卫星与水下设备之间信息安全传输的加密算法分析

李成杰[1,2]，崔丽琪[1]，应惟龙[1]，朱立东[2]，郑　艺[1]，李明秀[1]
（1. 西南民族大学，四川 成都 610000；
2. 电子科技大学，四川 成都 610000）

摘　要：为保障卫星与水下设备之间的数据安全传输，对 Ascon、Speck、ChaCha20-Poly1305、ECC_AES 和 RSA_AES 5 种加密方案进行比较。考虑到目标节点的计算和存储资源有限，使用通信指令与导航信息两类数据集，在通用计算平台上评估各方案的加解密时间、CPU 使用率、内存占用量和吞吐量。现有测试结果显示，Speck 在多数测试条件下具有较短的加密时间和较高的加密吞吐量，但各方案在解密时间和内存占用量等指标上的表现存在差异。本研究可为资源受限场景的方案选择提供初步参考，实际部署仍需结合认证能力、具体实现条件和目标硬件进行验证。

关键词：加密机制；轻量级加密算法；Speck 算法
中图分类号：TP393
文献标志码：A
doi：10.11959/j.issn.2096–8930.tdyt2025065

Analysis of encryption algorithms during secure information transmission between satellites and underwater devices

Li Chengjie[1,2], Cui Liqi[1], Ying Weilong[1], Zhu Lidong[2], Zheng Yi[1], Li Mingxiu[1]
1. Southwest Minzu University, Chengdu 610000, China
2. University of Electronic Science and Technology of China, Chengdu 610000, China

Abstract: To support secure data transmission between satellites and underwater devices, five encryption schemes, namely Ascon, Speck, ChaCha20-Poly1305, ECC_AES, and RSA_AES, were compared. Considering the limited computing and storage resources of the target nodes, communication-command and navigation datasets were used to evaluate encryption and decryption time, CPU utilization, memory usage, and throughput on a general-purpose computing platform. The reported results indicate that Speck achieves shorter encryption times and higher encryption throughput under most tested conditions, while the schemes differ in decryption time and memory usage. These findings provide a preliminary reference for scheme selection in resource-constrained environments. Practical deployment requires further validation of authentication capabilities, implementation details, and performance on the target hardware.
Keywords: encryption mechanisms, lightweight encryption algorithms, Speck algorithm

0　引言

在现代信息化社会，信息的安全传输已成为全球网络技术发展的重要课题。随着无线通信技术的迅猛发展，无线异构网络作为一种多样化的通信体系结构，逐渐在各种应用场景中发挥关键作用，其应用场景涵盖卫星通

收稿日期：2025–12–02；**修回日期：**2026–07–19
基金项目：国家自然科学基金资助项目（No. 62371098）；四川省科技计划项目（No. 2024ZHCG0194，2026YFHZ0198）；四川省中央引导地方科技发展专项（No. 2024ZYD0110）；中央高校基本科研业务费专项资金项目（No. ZYN2026126，ZYN2025215）；西南民族大学青藏高原研究科技创新团队项目（No. 2024CXTD09）；四川警察学院智慧警务与国家安全风险管理重点实验室开放课题（No. ZHKFYB2401）
Foundation Items: The National Natural Science Foundation of China (No. 62371098), The Sichuan Science and Technology Program (No. 2024ZHCG0194, 2026YFHZ0198), The Sichuan Central Government Guiding Local Science and Technology Development Special Project (No. 2024ZYD0110), The Special Fund Project for Basic Scientific Research Operating Expenses of Central Universities (No. ZYN2026126, ZYN2025215), The Scientific and Technological Innovation Team for Qinghai-Tibetan Plateau Research in Southwest Minzu University (No. 2024CXTD09), The Open Project of Key Laboratory of Intelligent Policing and National Security Risk Management of Sichuan Police College (No. ZHKFYB2401)

信、地面无线网络、海洋浮标以及水下通信等多个领域，体现出信息传输的广泛应用和复杂性。然而，这些无线异构网络在提供高效通信服务的同时，也面临着严峻的安全挑战。

卫星与水下设备之间通过激光通信链路进行数据传输，这在军事侦察、环境监测和科学研究等领域具有重要意义。由于这类通信涉及大量敏感信息，因此必须采取有效的安全措施防止未经授权的访问和篡改[1]。然而，水下物联网设备面临严峻的资源约束，包括有限的电池寿命、内存容量和实时性要求，这些限制使得传统加密算法难以适用，因为传统加密算法通常计算复杂度高，在资源受限环境下效率显著降低。在此背景下，轻量级加密算法应运而生。这类算法旨在降低计算与存储开销。其中，认证加密方案可同时提供机密性与完整性保护，而单独的分组加密算法还需配合认证机制。随着物联网技术在军事和环境监测等领域的深入应用，卫星与水下设备间的数据安全传输已成为研究热点。当前，该领域的研究主要沿着两条技术路线发展：一是算法本身的优化改进；二是不同加密技术的异构融合。深入探讨轻量级加密算法在资源受限环境中的应用现状和发展趋势，对提升卫星与水下设备间激光通信的安全性和系统性能具有重要价值。

在相关应用与安全研究中，向鹏等[2]讨论了天地互联网的智能遥感卫星，Xie 等[3]研究了水下攻防对抗体系及其未来发展方向，马多贺等[4]分析了卫星互联网的安全风险与防护。

在轻量级密码算法研究方面，研究者通过结构设计和计算流程优化降低资源开销。Adnan 等[5]分析了 AAβ（AA-Beta）轻量级非对称加密方案在嵌入式 Linux 平台上的运行时间。Oualha 等[6]研究了面向物联网的轻量级属性基加密方案。Hosseinzadeh 等[7]在 AVR 系列精简指令集微控制器上评估了轻量级分组密码的能耗与存储开销，结果表明 Speck(64,96)在所评估算法中具有较好的能耗表现。Usman 等[8]提出了面向安全物联网（secure IoT, SIT）的轻量级加密算法，采用 Feistel 结构与替代—置换（substitution-permutation, SP）网络相结合的设计，其分组长度和密钥长度均为 64 bit。上述研究为资源受限设备的密码方案选择提供了参考，但具体性能仍取决于算法参数、实现方式和运行平台。

在物联网安全研究方面，Gupta 等[9]讨论了智慧农业中的安全与隐私挑战，Andrea 等[10]分析了物联网的安全漏洞与挑战，Buchanan 等[11]综述了轻量级密码方法。

现有轻量级密码研究为资源受限设备提供了多种选择[12-13]。面向卫星与水下设备之间的通信，仍需结合设备资源和链路条件评估加密开销。本文选择 5 种具有不同结构和功能的方案：Ascon 为基于置换的轻量级认证加密方案，Speck 为轻量级分组密码，ChaCha20-Poly1305 为流密码与消息认证码相结合的认证加密方案，ECC_AES 和 RSA_AES 为非对称密码与对称密码相结合的混合方案。通过两类数据集在通用平台上的测试，比较各方案的时间与资源开销，为后续目标平台验证提供依据。

通过选择适当的加密算法和安全机制，能够显著提高网络系统的安全性，同时优化通信系统的性能，轻量级加密算法不仅可以提升数据传输的安全性，还能够在保证加密强度的同时降低对计算资源的消耗。此外，对不同加密算法的性能进行比较与优化[14]，有助于在实际应用中实现高效、可靠的通信系统。因此，深入研究无线异构网络的信息安全传输技术，不仅能够应对当前的安全威胁，还能够推动相关技术的发展与应用。

针对具体应用，已有研究分别探讨了轻量级加密算法在物联网中的应用[15]、物联网系统中的网络编码混合加密方案[16]，以及面向无线传感网络安全的轻量级加密算法[17]。

针对卫星与水下设备节点之间的数据传输场景，开放链路存在被窃听和被篡改的风险，需要采用加密与认证机制。卫星和水下设备节点的计算能力、处理能力和存储空间有限，部署安全机制时需控制计算负载。本文在通用平台上比较 5 种加密方案的性能，以评估其时间和资源开销。测试结果可为后续方案选择提供参考，但不能直接替代资源受限设备上的性能测试和安全评估。

1　加密算法研究

1.1　异构激光链路与数据集

1.1.1　异构激光链路

跨介质激光通信是指激光在卫星、太空或大气层与水下设备之间进行传输，涉及不同介质的切换，激光从空气进入水中时会发生折射、散射和衰减，尤其是水对蓝光波段吸收较小，因此常采用蓝绿激光进行水下通信。卫星端需配备高功率激光器，水下端需配备敏感接收器以应对信号损失。虽然激光链路具有高带宽、低时延、强指向性等特点，但受介质界面反射和湍流影响，信号可能不稳定。

激光通信链路的数据率较高，且易受传播环境影响，对加密处理的实时性提出了要求。因此，需控制加密算法的计算开销，并结合具体设备资源与链路条件选择适用方案。

1.1.2 数据集

卫星通过激光通信链路向水下设备发送的数据包括控制指令、传感器数据和状态信息，卫星对水下设备发送的指令数据集见表1。导航数据用于支持水下设备的航行、定位和避障，包含位置、姿态、航向和速度等参数，卫星对水下设备发送的导航数据集示例见表2。

表1 卫星对水下设备发送的指令数据集

序号	指令样例
1	Collect underwater environmental data and report
2	Send the latest mission updates and waypoint list
3	Perform submarine system inspection
4	Request communication link check
5	Send the latest commander orders and strategies
6	Perform submarine weapon system test
7	Adjust submarine depth to 150 meters, speed to 5 knots
8	Stop diving and surface, confirm position and report coordinates (longitude, latitude)
9	Change submarine course to southeast direction, heading angle to 135 degrees
10	Check and report current position coordinates and ocean depth information
11	Enter cruise mode, maintain constant speed of 8 knots, heading northwest direction
12	Stop all power systems and remain stationary, record surrounding environmental sounds
13	Prepare for underwater target search, search radius is 500 meters
…	…
100	Send the latest ocean weather report, including ocean temperature, salinity, and meteorological conditions

1.2 Ascon算法分析

已有研究探讨了基于Ascon加密算法的信息安全保护模型[18]。Ascon是一类基于置换的轻量级密码算法，其认证加密功能可同时保护数据机密性和完整性。Ascon在CAESAR竞赛的轻量级应用类别中胜出，并于2023年被美国国家标准与技术研究院（national institute of standards and technology, NIST）选为轻量级密码标准化的基础算法[13]。Ascon通过初始化、关联数据处理、明文处理和最终化等阶段生成密文与认证标签，其侧信道防护能力仍取决于具体实现。不同版本的参数和编码规则存在差异，实验复现时应明确所用版本。

Ascon首先利用密钥K、nonce N和初始化向量（initialization vector, IV）构造内部状态S，并执行初始化置换和密钥注入。随后，按所用版本规定的填充和分块规则处理关联数据A，并逐块加密明文P。最终阶段再次注入密钥，生成认证标签T。解密端使用相同参数处理关联数据和密文，重新计算标签；只有标签验证通过时才输出明文，否则返回失败。Ascon加解密与验证流程见算法1。

为测试算法性能，分别使用原始数据集的25%、50%、75%和100%进行10轮重复实验，评估不同数据规模下的加解密时间、加解密CPU使用率、加解密内存占用量以及加解密吞吐量。该实验运行于Windows操作系统，处理器为13th Gen Intel(R) Core(TM) i5-13450HX（2.40 GHz），Python版本为3.10，内存为16.0 GB。测试流程包括数据准备、参数生成、加解密测量、结果验证和指标记录。吞吐量定义为实际处理的明文数据字节数与相应操作耗时之比，单位为$B\cdot s^{-1}$。性能测试流程见算法2。

利用Ascon算法对包含100条通信指令的数据集进行

表2 卫星对水下设备发送的导航数据集示例

序号	纬度	经度	深度	侧倾角	俯仰角	航向角度	水速	地速	…
1	50.333 973°	−7.711 981°	−0.421 316	0.063 879°	−0.016 162°	0.083 112	0.075 928	0.075 928	
2	50.331 646°	−7.712 076°	−0.421 902	0.054 192°	0.000 593°	6.277 174	0.064 195	0.064 195	
3	50.329 678°	−7.712 089°	−0.422 19	0.031 904°	0.003 089°	6.261 85	0.059 211	0.059 211	
4	50.325 058°	−7.711 945°	−0.460 442	−0.001 047°	0.002 426°	0.054 803	0.044 102	0.044 102	
5	50.324 039°	−7.711 943°	−0.460 442	−0.015 481°	−0.011 1°	0.051 086	0.048 052	0.048 052	
6	50.322 83°	−7.711 95°	−0.460 99	0.037 542°	0.006 44°	0.067 282	0.043 417	0.043 417	
7	50.322 838°	−7.711 95°	−0.460 99	0.057 386°	−0.012 69°	0.070 319	0.043 417	0.043 417	
8	50.321 754°	−7.711 95°	−0.423 85	0.013 98°	−0.006 63°	0.066 322	0.037 054	0.037 054	
9	50.311 069°	−7.712 69°	−0.463 28	−0.025 34°	−0.008 74°	4.329 429	0.008 246	0.008 246	
10	50.315 048°	−7.712 25°	−0.499 69	−0.014 02°	−0.005 95°	0.236 631	0.015	0.015	
...									
5 000	50.397 449°	−7.721 78°	−0.226 28	−0.000 75°	−0.023 41°	2.520 664	0.095 755	0.095 755	

算法 1　Ascon 加解密与验证流程

认证加密 $E_{k,r,a,b}(K,N,A,P)$	验证解密 $D_{k,r,a,b}(K,N,A,C,T)$
输入: 密钥 $K \in \{0,1\}^k$（密钥长度由所用版本确定） nonce $N \in \{0,1\}^{128}$ 关联数据 $A \in \{0,1\}^*$ 明文 $P \in \{0,1\}^*$ 输出: 密文 $C \in \{0,1\}^{\lvert P\rvert}$ 标签 $T \in \{0,1\}^{128}$	输入: 密钥 $K \in \{0,1\}^k$（密钥长度由所用版本确定） nonce $N \in \{0,1\}^{128}$ 关联数据 $A \in \{0,1\}^*$ 密文 $C \in \{0,1\}^*$ 标签 $T \in \{0,1\}^{128}$ 输出: 明文 $P \in \{0,1\}^{\lvert C\rvert}$ 或 $\perp$
Ascon 加密 Ascon_Encrypt(K, N, A, P): 初始化 $S \leftarrow \mathrm{IV}_{k,r,a,b} \| K \| N$ $S \leftarrow p^a(S) \oplus (0^{320-k} \| K)$ 处理关联数据 若 $\lvert A\rvert > 0$: 依次处理 A 中的分组 A_i: $S \leftarrow S \oplus A_i$ $S \leftarrow \mathrm{Permutation}(S)$ $S \leftarrow S \oplus \mathrm{Padding}$ 加密明文 $C \leftarrow [\,]$ 依次处理 P 中的分组 P_i: $C_i \leftarrow P_i \oplus \mathrm{Extract}(S)$ $C \leftarrow C \| C_i$ $S \leftarrow \mathrm{Inject}(C_i)$ $S \leftarrow \mathrm{Permutation}(S)$ 最终化 $S \leftarrow \mathrm{Inject}(K)$ $T \leftarrow \mathrm{Finalize}(S)$ 返回 C,T	Ascon 解密 Ascon_Decrypt(K, N, A, C, T): 初始化 $S \leftarrow \mathrm{IV}_{k,r,a,b} \| K \| N$ $S \leftarrow p^a(S) \oplus (0^{320-k} \| K)$ 处理关联数据 若 $\lvert A\rvert > 0$: 依次处理 A 中的分组 A_i: $S \leftarrow S \oplus A_i$ $S \leftarrow \mathrm{Permutation}(S)$ $S \leftarrow S \oplus \mathrm{Padding}$ 解密密文 $P \leftarrow [\,]$ 依次处理 C 中的分组 C_i: $P_i \leftarrow C_i \oplus \mathrm{Extract}(S)$ $P \leftarrow P \| P_i$ $S \leftarrow \mathrm{Inject}(C_i)$ $S \leftarrow \mathrm{Permutation}(S)$ 最终化 $S \leftarrow \mathrm{Inject}(K)$ $T' \leftarrow \mathrm{Finalize}(S)$ 若 $T' = T$（以常数时间比较）: 返回 P 否则: 返回 $\perp$（标签验证失败）

算法 2　加密方案性能测试流程

性能测试
初始化阶段
输入： 原始数据集 D 测试比例 $P = [0.25,0.50,0.75,1.0]$ 重复次数 $N = 10$
输出： 包含性能指标的工作簿 W 创建工作簿 W 及工作表 S 设置表头 H=[比例, 加密时间, 解密时间, 加密CPU, 解密CPU, 加密内存, 解密内存, 加密吞吐量, 解密吞吐量] 将 H 写入 $S[0,:]$
将 D 按统一编码转为字节数组：$B \leftarrow$bytearray(D)
测试过程 对每个 $p \in P$ 执行： 计算测试字节数 $s \leftarrow \lfloor \lvert B\rvert \times p \rfloor$ 截取测试数据 $B_p \leftarrow B[0{:}s]$ 令 $i \leftarrow 1$ 至 N，循环执行： 生成当前方案所需参数 key ← KeyGen(当前方案参数) 按当前方案生成合法的nonce及其他参数 测量加密过程 $(C,t_e,u_e,m_e) \leftarrow$ MonitorResources (Encrypt(B_p,key,nonce)) $q_e \leftarrow s/t_e$（单位：B/s） 测量解密过程 使用本轮加密得到的密文C $(P',t_d,u_d,m_d) \leftarrow$ MonitorResources (Decrypt(C,key,nonce))；验证 $P' = B_p$ $q_d \leftarrow s/t_d$（单位：B/s） 记录指标 将 $p,t_e,t_d,u_e,u_d,m_e,m_d,q_e,q_d$ 写入工作表 S； 统一计时范围、资源统计口径和汇总方法 最终化 将工作簿 W 保存为“crypto_benchmark.xlsx” 返回各项指标的统计汇总

加解密测试，针对不同长度的指令进行验证，分析Ascon算法的加解密时间、加解密CPU使用率、加解密内存占用量，以及不同样本比例下的加解密吞吐量，Ascon算法在通信数据集下的加解密分析结果如图1所示。同时，对包含5 000条导航位置信息的数据集进行加解密测试，收集上述指标信息，Ascon算法在导航数据集下的加解密分析结果如图2所示。

1.3　Speck算法分析

Speck算法是一种专为轻量级应用设计的加密算法，由美国国家安全局在2013年提出，该算法是为满足对轻量级加密解决方案的需求而开发的，特别是在计算能力有限的环境下。它适用于各种块大小和密钥长度，具有简单的设计和高效的加解密操作。Speck算法基于变形Feistel结构设计，将数据分为固定长度的块，通过一系列轮函数操作加密数据块，它的轮函数包含4种基本操作：按位异或(⊕)、模加(⊞)、循环左移(<<<)、循环右移(>>>)，通过组合这些操作增加算法破解难度。Speck算法的轮函数可以看作变形的两轮Feistel结构，每一轮中输入的明文块通过轮函数转换为输出的密文块。

Speck通过对固定长度分组反复执行轮函数完成加密，具有结构紧凑、易于软件实现的特点。不同变体使用不同

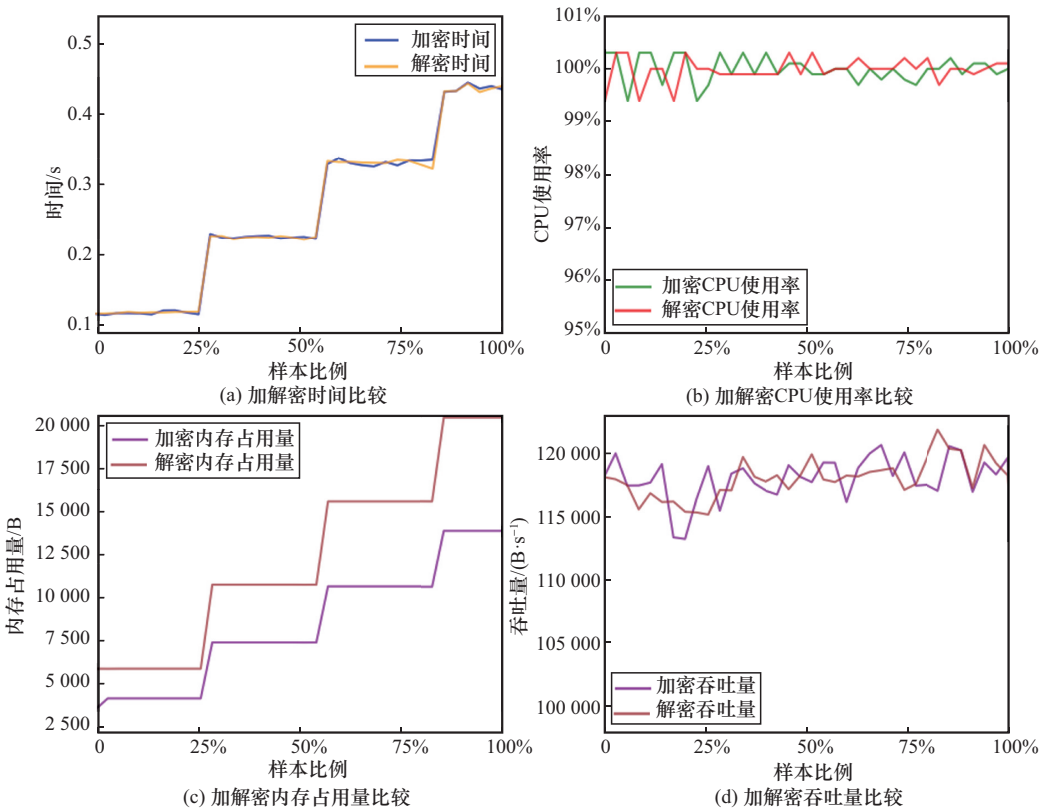


图 1　Ascon 算法在通信数据集下的加解密分析结果

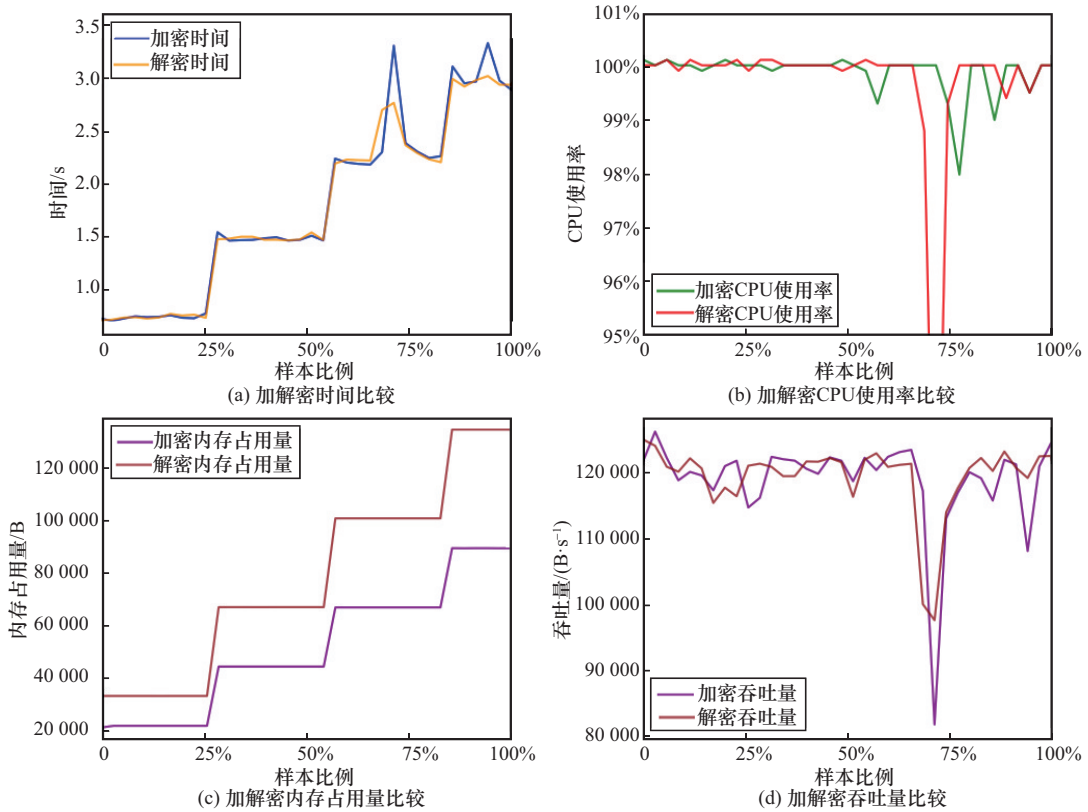


图 2　Ascon 算法在导航数据集下的加解密分析结果

的分组长度、密钥长度和轮数，实际性能与安全性需要结合所选参数及实现方式评估。本文的运行时间比较不能作为算法安全性证明。尤其是，分组加密本身不提供消息认证，实际通信还需配合完整性保护和密钥管理机制。

首先，将一个明文分组拆分为两个等长字，并按顺序使用各轮密钥执行Speck轮函数，最后将两个字拼接为密文分组。解密时，按相反顺序使用轮密钥执行逆轮函数，恢复明文。本文所述电码本（electronic codebook, ECB）模式对各分组独立处理，但会使相同明文分组产生相同密文分组，也不提供完整性保护，因此仅作为所测实现的模式说明。Speck分组加解密流程见算法3。

算法3　Speck分组加解密流程

Speck分组密码	
1. 参数	拼接密文分组
分组长度 $2w$ bit	$C \leftarrow (b \| a)$
$(w \in \{16,24,32,48,64\})$	返回 C
轮数 T	4. 解密过程
旋转常数	函数 Decrypt (C,K_{sch})
$w=16$时，$(\alpha,\beta)=(7,2)$	输入
其余标准字长：$(\alpha,\beta)=(8,3)$	密文 $C \in \{0,1\}^{\{2w\}}$
模数 $M=2^w$	轮密钥序列 $K_{sch}=[k_0,\cdots,k_{\{T-1\}}]$
2. 轮函数SpeckRound (x,y,k)	输出
输入	明文 $P \in \{0,1\}^{\{2w\}}$
x,y:w bit字	拆分密文分组
k: 轮密钥	$(b,a) \leftarrow \text{Split}(C)$
输出:	逆序迭代轮函数
(x',y')变换后的字对	令 $t \leftarrow T-1$ 递减至0，循环执行：
循环右移与模加	$(b,a) \leftarrow \text{InvSpeckRound}(b,a,K_{sch}[t])$
$x \leftarrow \text{ROR}(x,\alpha)$	拼接明文分组
$\bar{x} \leftarrow (\bar{x}+y) \bmod M$	$P \leftarrow (b \| a)$
密钥混合与循环左移	返回 P
$x \leftarrow x \oplus k$	5. 逆轮函数
$y \leftarrow \text{ROL}(y,\beta) \oplus x$	函数 InvSpeckRound(x,y,k)
输出轮运算结果	输入
$(x',y') \leftarrow (x,y)$	x,y:w bit字
返回(x',y')	k: 轮密钥
3. 加密过程	输出:
函数 Encrypt (P,K_{sch})	(x',y')还原后的字对
输入	逆异或与逆旋转
明文 $P \in \{0,1\}^{\{2w\}}$	$y \leftarrow y \oplus x$
轮密钥序列 $K_{sch}=[k_0,\cdots,k_{\{T-1\}}]$	恢复第二个字
输出	$y' \leftarrow \text{ROR}(y,\beta)$
密文 $C \in \{0,1\}^{\{2w\}}$	$x \leftarrow x \oplus k$
拆分明文分组	逆模加运算
$(b,a) \leftarrow \text{Split}(P)$	$x \leftarrow (x-y') \bmod M$
迭代轮函数	$x' \leftarrow \text{ROL}(x,\alpha)$
令 $t \leftarrow 0$ 至 $T-1$，循环执行：	返回 (x',y')
$(b,a) \leftarrow \text{SpeckRound}(b,a,K_{sch}[t])$	

该实验运行在Windows操作系统上，处理器为13th Gen Intel Core i5-13450HX (2.40 GHz)，Python版本为3.10，内存大小为16.0 GB。与算法2所示算法性能测试框架相同，对包含100条通信指令的数据集进行Speck加解密测试，针对不同长度的指令进行验证，分析Speck算法的加解密时间、加解密CPU使用率、加解密内存占用量，以及不同样本比例下的加解密吞吐量，Speck算法在通信数据集下的加解密分析结果如图3所示。同时，对包含5 000条导航位置信息的数据集进行加解密测试，收集上述指标信息，Speck算法在导航数据集下的加解密分析结果如图4所示。

1.4　ChaCha20-Poly1305算法分析

ChaCha20是一种快速流密码，用于生成密钥流，并对数据进行加解密。Poly1305是一个消息认证码（message authentication code, MAC）算法，用于生成和验证保障数据完整性和真实性的认证标签。ChaCha20-Poly1305结合了ChaCha20和Poly1305，为数据提供加密和认证保护。

ChaCha20首先使用固定常数、256 bit密钥、32 bit计数器和96 bit nonce初始化内部状态。随后执行10次双轮运算，每次双轮包含一轮列运算和一轮对角线运算，共20轮。将运算结果与初始状态逐字相加并序列化后得到密钥流，再与输入数据异或，完成加密或解密。末尾不足一个分组的数据仅使用相应长度的密钥流。ChaCha20加解密流程见算法4。

Poly1305利用一次性密钥对消息计算认证标签。在ChaCha20-Poly1305中，计数器为0的ChaCha20块输出的前32 B用于派生该密钥；消息按不超过16 B分块，经过整数编码和模运算后生成128 bit标签。Poly1305认证流程见算法5。

ChaCha20-Poly1305是一种带关联数据的认证加密（authenticated encryption with associated data, AEAD）方案。加密端先用计数器0生成Poly1305一次性密钥，再从计数器1开始生成用于加密明文的ChaCha20密钥流。将关联数据、密文、相应填充以及两者的字节长度按规定格式拼接后，由Poly1305计算认证标签。解密端根据接收的关联数据和密文重新计算标签；仅在验证通过后输出明文。具体流程见算法6。

该实验运行在Windows操作系统上，处理器为13th Gen Intel Core i5-13450HX (2.40 GHz)，Python版本为3.10，内存大小为16.0 GB。与算法2的加密算法性能测试框架相同，利用ChaCha20-Poly1305算法对包含

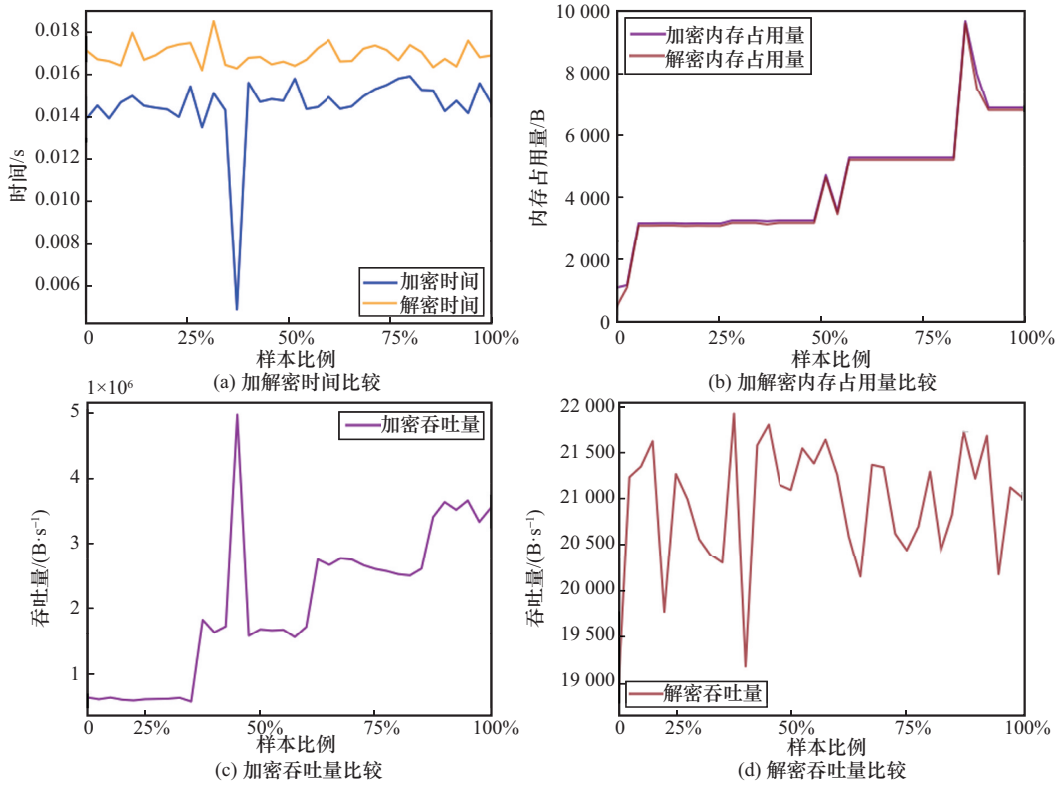


图 3　Speck 算法在通信数据集下的加解密分析结果

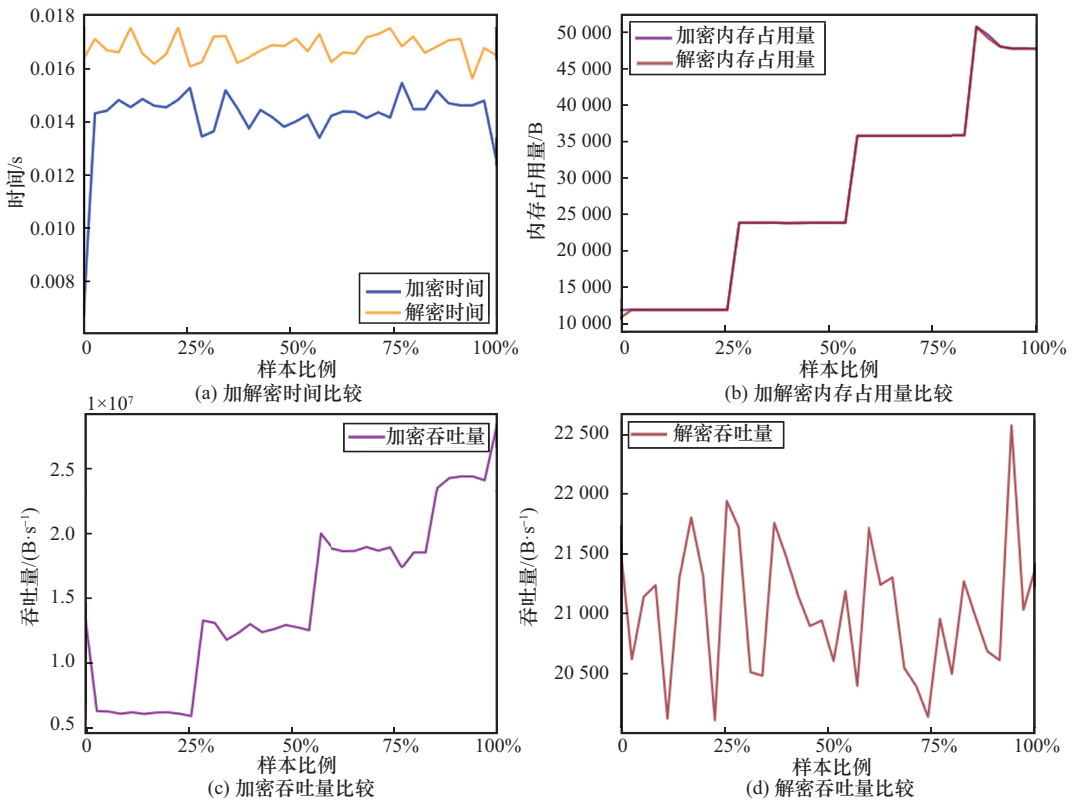


图 4　Speck 算法在导航数据集下的加解密分析结果

算法 4　ChaCha20 加解密流程

ChaCha20 流密码

$C = \text{ChaCha20} - \text{SC}(K,N,P,j)$

输入：$K \in \{0,1\}^{256}, N \in \{0,1\}^{96}, P \in \{0,1\}^{*}$；初始计数器 j

输出：$C \in \{0,1\}^{|P|}$

令 $i \leftarrow 0$ 至 $\lceil |P|/512 \rceil - 1$，循环执行

/* 初始化状态 */

$S[0] \leftarrow$ 0x61707865，$S[1] \leftarrow$ 0x3320646e

$S[2] \leftarrow$ 0x79622d32，$S[3] \leftarrow$ 0x6b206574

$S[4..11] \leftarrow K$　{设置密钥}

$S[12] \leftarrow j + i$　{设置计数器}

$S[13..15] \leftarrow N$　{设置 nonce}

$S' \leftarrow S$　{保存初始状态}

令 $n \leftarrow 0$ 至 9，循环执行 {10 次双轮}

/* 列轮 */

$S[0,4,8,12] \leftarrow \text{QR}(S[0],S[4],S[8],S[12])$

$S[1,5,9,13] \leftarrow \text{QR}(S[1],S[5],S[9],S[13])$

$S[2,6,10,14] \leftarrow \text{QR}(S[2],S[6],S[10],S[14])$

$S[3,7,11,15] \leftarrow \text{QR}(S[3],S[7],S[11],S[15])$

/* 对角线轮 */

$S[0,5,10,15] \leftarrow \text{QR}(S[0],S[5],S[10],S[15])$

$S[1,6,11,12] \leftarrow \text{QR}(S[1],S[6],S[11],S[12])$

$S[2,7,8,13] \leftarrow \text{QR}(S[2],S[7],S[8],S[13])$

$S[3,4,9,14] \leftarrow \text{QR}(S[3],S[4],S[9],S[14])$

结束循环

$k_i^N \leftarrow S \boxplus S' \{\forall 0 \leqslant x \leqslant 15 : S[x] \boxplus S'[x]\}$

$c_i \leftarrow p_i \oplus k_i^N$　{末块仅取所需长度的密钥流}

结束循环

返回 C

算法 5　Poly1305 认证流程

Poly1305-ChaCha20 认证标签计算

$T = \text{Poly1305} - \text{ChaCha}(K,N,M)$.

输入：$K \in \{0,1\}^{256}, N \in \{0,1\}^{96}, M \in \{0,1\}^{8l}$

输出：$T \in \{0,1\}^{128}$

/* ChaCha20 */

$(r,s) \leftarrow \text{Split32}(\text{ChaCha20} - \text{Block}(K,0,N)[0:32])$　{取块输出的前 32 B，拆为 r 和 s}

/* Poly1305 */

$(m_1,...,m_q) \leftarrow \text{Pad1305}(M)$　{将 M 编码为 q 个分组}

$\bar{r} \leftarrow \text{Clamp}(r)$　{按规范清除 r 的指定比特}

$h \leftarrow 0$

令 $i \leftarrow 1$ 至 q，循环执行　{计算多项式}

$h \leftarrow ((h + m_i) \cdot r) \bmod (2^{130} - 5)$

结束循环

$T \leftarrow (h + s) \bmod 2^{128}$　{生成标签}

返回 T

算法 6　ChaCha20-Poly1305 认证加密流程

ChaCha20-Poly1305 认证加密

输入：　$K \in \{0,1\}^{256}, N \in \{0,1\}^{96}, P \in \{0,1\}^{*}, \text{AD} \in \{0,1\}^{*}$

输出：　$C \in \{0,1\}^{|P|}, T \in \{0,1\}^{128}$

$C \leftarrow \text{ChaCha20} - \text{SC}(K,N,P,1)$　{数据加密从计数器 1 开始}

$M \leftarrow \text{Pad}(\text{AD}) \| \text{Pad}(C) \| L(\text{AD}) \| L(C)$　{填充消息}

$T \leftarrow \text{Poly1305} - \text{ChaCha20}(K,N,M)$　{计算认证标签}

返回 (C,T)

100 条通信指令的数据集进行加解密测试，针对不同长度的指令进行验证，分析 ChaCha20-Poly1305 算法的加解密时间、加解密 CPU 使用率、加解密内存占用量，以及不同样本比例下的加解密吞吐量，ChaCha20-Poly1305 算法在通信数据集下的加解密分析结果如图 5 所示。同时，对包含 5 000 条导航位置信息的数据集进行加解密测试，收集上述指标信息，ChaCha20-Poly1305 算法在导航数据集下的加解密分析结果如图 6 所示。

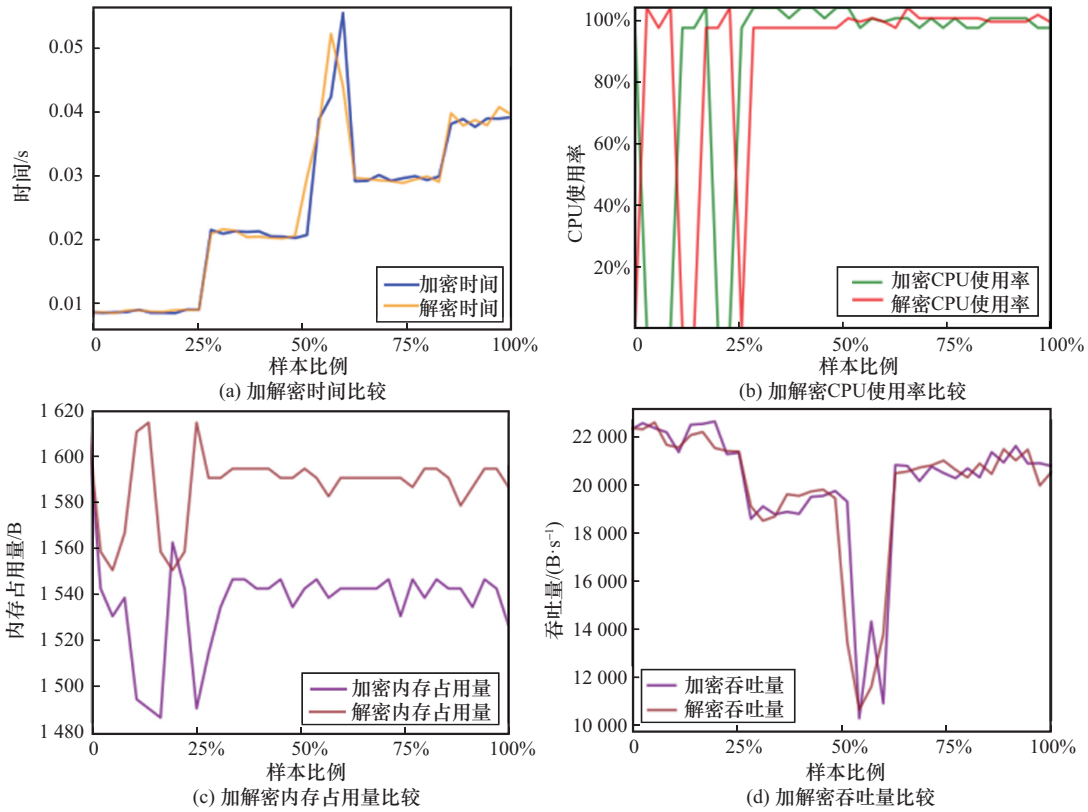
1.5　ECC_AES 算法分析

已有研究探讨了 AES 与 ECC 混合算法在计算机网络通信数据加密中的应用[19]。本文的 ECC_AES 结合椭圆曲线密钥协商与 AES-GCM 认证加密，属于非对称密码和对称密码组合的方案。加密端利用临时私钥与接收方公钥计算共享秘密，并由其派生 256 bit AES 密钥，再通过 AES-GCM 加密消息。解密端利用接收方私钥与发送方临时公钥计算相同共享秘密，派生相同密钥，并验证认证标签。接收方公钥应来自可信渠道，接收到的临时公钥需进行有效性检查。ECC_AES 流程见算法 7。其中哈希派生方式沿用原稿，实际部署还需验证密钥派生与身份认证设计。

该实验运行在 Windows 操作系统上，处理器为 13th Gen Intel Core i5-13450HX (2.40 GHz)，Python 版本为 3.10，内存大小为 16.0 GB。与算法 2 所示算法性能测试框架相同，利用 ECC_AES 方案对包含 100 条通信指令的数据集进行加解密测试，针对不同长度的指令进行验证，分析 ECC_AES 方案的加解密时间、加解密 CPU 使用率、加解密内存占用量，以及不同样本比例下的加解密吞吐量，ECC_AES 算法在通信数据集下的加解密分析结果如图 7 所示。同时，对包含 5 000 条导航位置信息的数据集进行加解密测试，收集上述指标信息，ECC_AES 算法在导航数据集下的加解密分析结果如图 8 所示。

1.6　RSA_AES 算法分析

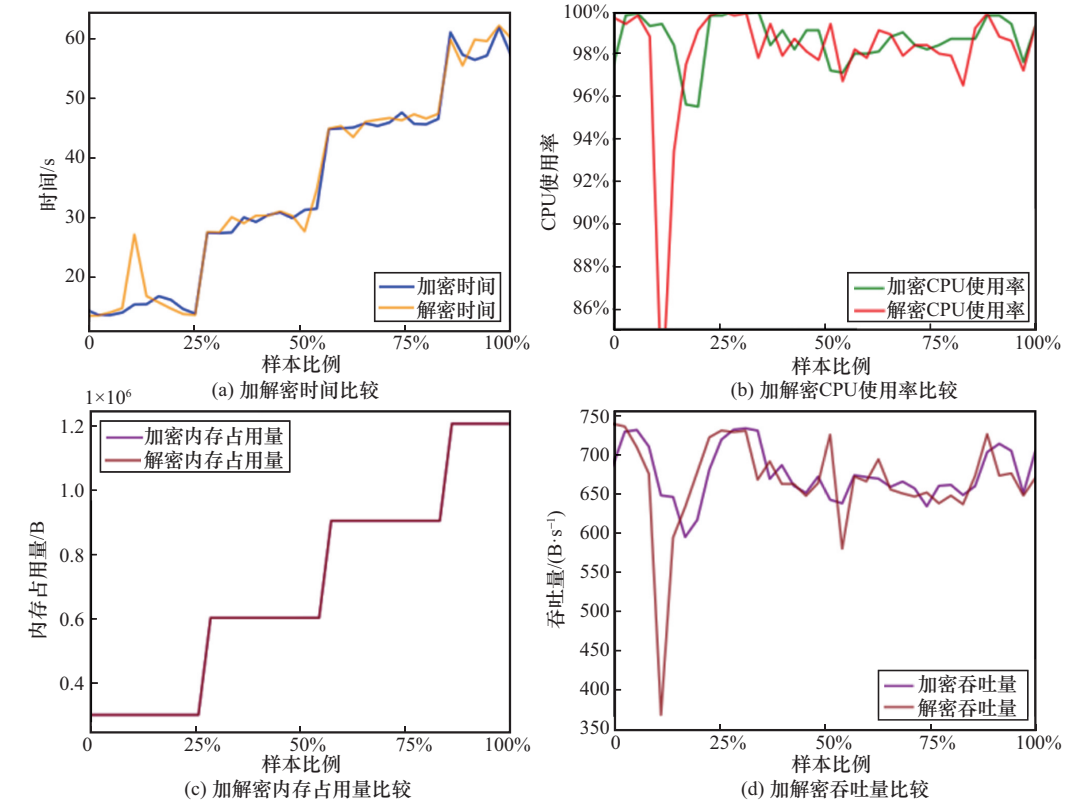
已有研究探讨了基于 AES 加密优化算法的组合加密方法[20]。RSA_AES 结合 RSA-OAEP 与 AES-GCM。



(a) 加解密时间比较　(b) 加解密CPU使用率比较

(c) 加解密内存占用量比较　(d) 加解密吞吐量比较

图 5　ChaCha20-Poly1305 算法在通信数据集下的加解密分析结果



(a) 加解密时间比较　(b) 加解密CPU使用率比较

(c) 加解密内存占用量比较　(d) 加解密吞吐量比较

图 6　ChaCha20-Poly1305 算法在导航数据集下的加解密分析结果

算法 7　ECC_AES 加解密流程

ECC_AES 算法
系统参数
设 E 为素数域 F_p 上的椭圆曲线
设 $G \in E(F_p)$ 为阶为素数 n 的基点
哈希函数 $H:\{0,1\}^* \to \{0,1\}^{256}$ 采用 SHA-256
对称加密采用 AES-GCM 模式
密钥生成
用户 U 生成：
$(d_U, Q_U) \leftarrow$KeyGen()
其中 $d_U \leftarrow \$[1, n-1]$　{私钥}
$Q_U = d_U \cdot G \in E(F_p)$　{公钥}
加密算法 $\mathrm{Enc}(Q_R, m)$
输入: 接收方公钥 $Q_R \in E(F_p)$, 明文 $m \in \{0,1\}^*$
输出: 密文 $C = (c, \mathrm{nonce}, \mathrm{tag}, Q_S)$
1) 生成临时密钥对
$d_S \leftarrow \$[1, n-1]$　{临时私钥}
$Q_S \leftarrow d_S \cdot G$　{临时公钥}
2) 计算共享秘密
$S \leftarrow d_S \cdot Q_R$　{ECDH 共享秘密}
$K \leftarrow H(Sx \| Sy)$　{密钥派生}
3) 对称加密
$(c, \mathrm{nonce}, \mathrm{tag}) \leftarrow$ AES - GCM.Enc (K, m)
4) 返回
$C \leftarrow (c, \mathrm{nonce}, \mathrm{tag}, Q_S)$
解密算法 $\mathrm{Dec}(d_R, c)$
输入: 接收方私钥 $d_R \in [1, n-1]$,
密文 $C = (c, \mathrm{nonce}, \mathrm{tag}, Q_S)$
输出: 明文 m 或 $\perp$
1) 计算共享秘密
$S \leftarrow d_R \cdot Q_S$　{ECDH 共享秘密}
$K \leftarrow H(Sx \| Sy)$　{密钥派生}
2) 对称解密
$m \leftarrow$ AES_GCMDecrypt $(K, c, \mathrm{nonce}, \mathrm{tag})$
3) 返回
若认证成功
返回 m
否则
返回 $\perp$

加密端首先生成随机 AES 会话密钥，用 AES-GCM 加密明文并生成认证标签，再用接收方 RSA 公钥通过 OAEP 填充加密会话密钥。解密端先使用 RSA 私钥恢复会话密钥，再使用收到的 nonce 和认证标签进行 AES-GCM 解密与验证；任一步骤失败均返回错误。对于同一 AES 密钥，nonce 应保持唯一。RSA_AES 流程见算法 8。

算法 8　RSA_AES 加解密流程

RSA_AES 算法	
1. 系统参数	私钥 sk
对称加密	密文 $C = (c, v, \tau, c_k)$
AES_GCM-256 (密钥长度 k = 256,	输出
nonce 长度 n = 96, 标签长度	明文 m 或错误符号 $\perp$
t = 128)	RSA-OAEP 密钥解密
非对称加密	$k \leftarrow$ RSA - OAEP.Dec (sk, c_k)
RSA-OAEP（模数长度 2 048 bit，	按 OAEP 规范解密并校验编码
哈希函数及 MGF1 均采用 SHA-256）	若解密失败，则返回 $\perp$
密钥空间	AES-GCM 解密
$K_{\mathrm{sym}} = \{0,1\}^{256}$；RSA 密钥见下述生成过程	$m \leftarrow$ AES_GCM.Dec (k, v, c, τ)
2. 密钥生成	子程序验证标签，失败则返回错误
生成 RSA 密钥对	若解密失败，则返回 $\perp$
$(pk, sk) \leftarrow$ RSA.KeyGen (2 048)	返回 m
其中 pk = (e, N), sk = (d, N)	5. AES-GCM 子程序
返回 (pk, sk)	函数 AES_GCM.Enc (k, m)
3. 加密过程 Enc(pk, m)	输入: 密钥 k, 明文 m
输入	生成: $v \leftarrow \$\{0,1\}^{96}$
公钥 pk	初始化密码对象：
明文 $m \in \{0,1\}^*$	ctx.AES_GCM(k, v)
输出	计算 c = ctx.encrypt (m)
密文 $C = (c, v, \tau, c_k)$	生成标签 τ = ctx.digest()
生成临时对称密钥	返回 (c, v, τ)
$k \leftarrow \$K_{\mathrm{sym}}$	函数 AES_GCM.Dec (k, v, c, τ)
AES-GCM 加密	输入: 密钥 k, nonce v,
$(c, v, \tau) \leftarrow$ AES_GCM.Enc (k, m)	密文 c, 标签 τ
nonce 由 AES-GCM.Enc 内部生成	使用密文中接收的 nonce v
保留返回的密文、nonce 与标签	初始化密码对象：
RSA-OAEP 密钥加密	ctx.AES_GCM(k, v)
$c_k \leftarrow$ RSA - OAEP.Enc (pk, k)	解密：m = ctx.decrypt (c)
按 OAEP 规范编码并加密	若标签验证 ctx.verify (τ) 成功
返回密文	返回 m
$C \leftarrow (c, v, \tau, c_k)$	否则
4. 解密过程 Dec(sk, C)	返回 $\perp$
输入	

该实验运行在 Windows 操作系统上，处理器为 13th Gen Intel Core i5-13450HX (2.40 GHz)，Python 版本为 3.10，内存大小为 16.0 GB。与算法 2 所示算法性能测试框架相同，利用 RSA_AES 方案对包含 100 条通信指令的数据集进行加解密测试，针对不同长度的指令进行测试，分析 RSA_AES 方案的加解密时间、加解密 CPU 使用率、加解密内存占用量，以及不同样本比例下的加解密吞吐量，RSA_AES 算法在通信数据集下的加解密分析结果如图 9 所示。同时，对包含 5 000 条导航位置信息的数据集进行加解密测试，收集上述指标信息，RSA_AES 算法在导航数据集下的加解密分析结果如图 10 所示。

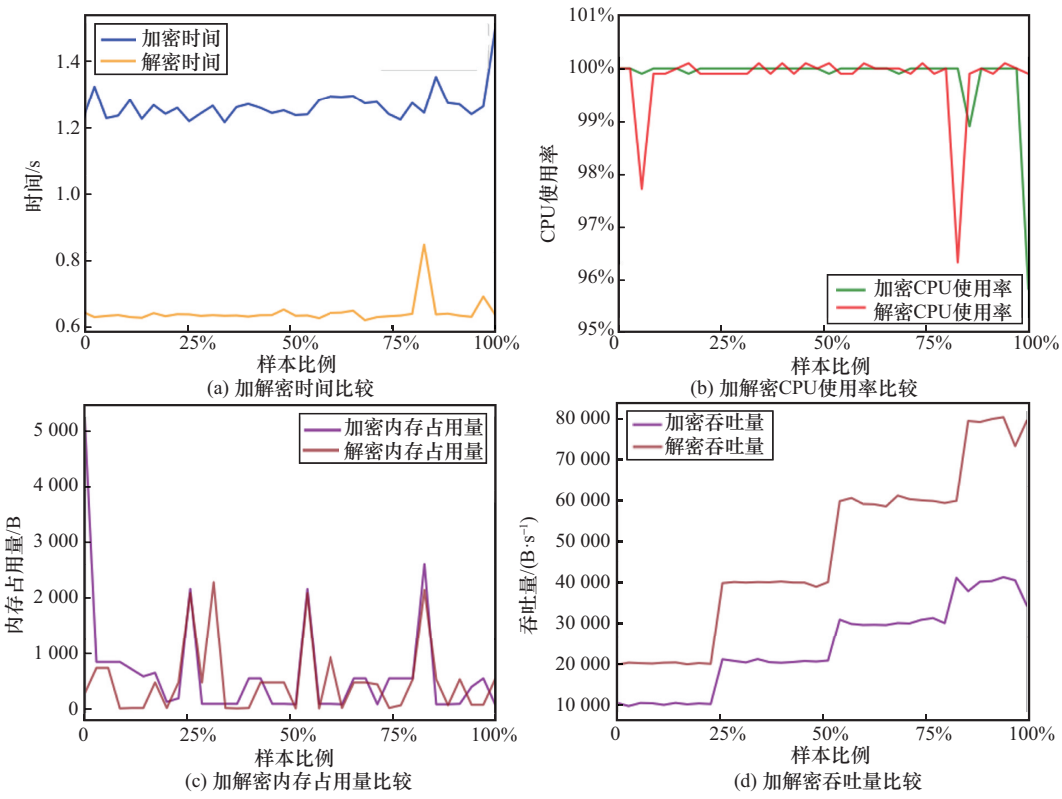


图7　ECC_AES算法在通信数据集下的加解密分析结果

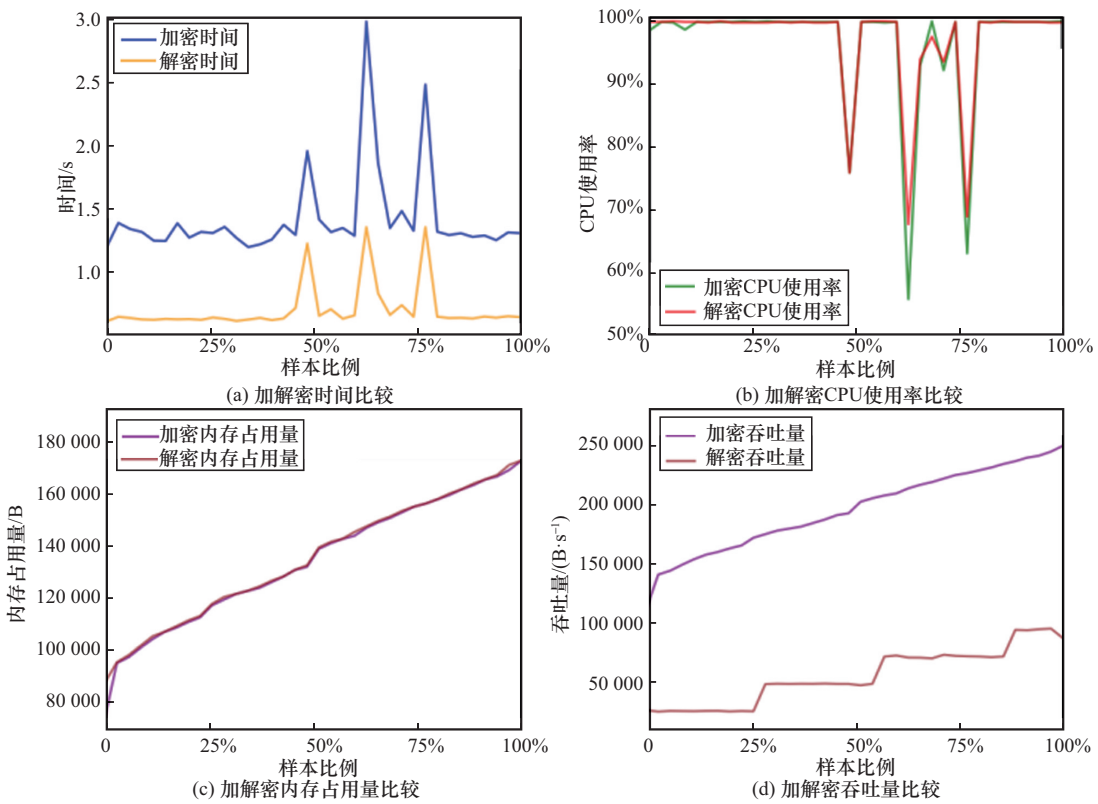


图8　ECC_AES算法在导航数据集下的加解密分析结果

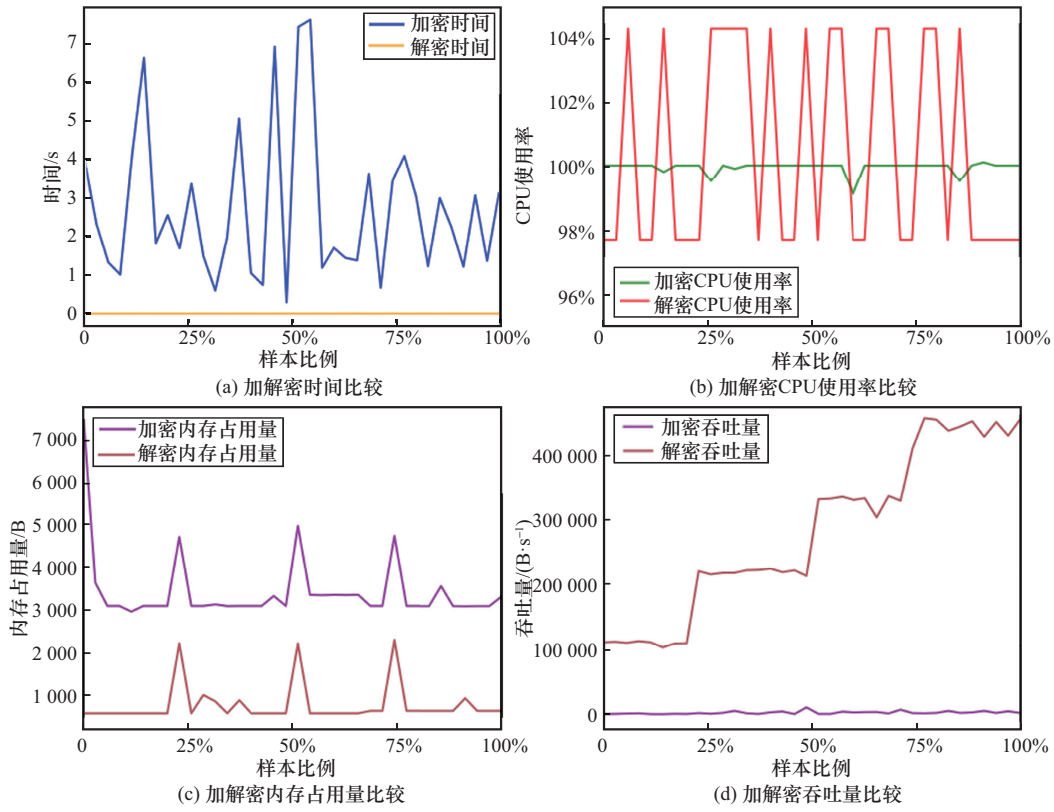


图 9　RSA_AES 算法在通信数据集下的加解密分析结果

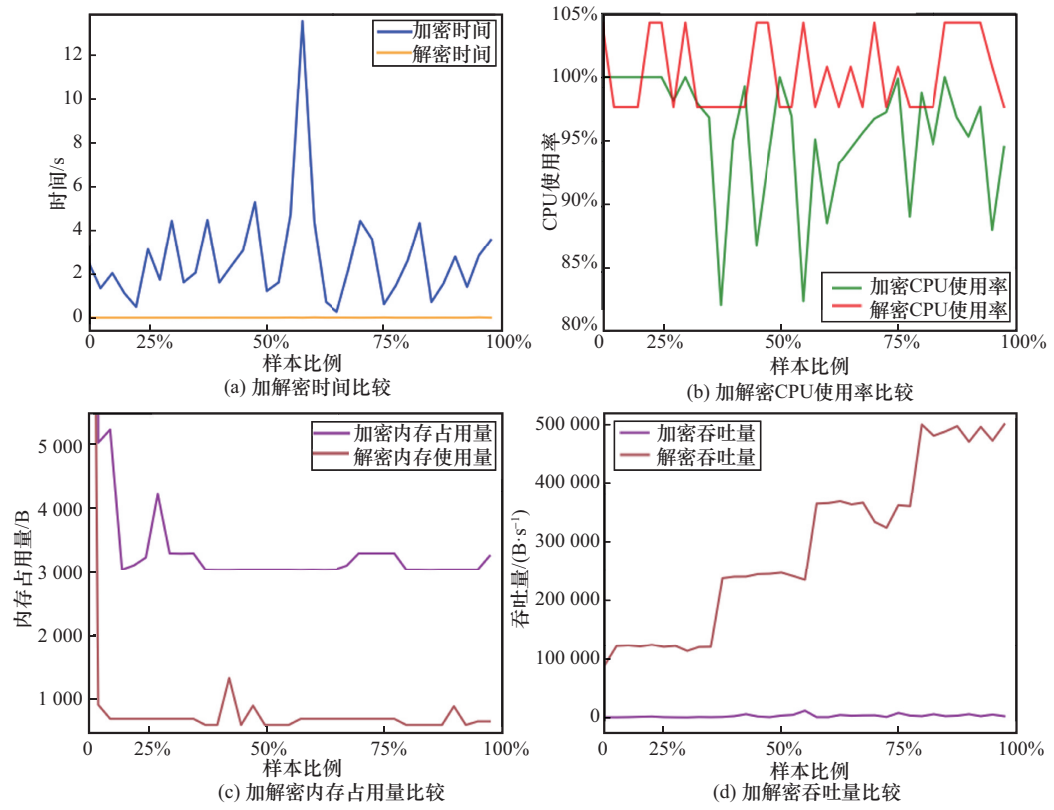


图 10　RSA_AES 算法在导航数据集下的加解密分析结果

2　加密算法性能分析

下面比较ECC_AES和RSA_AES 5种方案在两类数据集下的时间与资源开销。Ascon和ChaCha20-Poly1305提供认证加密，Speck为分组密码，ECC_AES和RSA_AES还涉及非对称密码操作，因此各方案的功能和安全属性并不完全相同。表3汇总了加密时间、解密时间、加密CPU使用率、解密CPU使用率、加密内存占用量、解密内存占用量、加密吞吐量和解密吞吐量8项指标。性能比较应在数据字节数、实现参数及计时范围一致的前提下进行。

表3　5种加密方案性能比较

算法	明文选择	所占比例	性能指标							
			加密时间/s	解密时间/s	加密CPU使用率	解密CPU使用率	加密内存占用量/B	解密内存占用量/B	加密吞吐量/(B·s^{-1})	解密吞吐量/(B·s^{-1})
Ascon	通信数据集	25%	0.109 7	0.110 4	100%	99.91%	3 479.2	5 243.6	117 252	116 487
		50%	0.218 2	0.217 6	100.06%	99.98%	6 755.6	10 067.6	117 891	118 192
		75%	0.325 5	0.325 1	99.9%	100.03%	9 972	14 891.6	118 576	118 701
		100%	0.452 0	0.446 4	99.99%	89.63%	13 190	19 718.4	114 281	115 506
	导航数据集	25%	0.741 2	0.741 2	100.02%	100.01%	22 535	33 827.6	120 269	120 254
		50%	1.476 6	1.478 1	99.99%	100.02%	44 868	67 234.8	120 696	120 564
		75%	2.342 8	2.324 1	99.66%	98.73%	67 140	100 644	115 655	115 751
		100%	3.078 0	3.065 1	99.61%	99.38%	89 396	134 060	116 077	116 661
Speck	通信数据集	25%	0.014 3	0.016 9	/	/	2 750.4	2 615.6	603 511	20 612
		50%	0.013 7	0.016 5	/	/	3 420.8	3 342.4	1 990 205	21 232
		75%	0.015 0	0.016 9	/	/	5 260	5 184	2 628 517	20 756
		100%	0.014 7	0.016 6	/	/	7 549	7 435.6	3 478 091	21 174
	导航数据集	25%	0.013 8	0.016 6	/	/	11 922.4	11 741.6	6 835 900	21 124
		50%	0.014 1	0.016 7	/	/	23 784.8	23 706.4	12 633 220	21 080
		75%	0.014 3	0.016 8	/	/	35 696	35 620	18 668 288	20 857
		100%	0.014 3	0.016 6	/	/	48 006.6	47 893.2	24 835 258	21 101
ChaCha20-Poly1305	通信数据集	25%	0.008 9	0.008 8	99.5%	100.3%	1 520	1 587.2	21 437.19	21 671.28
		50%	0.021 6	0.020 7	98.32%	102.87%	1 538.4	1 588.4	18 237.03	18 871.45
		75%	0.029 9	0.029 5	99.87%	99.87%	1 543.2	1 590.4	19 790.67	20 071.02
		100%	0.038 4	0.038 5	100.05%	99.74%	1 540.4	1 590.8	20 599.73	20 563.18
	导航数据集	25%	14.923	15.869	98.5%	96.9%	300 071	300 127	672.66	655.25
		50%	29.567	29.854	98.81%	98.54%	600 082	600 120	677.94	672.54
		75%	45.687	45.987	98.43%	98.11%	900 175	900 175	656.64	652.57
		100%	58.521	58.522	99.2%	89.34%	1 200 101	1 200 127	684.21	684.12
ECC_AES	通信数据集	25%	1.258 7	0.642 0	99.9%	99.71%	626	339.5	10 181	19 951
		50%	1.251 1	0.643 4	99.99%	99.99%	421.8	662.6	2 048	39 825
		75%	1.273 6	0.642 2	99.99%	99.98%	555.3	535.9	30 185	59 847
		100%	1.301 6	0.671 6	99.46%	99.61%	448.5	539.4	39 477	76 868
	导航数据集	25%	1.290 4	0.639 1	99.75%	99.98%	103 325	105 151	148 789	23 022
		50%	1.352 0	0.713 7	97.67%	97.63%	128 227	128 624	184 648	45 798
		75%	1.648 8	0.820 0	90.68%	92.41%	150 604	151 044	216 870	68 824
		100%	1.291 7	0.653 2	100.01%	99.99%	172 006	172 451	247 689	88 398
RSA_AES	通信数据集	25%	2.969 1	0.014 7	99.975%	99.325%	3 640	576	750.74	108 671
		50%	2.319 5	0.014 7	99.94%	101.6%	3 225.9	835.9	3 092.19	217 753
		75%	3.127 6	0.014 7	99.88%	100.95%	3 430.1	790.125	2 918.05	326 360
		100%	2.576 8	0.014 6	99.96%	99.65%	3 270	823.8	2 985.18	438 053
	导航数据集	25%	1.961 8	0.015 0	99.23%	99.86%	3 753.3	696.2	817.03	119 512
		50%	2.980 2	0.016 8	92.13%	100.1%	3 039	705	3 927.59	239 056
		75%	3.403 0	0.019 5	94.51%	99.45%	3 160	672	3 175.96	353 793
		100%	2.428 3	0.017 7	95.8%	101.3%	3 066.7	626.5	3 632.45	484 144

根据表 3 所列数值，Speck 在通信数据集 50%、75% 和 100% 比例以及全部导航数据集比例下的加密时间较短，加密吞吐量在各组中均较高；但在通信数据集 25% 比例下，ChaCha20-Poly1305 的加密和解密时间分别为 0.008 9 s 和 0.008 8 s，均小于 Speck 对应数值。在通信数据集 50%、75% 和 100% 比例下，RSA_AES 的解密时间低于 Speck。对于导航数据集，25% 比例下 RSA_AES 解密时间更短，其余比例下 Speck 解密时间更短。内存方面，ECC_AES 在通信数据集中的加密内存占用量最低；解密内存占用量在 25%、50%、75% 和 100% 比例下也均为 ECC_AES 最低。RSA_AES 在导航数据集中具有较低的加解密内存占用量。解密吞吐量方面，在通信数据集 25% 比例下 Ascon 最高，其余通信数据集比例和全部导航数据集比例下 RSA_AES 最高。上述比较仅依据表中记录；在统一核验数据字节数、计时范围和吞吐量计算口径后，才能进一步判断各方案的综合性能。

3　结束语

卫星与水下设备之间的安全通信需要同时考虑密码功能、计算开销和设备资源。本文在通用计算平台上比较了 5 种加密方案，现有记录显示 Speck 在多数测试条件下具有较短的加密时间，但不同方案在解密时间和内存占用量等指标上各有优势。Ascon 和 ChaCha20-Poly1305 提供认证加密，ECC_AES 和 RSA_AES 结合了密钥协商或密钥传输与认证加密；Speck 的 ECB 实现则不具备相同的认证功能。因此，不能仅依据运行速度判断方案的安全性或实际部署适用性，也不能据此认定某一方案在所有条件下最优。

后续工作需统一数据编码、处理字节数、算法参数和性能统计口径，并在目标卫星与水下设备硬件上验证时间、内存和能耗开销。同时，应结合密钥管理、身份认证、完整性保护和侧信道防护评估完整通信方案，以提高性能比较的可复现性和实际应用价值。

参考文献

[1] 孙茜, 刘慧梁, 王冀莲 . 卫星互联网信息安全风险分析与发展建议[J]. 中国电子科学研究院学报, 2023, 18(5): 469-475.
Sun Q, Liu H L, Wang J L. Risk analysis and development suggestions for satellite internet information security[J]. Journal of China Academy of Electronic Sciences, 2023, 18(5): 469-475.

[2] 向鹏,李德仁 . 天地互联网的智能遥感卫星[J]. 高科技与产业化, 2023(9): 12-15.
Xiang P, Li D R. Intelligent remote sensing satellite for space-earth internet[J]. High-Tech and Industrialization, 2023(9): 12-15.

[3] Xie W, Yang M, Gong J B. Underwater attack-defense confrontation system and its future development[J]. Chinese Journal of Engineering Science, 2019, 21(6): 71-79.

[4] 马多贺, 杨建, 江俊烨, 等 . 卫星互联网的安全风险分析与防护[J]. 保密工作, 2023(6): 54-56.
Ma D H, Yang J, Jiang J Y, et al. Security risk analysis and protection of satellite internet[J]. Confidential Work, 2023(6): 54-56.

[5] Adnan S F S, Isa M A M, Hashim H. Timing analysis of the lightweight AAβ encryption scheme on embedded Linux for internet of things[C]//Proceedings of the 2016 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE). Piscataway: IEEE Press, 2016.

[6] Oualha N, Nguyen K T. Lightweight attribute-based encryption for the internet of things[C]//Proceedings of the 2016 25th International Conference on Computer Communication and Networks (ICCCN). Piscataway: IEEE Press, 2016.

[7] Hosseinzadeh J, Ghaemi Bafghi A. Software implementation and evaluation of lightweight symmetric block ciphers of the energy perspectives and memory[EB/OL]. arXiv, 2017, arXiv:1706.03909.

[8] Usman M, Ahmed I, Aslam M I, et al. SIT: a lightweight encryption algorithm for secure internet of things[J]. International Journal of Advanced Computer Science and Applications, 2017, 8(1).

[9] Gupta M, Abdelsalam M, Khorsandroo S, et al. Security and privacy in smart farming: challenges and opportunities[J]. IEEE Access, 2020, 8: 34564-34584.

[10] Andrea I, Chrysostomou C, Hadjichristofi G. Internet of things: security vulnerabilities and challenges[C]//Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC). Piscataway: IEEE Press, 2015: 180-187.

[11] Buchanan W J, Li S C, Asif R. Lightweight cryptography methods [J]. Journal of Cyber Security Technology, 2017, 1(3-4): 187-201.

[12] McKay K A, Bassham L, Turan M S, et al. Report on lightweight cryptography: NISTIR 8114[R]. Gaithersburg, MD: National Institute of Standards and Technology, 2017.

[13] Turan M S, McKay K, Chang D, et al. Status report on the final round of the NIST lightweight cryptography standardization process: NISTIR 8454[R]. Gaithersburg, MD: National Institute of Standards and Technology, 2023.

[14] Manifavas C, Hatzivasilis G, Fysarakis K, et al. Lightweight cryptography for embedded systems-a comparative analysis[C]//Data Privacy Management and Autonomous Spontaneous Security. Berlin, Heidelberg: Springer, 2014: 333-349.

[15] 武岳 . 轻量级加密算法在物联网中应用的研究[J]. 科技创新与生产, 2024, 45(9): 126-129.
Wu Y. Research on the application of lightweight encryption algo-

rithm in the internet of things[J]. Scientific and Technological Innovation and Productivity, 2024, 45(9): 126-129.

[16] 俞惠芳, 郭欣. 物联网系统中网络编码混合加密方案[J]. 信息安全研究, 2025, 11(4): 326-332.
Yu H F, Guo X. Hybrid encryption scheme of network coding in internet of things system[J]. Information Security Research, 2025, 11 (4): 326-332.

[17] 石鲁生, 朱慧博. 面向无线传感网络安全的轻量级加密算法研究[J]. 传感技术学报, 2025, 38(1): 168-173.
Shi L S, Zhu H B. Research on lightweight encryption algorithm for wireless sensor network security[J]. Journal of Sensing Technology, 2025, 38(1): 168-173.

[18] 施玲玲, 陶亮. 基于ASCON加密算法的信息安全保护模型构建[J]. 佳木斯大学学报(自然科学版), 2025, 43(4): 35-38+43.
Shi L L, Tao L. Construction of information security protection model based on ASCON encryption algorithm[J]. Journal of Jiamusi University(Natural Science Edition), 2025, 43(4): 35-38+43.

[19] 袁国旗, 曹鸿飞. 基于AES与ECC混合算法的计算机网络通信数据加密研究[J]. 长江信息通信, 2025, 38(4): 123-124+127.
Yuan G Q, Cao H F. Research on data encryption of computer network communication based on AES and ECC hybrid algorithm[J]. Yangtze River Information and Communication, 2025, 38(4): 123-124+127.

[20] 李温晗华, 杨凯. 基于AES加密优化算法的组合加密方法[J]. 信息与电脑(理论版), 2022, 34(15): 232-234.
Li W H H,Yang K. Combined encryption method based on AES encryption optimization algorithm[J]. Informationand Computer (Theoretical Edition), 2022, 34(15): 232-234.

[作者简介]



李成杰（1979− ），男，西南民族大学副教授、硕士生导师，主要研究方向为信息安全技术、智能信息处理技术、通信抗干扰技术、现代通信中的信息处理技术。



崔丽琪（1999− ），女，西南民族大学硕士生，主要研究方向为网络信息安全。



应惟龙（2000− ），男，西南民族大学硕士生，主要研究方向为空间信息网络、深度强化学习。



朱立东（1968− ），男，博士，电子科技大学教授、博士生导师，主要研究方向为卫星通信的传输与组网、星地网络融合等。



郑艺（2001− ），女，西南民族大学硕士生，主要研究方向为空间信息网络、卫星通信抗干扰技术、资源分配等。



李明秀（2001− ），女，西南民族大学硕士生，主要研究方向为联邦学习、通信信号处理。