13868/j.cnki.jcr.000228]
ZUC 算法研制组. ZUC-256 流密码算法 [J]. 密码学报, 2018, 5(2): 167–179. [DOI: 10.13868/j.cnki.jcr.000228]

[20] ISO, ISO/IEC 18033-4:2011. Information Technology—Security Techniques—Encryption Algorithms—Part 4: Stream ciphers[S/OL]. 2011. https://www.iso.org/standard/54532.html

[21] ISO, ISO/IEC 29192-3:2012. Information Technology—Security Techniques Lightweight Cryptography—Part 3: Stream ciphers[S/OL]. 2012. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?$csnumber$=56426

[22] ISO, ISO/IEC 29167-13:2015. Information Technology—Automated Identification and Data Capture Techniques—Part 13: Crypto Suite Grain-128A Security Services for Air Interface Communications[S/OL]. 2015. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?$csnumber$=60682

[23] ISO, ISO/IEC 18033-4:2011/AMD 1: 2020. Information Technology—Security Techniques—Encryption Algorithms—Part 4: Stream Ciphers—Amendment 1: ZUC [S/OL]. 2020. https://www.iso.org/standard/77982.html

[24] ARNAULT F, BERGER T, LAURADOUX C, et al. A new approach for FCSRs[C]. In: Selected Areas in Cryptography—SAC 2009. Springer Berlin Heidelberg, 2009: 433–448. [DOI: 10.1007/978-3-642-05445-7_27]

[25] ZHANG W G. Preface on Cryptographic Functions[J]. Journal of Cryptologic Research, 2017, 4(3). 张卫国. “密码函数” 专栏序言 [J]. 密码学报, 2017, 4(3).

[26] CRAMA Y, HAMMER P L. Boolean Functions: Theory, Algorithms, and Applications[M]. Cambridge University Press, 2011: 564–607.

[27] RUEPPEL R A. Stream Ciphers[M]. In: Analysis and Design of Stream Ciphers. Springer Berlin Heidelberg, 1986: 5–16. [DOI: 10.1007/978-3-642-82865-2_2]

[28] YANG Y X, HU Z M. Boolean function counting problem for sequence ciphers[J]. Journal on Communications, 1992, 13(4): 18–24. [DOI: CNKI:SUN:TXXB.0.1992-04-002] 杨义先, 胡正名. 用于序列密码的布尔函数计数问题 [J]. 通信学报, 1992, 13(4): 18–24. [DOI: CNKI:SUN:TXXB.0.1992-04-002]

[29] WOLFRAM S. Cryptography with cellular automata[C]. In: Advances in Cryptology—CRYPTO ’85. Springer Berlin Heidelberg, 1985: 429–432. [DOI: 10.1007/3-540-39799-X_32]

[30] WOLFRAM S. Random sequence generation by cellular automata[J]. Advances in Applied Mathematics, 1986, 7(2): 123–169. [DOI: 10.1016/0196-8858(86)90028-X]

[31] HORTENSIUS P D, MCLEOD R D, CARD H C. Parallel random number generation for VLSI systems using cellular automata[J]. IEEE Computer Architecture Letters, 1989, 38(10): 1466–1473. [DOI: 10.1109/12.35843]

[32] NANDI S, KAR B K, CHAUDHURI P P. Theory and applications of cellular automata in cryptography[J]. IEEE Transactions on Computers, 1994, 43(12): 1346–1357. [DOI: 10.1109/12.338094]

[33] TOMASSINI M, SIPPER M, PERRENOUD M. On the generation of high-quality random numbers by two-dimensional cellular automata[J]. IEEE Transactions on Computers, 2000, 49(10): 1146–1151. [DOI: 10.1109/12.888056]

[34] OISHI S, INOUE H. Pseudo-random number generators and chaos[J]. Transactions of the Institute of Electronics and Communication Engineers of Japan, Section E, 1982, 65E(9): 534–541.

[35] BOESGAARD M, VESTERAGER M, PEDERSEN T, et al. Rabbit: A new high-performance stream cipher[C] In: Fast Software Encryption—FSE 2003. Springer Berlin Heidelberg, 2003: 307–329. [DOI: 10.1007/978-3-540-39887-5_23]

[36] KLIMOV A, SHAMIR A. A new class of invertible mappings[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2002. Springer Berlin Heidelberg, 2002: 470–483. [DOI: 10.1007/3-540-36400-5_34]

[37] KLIMOV A, SHAMIR A. Cryptographic applications of T-functions[C]. In: Selected Areas in Cryptography—SAC 2003. Springer Berlin Heidelberg, 2003: 248–261. [DOI: 10.1007/978-3-540-24654-1_18]

[38] DAEMEN J, CLAPP C. Fast hashing and stream encryption with PANAMA[C]. In: Fast Software Encryption—FSE ’98. Springer Berlin Heidelberg, 1998: 60–74. [DOI: 10.1007/3-540-69710-1_5]

[39] WATANABE D, FURUYA S, YOSHIDA H, et al. A new keystream generator MUGI[C]. In: Fast Software Encryption—FSE 2002. Springer Berlin Heidelberg, 2002: 179–194. [DOI: 10.1007/3-540-45661-9_14]

[40] MÉAUX P, JOURNAULT A, STANDAERT F X, et al. Towards stream ciphers for efficient FHE with low-noise ciphertexts[C]. In: Advances in Cryptology—EUROCRYPT 2016, Part I. Springer Berlin Heidelberg, 2016: 311–343. [DOI: 10.1007/978-3-662-49890-3_13]

[41] JIAO L, HAO Y L, FENG D G. Stream cipher designs: A review[J]. SCIENCE CHINA Information Sciences, 2020, 63(3): 131101. [DOI: 10.1007/s11432-018-9929-x]

[42] BERTONI G, DAEMEN J, PEETERS M, et al. Sponge functions[C/OL]. In: ECRYPT Hash Workshop. Barcelona, Spain. 2007. https://keccak.team/files/SpongeFunctions.pdf

[43] BERTONI G, DAEMEN J, PEETERS M, et al. Cryptographic sponges[J/OL]. http://sponge.noekeon.org, 2011.

[44] BERTONI G, DAEMEN J, PEETERS M, et al. Sponge-based pseudo-random number generators[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2010. Springer Berlin Heidelberg, 2010: 33–47. [DOI: 10.1007/978-3-642-15031-9_3]

[45] MANIFAVAS C, HATZIVASILIS G, FYSARAKIS K, et al. A survey of lightweight stream ciphers for embedded systems[J]. Security and Communication Networks, 2016, 9(10): 1226–1246. [DOI: 10.1002/sec.1399]

[46] WU H J. Cryptanalysis and Design of Stream Ciphers[D]. Leuven: Katholieke Universiteit Leuven. 2008.

[47] GU L Z, ZHENG S H, YANG Y X. Modern Cryptography Tutorial[M]. Beijing: Beijing University of Posts and Telecommunications Press, 2015: 125–154.
谷利泽, 郑世慧, 杨义先. 现代密码学教程 [M]. 北京: 北京邮电大学出版社, 2015: 125–154.

[48] MCKAY K, BASSHAM L, SÖNMEZ TURAN M, et al. Report on lightweight cryptography[R]. National Institute of Standards and Technology, 2016: NISTIR 8114. [DOI: 10.6028/NIST.IR.8114]

[49] DAWSON E, CLARK A, GOLIC J, et al. The LILI-128 keystream generator[C]. In: Proceedings of First Open NESSIE Workshop. Heverlee, Belgium. 2000.
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.1477&rep=rep1&type=pdf

[50] EKDAHL P, JOHANSSON T. A new version of the stream cipher SNOW[C]. In: Selected Areas in Cryptography—SAC 2002. Springer Berlin Heidelberg, 2002: 47–61. [DOI: 10.1007/3-540-36492-7_5]

[51] EKDAHL P, JOHANSSON T, MAXIMOV A, et al. A new SNOW stream cipher called SNOW-V[J]. IACR Transactions on Symmetric Cryptology, 2019, 2019(3): 1–42. [DOI: 10.13154/tosc.v2019.i3.1-42]

[52] BERBAIN C, BILLET O, CANTEAUT A, et al. Decim$^{\nu 2}$[M]. In: New Stream Cipher Designs. Springer Berlin Heidelberg, 2008: 140–151. [DOI: 10.1007/978-3-540-68351-3_11]

[53] BERBAIN C, BILLET O, CANTEAUT A, et al. Sosemanuk, a fast software-oriented stream cipher[M]. In: New Stream Cipher Designs. Springer Berlin Heidelberg, 2008: 98–118. [DOI: 10.1007/978-3-540-68351-3_9]

[54] BABBAGE S, DODD M. The MICKEY stream ciphers[M]. In: New Stream Cipher Designs. Springer Berlin Heidelberg, 2008: 191–209. [DOI: 10.1007/978-3-540-68351-3_15]

[55] WU H. ACORN: A lightweight authenticated cipher (v3)[EB/OL]. CAESAR Second Round Submission, 2016. http://competitions.cr.yp.to/round3/acornv3.pdf

[56] KUZNETSOV O, LUTSENKO M, IVANENKO D. Strumok stream cipher: Specification and basic properties[C]. In: Proceedings of 2016 Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S & T). IEEE, 2016: 59–62. [DOI: 10.1109/INFOCOMMST.2016.7905335]

[57] DE CANNIERE C, PRENEEL B. Trivium[M]. In: New Stream Cipher Designs. Springer Berlin Heidelberg, 2008: 244–266. [DOI: 10.1007/978-3-540-68351-3_18]

[58] HELL M, JOHANSSON T, MEIER W. Grain: A stream cipher for constrained environments[J]. International Journal of Wireless and Mobile Computing, 2006, 2(1): 86–93. [DOI: 10.1504/ijwmc.2007.013798]

[59] AGREN M, HELL M, JOHANSSON T, et al. Grain-128a: A new version of Grain-128 with optional authentication[J]. International Journal of Wireless and Mobile Computing, 2011, 5(1): 48–59. [DOI: 10.1504/ijwmc.2011.044106]

[60] HELL M, JOHANSSON T, MEIER W, et al. An AEAD variant of the grain stream cipher[C]. In: Codes, Cryptology and Information Security—C2SI 2019. Springer Cham, 2019: 55–71. [DOI: 10.1007/978-3-030-16458-4_5]

[61] DUBROVA E, HELL M. Espresso: A stream cipher for 5G wireless communication systems[J]. Cryptography and Communications, 2017, 9(2): 273–289. [DOI: 10.1007/s12095-015-0173-2]

[62] ARMKNECHT F, MIKHALEV V. On lightweight stream ciphers with shorter internal states[C]. In: Fast Software Encryption—FSE 2015. Springer Berlin Heidelberg, 2015: 451–470. [DOI: 10.1007/978-3-662-48116-5_22]

[63] HAMANN M, KRAUSE M, MEIER W. Lizard—A lightweight stream cipher for power-constrained devices[J]. IACR Transactions on Symmetric Cryptology, 2017, 2017(1): 45–79. [DOI: 10.13154/tosc.v2017.i1.45-79]

[64] MIKHALEV V, ARMKNECHT F, MÜLLER C. On ciphers that continuously access the non-volatile key[J]. IACR Transactions on Symmetric Cryptology, 2016, 2016(2): 52–79. [DOI: 10.13154/tosc.v2016.i2.52-79]

[65] AMIN GHAFARI V, HU H. Fruit-80: A secure ultra-lightweight stream cipher for constrained environments[J]. Entropy, 2018, 20(3): 180. [DOI: 10.3390/e20030180]

[66] ROGAWAY P, COPPERSMITH D. A software-optimized encryption algorithm[C]. In: Fast Software Encryption—FSE '93. Springer Berlin Heidelberg, 1993: 56–63. [DOI: 10.1007/3-540-58108-1_8]

[67] WU H. The stream cipher HC-128[M]. In: New Stream Cipher Designs. Springer Berlin Heidelberg, 2008: 39–47. [DOI: 10.1007/978-3-540-68351-3_4]

[68] BIHAM E, SEBERRY J. Py (Roo): A fast and secure stream cipher using rolling arrays[R/OL]. Centre for Computer Security Research, SITACS, University of Wollongong, Australia. 2005.

https://documents.uow.edu.au/ jennie/WEB/WEB05/py-submission.pdf

[69] BIHAM E, SEBERRY J. Tweaking the IV setup of the Py family of stream ciphers—The ciphers TPy, TPypy, and TPy6[EB/OL]. 2007. https://www.cosic.esat.kuleuven.be/ecrypt/stream/papersdir/2007/038.ps

[70] RIVEST R L, SCHULDT J C N. Spritz—A spongy RC4-like stream cipher and hash function[J/OL]. IACR Cryptology ePrint Achieve, 2016: 2016/856. https://eprint.iacr.org/2016/856.pdf

[71] 3rd generation partnership project, technical specification group services and system aspects, 3G security, specification of the 3GPP confidentiality and integrity algorithms; Document 2: KASUMI algorithm specification[EB/OL]. 3GPP TS 35.202. V3.1.1, 2001. https://www.3gpp.org/specifications/60-confidentiality-algorithms

[72] HALEVI S, COPPERSMITH D, JUTLA C. Scream: A software-efficient stream cipher[C]. In: Fast Software Encryption—FSE 2002. Springer Berlin Heidelberg, 2002: 195–209. [DOI: 10.1007/3-540-45661-9_15]

[73] FERGUSON N, WHITING D, SCHNEIER B, et al. Helix: Fast encryption and authentication in a single cryptographic primitive[C]. In: Fast Software Encryption—FSE 2003. Springer Berlin Heidelberg, 2003: 330–346. [DOI: 10.1007/978-3-540-39887-5_24]

[74] CHEN K, HENRICKSEN M, MILLAN W, et al. Dragon: A fast word based stream cipher[C]. In: Information Security and Cryptology—ICISC 2004. Springer Berlin Heidelberg, 2004: 33–50. [DOI: 10.1007/11496618_5]

[75] WHITING D, SCHNEIER B, LUCKS S, et al. Fast encryption and authentication in a single cryptographic primitive[J]. ECRYPT stream cipher project report, 2005, 27(200): 5.

[76] BIRYUKOV A. The design of a stream cipher LEX[C]. In: Selected Areas in Cryptography—SAC 2006. Springer Berlin Heidelberg, 2006: 67–75. [DOI: 10.1007/978-3-540-74462-7_6]

[77] Bernstein D J. The Salsa20 Family of Stream Ciphers[M]. In: New Stream Cipher Designs. Springer Berlin Heidelberg, 2008: 84–97. [DOI: 10.1007/978-3-540-68351-3_8]

[78] WU H J, HUANG T, 2016. The authenticated cipher MORUS (v2) [EB/OL]. https://competitions.cr.yp.to/round3/morusv2.pdf, 2020.

[79] ANDERSON R, MANIFAVAS C. Chameleon—A new kind of stream cipher[C]. In: Fast Software Encryption—FSE '97. Springer Berlin Heidelberg, 1997: 107–113. [DOI: 10.1007/BFb0052339]

[80] WATANABE D, KANEKO T. A construction of light weight Panama-like keystream generator[J]. IEICE Technical Report, 2007, 107(209): 33–40.

[81] WATANABE D, OWADA T, OKAMOTO K, et al. Update on enocoro stream cipher[C]. In: Proceedings of 2010 International Symposium on Information Theory & Its Applications. IEEE, 2010: 778–783. [DOI: 10.1109/ISITA.2010.5649627]

[82] SARKAR P. Hiji-bij-bij: A new stream cipher with a self-synchronizing mode of operation[C]. In: Progress in Cryptology—INDOCRYPT 2003. Springer Berlin Heidelberg, 2003: 36–51. [DOI: 10.1007/978-3-540-24582-7_3]

[83] DAS S, ROYCHOWDHURY D. CAR30: A new scalable stream cipher with rule 30[J]. Cryptography and Communications, 2013, 5(2): 137–162. [DOI: 10.1007/s12095-012-0079-1]

[84] ARNAULT F, BERGER T, LAURADOUX C. F-FCSR stream ciphers[M]. In: New Stream Cipher Designs. Springer Berlin Heidelberg, 2008: 170–178. [DOI: 10.1007/978-3-540-68351-3_13]

[85] MORAWIECKI P, GAJ K, HOMSIRIKAMOL E, et al. ICEPOLE: High-speed, hardware-oriented authenticated encryption[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2014. Springer Berlin Heidelberg, 2014: 392–413. [DOI: 10.1007/978-3-662-44709-3_22]

[86] MÉAUX P, JOURNAULT A, STANDAERT F X, et al. Towards stream ciphers for efficient FHE with low-noise ciphertexts[C]. In: Advances in Cryptology—EUROCRYPT 2016, Part I. Springer Berlin Heidelberg, 2016: 311–343. [DOI: 10.1007/978-3-662-49890-3_13]

[87] WEI S J, LIU LB, YIN S Y. Key techniques of reconfigurable computing processor[J]. SCIENTIA SINICA Informationis, 2012, 42(12): 1559–1576. [DOI: 10.1360/112012-505]
魏少军, 刘雷波, 尹首一. 可重构计算处理器技术 [J]. 中国科学: 信息科学, 2012, 42(12): 1559–1576. [DOI: 10.1360/112012-505]

[88] LIU L B, WANG B, WEI S J. Reconfigurable Computing Cryptographic Processor[M]. Beijing: Science Press, 2018: 2–66.
刘雷波, 王博, 魏少军. 可重构计算密码处理器 [M]. 北京: 科学出版社, 2018: 2–66.

[89] HELL M, JOHANSSON T, MAXIMOV A, et al. A stream cipher proposal: Grain-128[C]. In: 2006 IEEE International Symposium on Information Theory. IEEE, 2006: 1614–1618. [DOI: 10.1109/ISIT.2006.261549]

[90] BABBAGE S, DODD M. The stream cipher MICKEY-128 2.0[EB/OL]. ECRYPT stream cipher project, 2006. https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=DD6D08BEBB5F0806DCB4D5EAE116F9D5?doi=10.1.1.379.1196&rep=rep1&type=pdf

[91] DAVID M, RANASINGHE D C, LARSEN T. A2U2: A stream cipher for printed electronics RFID tags[C]. In:

Proceedings of 2011 IEEE International Conference on RFID. IEEE, 2011: 176–183. [DOI: 10. 1109/RFID. 2011.5764619]

[92] FAN X X, GONG G. Specification of the stream cipher WG-16 based confidentiality and integrity algorithms[R/OL]. University of Waterloo, Waterloo, ON, Canada, Technical Report CACR, 2013. https://cacr.uwaterloo.ca/techreports/2013/cacr2013-06.pdf

[93] ROSE G. A stream cipher based on linear feedback over GF(2^8)[C]. In: Information Security and Privacy—ACISP 1998. Springer Berlin Heidelberg, 1998: 135–146. [DOI: 10.1007/BFb0053728]

[94] HAWKES P. Primitive specification and supporting documentation for SOBER-t16 submission to NESSIE[C]. In: Proceedings of First Open NESSIE Workshop. Heverlee, Belgium, 2000. https://www.cosic.esat.kuleuven.be/nessie/workshop/

[95] HAWKES P, ROSE G G. Primitive specification for SOBER-128[J/OL]. IACR Cryptology ePrint Archieve, 2003: 2003/081. https://eprint.iacr.org/2003/081.pdf

[96] FENG D G, FENG X T, ZHANG W T, et al. Loiss: A byte-oriented stream cipher[C]. In: Coding and Cryptology—IWCC 2011. Springer Berlin Heidelberg, 2011: 109–125. [DOI: 10.1007/978-3-642-20901-7_7]

作者信息



赵石磊 (1979–), 黑龙江肇源人, 博士, 副教授, 硕士生导师. 主要研究领域为信息安全、可重构计算、集成电路设计和 VLSI 数字信号处理.
zhaosl@hrbust.edu.cn



刘玲 (1996–), 山东济南人, 硕士生在读. 主要研究领域为流密码算法、可重构计算.
1325119945@qq.com



黄海 (1982–), 内蒙古巴彦淖尔人, 博士, 教授, 博士生导师, CCF 高级会员. 主要研究领域为信息安全、可重构计算和集成电路设计.
ic@hrbust.edu.cn



徐江 (1997–), 山东日照人, 硕士生在读. 主要研究领域为信息安全、可重构计算.
x_jiang001@qq.com



刘志伟 (1987–), 黑龙江哈尔滨人, 博士, 讲师. 主要研究领域为可重构计算、高速密码算法、并行加密技术和密码芯片的安全设计等.
zwliu@hrbust.edu.cn



于斌 (1984–), 黑龙江饶河人, 硕士, 讲师. 主要研究领域为密码算法、密码芯片设计和数字集成电路设计等.
yubin@hrbust.edu.cn