从共识到混乱：RAFT算法的漏洞评估

Tamer Afifi、Dr. Abdelfatah Hegazy、Dr. Ehab Abousaif，阿拉伯科学、技术与海运学院计算与信息技术学院计算机科学系，埃及开罗

摘要——**近几十年来，RAFT分布式共识算法已成为分布式系统生态的重要支柱，确保多个节点间的数据一致性和容错性。尽管RAFT以简洁、可靠和高效著称，但其安全属性尚未得到充分认识，使得其实现容易受到各种攻击和威胁，可能将RAFT协调一致的共识局面转变为数据不一致的混乱状态。本文对RAFT协议进行了系统的安全分析，重点考察其易受消息重放攻击和消息伪造攻击等安全威胁影响的情况。本文研究了恶意行为者如何利用该协议的消息传递机制重新引入旧消息，从而扰乱共识过程并导致数据不一致。本文通过模拟场景考察了这些攻击在实践中的可行性，并识别了RAFT设计中使这些攻击得以实施的关键弱点。为应对这些漏洞，本文提出一种基于密码学、经认证的消息验证和新鲜度检查的新方法。该方案为增强RAFT 实现的安全性、指导开发更具韧性的分布式系统提供了框架。**

关键词——***RAFT***；共识协议；安全；分布式系统；消息伪造；重放攻击；密码学

I. Introduction

如今，分布式系统遍布世界各地，在幕后运行，悄无声息地协调多个计算机系统之间的任务，而这些系统往往分布在不同地理位置 [2]。从全球通信网络到关键金融交易，这些系统依赖分布式系统的基本原则来确保可靠性和数据一致性 [1]。分布式共识算法是确保这种可靠性的核心组成部分，它使自主节点即使在发生网络故障或节点崩溃时，也能够就单一值达成一致 [9]。近几十年来，RAFT分布式共识算法已成为分布式系统生态的支柱之一。

RAFT算法是确保分布式共识中数据完整性的基础要素。它以易于理解和实现为设计目标[3]，，不仅简化了构建弹性分布式应用的过程，也成为新共识研究的比较标准。RAFT通过在节点间复制状态，实现其容错这一主要目标，使其能够抵御系统故障和崩溃。

然而，尽管RAFT以容错性和简洁性著称，其共识机制本身通常基于“失效即停止”模型（节点崩溃或网络分区），并假设故障为非恶意故障[10]。因此，与其设计相关的安全问题，尤其是针对主动恶意攻击的缓解措施，往往未能充分纳入标准实现或得到充分覆盖[4, 5]。这一缺口使实现容易受到威胁和攻击，可能危及协议所确立的RAFT状态机和数据完整性，并可能导致关键应用程序中的数据损坏和服务中断[8]。

现有 RAFT 实现中的这一缺陷带来了显著的安全漏洞风险；安全的共识算法不仅要确保各方达成一致，还要保护整个分布式系统生态，保护数据、防止中断，并最终减少故障和数据损坏。尤其是，该协议缺乏自身的端到端安全机制，因此容易受到潜在威胁，包括针对 RAFT 消息传递机制并扰乱共识过程的重放攻击和消息伪造攻击。因此，本文将在后续章节中分析并评估RAFT算法的安全属性，重点考察其易受消息重放攻击和伪造攻击影响的情况，以展示其设计弱点。本研究的主要目标是提出并验证一种新颖、模块化且轻量级的安全增强方案，该方案基于加密和经认证的消息验证，旨在缓解这些漏洞并增强 RAFT 安全实现。

为实现这一目标，本文后续章节将首先在第二节构建必要的分布式系统与共识背景。随后，第三节将深入介绍 RAFT 分布式共识协议的细节。第四节将全面分析 RAFT 的漏洞，重点关注消息完整性。第五节介绍有关安全 RAFT 实现及对策的相关工作，并阐明研究空白。第六节提出用于缓解已识别安全问题的方案。第七节详述用于测试和验证的方法论，接着第八节展示模拟攻击的结果和性能评估。第九节专门讨论研究发现、其影响以及研究人员的观点。第十节总结本研究成果。最后，第十一节概述未来研究方向。

II. Background

A. 分布式系统

分布式系统是当今技术生态的核心基础，从支撑全球通信网络到协调关键金融交易，无处不在。分布式系统可定义为由一组自主计算元素（节点）组成的集合，在用户看来，它们构成一个统一连贯的系统[1], 在这个系统中，无论节点的数量、位置或组成部分如何，它们都作为一个整体协同运行，无论用户与系统在何处、何时以及以何种方式进行交互。

分布式系统为单个分布式应用的各个组件提供了相互通信的途径。同时，它向每个应用程序隐藏了硬件和操作系统之间的差异；这可以通过中间件来实现。

从某种意义上说，中间件对于分布式系统而言，就如同操作系统对于计算机一样：它是资源管理器，为应用程序提供服务，使其能够在网络中高效地共享和部署这些资源 [2]。

图1展示了基本的分布式系统架构及其主要组件之间的关系：硬件、应用程序、操作系统和中间件层。中间件层在协调通信以及抽象网络中的硬件差异方面发挥着至关重要的作用。

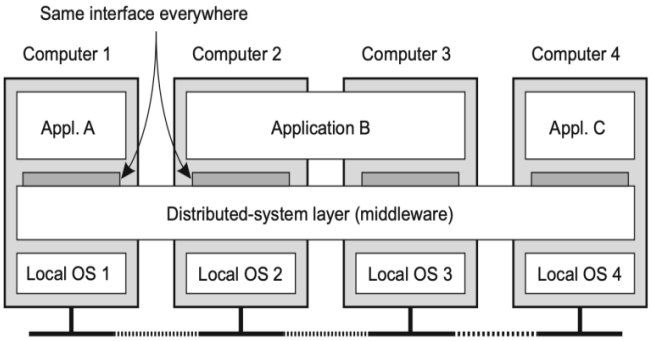

图1。 中间件层跨越多台机器，并为每个应用程序提供相同的接口。

中间件、节点、应用程序以及这个不断扩展的分布式系统领域，需要确保通信和交换消息的稳定性与一致性，并让所有参与方了解系统中的所有正确且最新的更新；因此，共识算法的重要作用便在于满足这些需求。

B。 分布式共识

通常，共识是指多个参与方达成一致；例如，一群朋友决定一起吃午饭时，选择去哪家餐厅就是一种共识。在计算机科学，尤其是分布式系统中，共识基本上也是这个意思：它可以定义为分布式系统中的多个节点就消息、事务或对象的取值达成一致的过程。这是分布式系统面临的一项基本挑战。然而，一旦节点就某个值达成一致，这种共识就应当是最终的。

一些最早的共识算法实现依赖基于投票的机制，这类机制具有合理的容错性，并有严谨的数学证明来确保完整性和稳定性。一些流行的基于投票的算法包括 Paxos 和 RAFT。Paxos 最初由 Leslie Lamport 于 1989年提出，但由于多种因素，直到 2001年年底才发表，其中当然包括其众所周知的高复杂性，这阻碍了它的广泛应用。为此，Diego Ongaro 于 2014年推出了 RAFT算法，作为 Paxos 的一种更简单的替代方案。RAFT 的设计强调简洁易管理，同时保持与前代算法相同水平的容错性和一致性。

III. RAFT 分布式共识协议

本节简要概述 Raft 分布式共识算法。如需了解更详细的说明，请参阅原始论文《寻找一种易于理解的共识算法》[3]。

A. *Raft*

Raft 是一种广泛应用且可靠的分布式共识算法，旨在成为其复杂前身 Paxos 算法的一种更易理解、更易实现的替代方案。Raft 是一种容错协议，依赖单个当选领导者、日志复制和概率方法。

为了了解 Raft 的工作原理，我们可以设想一个由三个节点（或副本）组成的服务器集群，每个节点都运行一个状态机、日志和 Raft 协议。只要它们一开始的状态相同，并以相同顺序执行相同操作，最终就应处于相同状态。每当某个副本收到一条命令（例如设置一个带有值的新键），它就会将该命令追加并保存为日志中的新条目。为了让其他副本保持同步，每个副本的日志都必须始终包含完全相同的命令序列。

在 Raft 集群中，一个节点会被选为领导者，其他节点则充当跟随者。领导者负责处理所有客户端请求，并将日志条目复制到跟随者，以确保它们保持同步。

B. 领导者选举

在任意时刻，Raft 集群中的每个节点只能处于一种状态：领导者、跟随者或候选者。最初，所有节点都以跟随者状态启动。每个跟随者都会设置一个随机选举超时时间。因此，如果跟随者连续未能收到心跳，它就会认为当前没有可用的领导者，并转换为候选者状态以启动选举 [13]。如果获得了全部选票中的多数，它就会赢得选举并成为新的领导者。节点当选后，会一直担任领导者，直到发生崩溃，或观察到另一个任期更高的节点 [15]。如果选举导致选票分裂，或另一位候选者当选，该节点就会恢复为跟随者状态，等待新领导者的心跳。图 2 展示了 RAFT 领导者选举流程。

领导者选举可能因任何原因启动，例如领导者故障或离线、网络延迟过高，或者网络分区隔离了某个跟随者，使其在领导者仍然存活时达到选举超时时间。

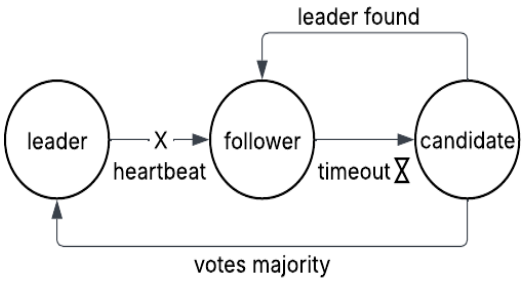

图2. RAFT基本领导者选举流程。

领导者当选后，便负责记录系统中的所有新变化。领导者会定期向 RAFT 集群中的所有跟随者发送追加条目消息。这些消息有双重作用：既作为心跳，防止跟随者发起新一轮选举，也指示跟随者复制新的日志条目，从而确保集群中的一致性。

C. 日志复制

系统中的所有变化，例如新增命令或事务，均由领导者专门处理。客户端将变化发送给领导者，领导者接收后会在其日志中追加一个新条目。在领导者将该日志条目复制到所有跟随者节点之前，该条目一直处于未提交状态。随后，领导者等待集群中多数节点（>50%）确认新的值；确认后，领导者提交该条目，并通知跟随者该条目已提交，此时集群即被视为达成共识。图 3 展示了 RAFT 的基本日志复制流程。

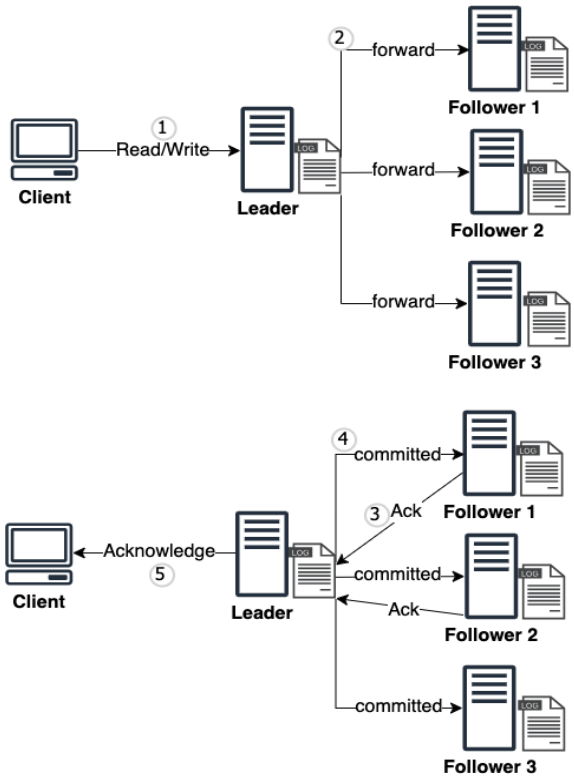

图3. RAFT基本日志复制流程。

根据 RAFT 共识[19, 21] 算法的创始人 Ongaro 和 Ousterhout [3], 的说法，RAFT 最初的设计目标是比 Paxos 更易理解、更简单且效率更高。此后，它成为众多应用程序的基础，这些应用程序需要容错数据存储和一致状态管理。然而，尽管 RAFT 在应对系统故障时具有韧性和稳健性，其基于消息传递的架构仍容易受到各种安全威胁的影响，需要进一步研究，包括消息伪造和重放攻击。

IV. RAFT SECURITY ANALYSIS

尽管 Raft 共识算法是分布式系统中实现状态复制的一项关键技术[14]，并以其简洁、高效和可靠而闻名，但其安全漏洞仍是一个值得深入探讨的关键领域。本节将探讨一些已知威胁和攻击方法，并讨论破坏共识、危及系统稳定性可能造成的后果。

A. 拒绝服务（*DoS*）攻击

拒绝服务（DoS）攻击是攻击者试图通过大量消息淹没领导者或其他节点，阻止它们正常运行并耗尽其资源的一种恶意行为。

由于 Raft 采用强领导者模型，恶意领导者可以故意延迟或丢弃消息，发动 DoS 或审查攻击，从而扰乱日志复制，并导致跟随者发起不必要的选举。此外，攻击者可以控制多个节点，并通过多数投票确保其中一个受控节点当选领导者[4]。

B. 拜占庭攻击 / 消息伪造或冒充

在 RAFT 共识算法中，附近的恶意/非法节点发起的拜占庭攻击（包括消息伪造和冒充）可能会破坏系统完整性。事实上，添加内容可以看作由某个外部且可能怀有恶意的实体秘密插入系统的消息[12]。在冒充攻击中，恶意节点会伪造身份，试图冒充合法节点/跟随者，以破坏共识机制 [5]。

发送伪造消息的拜占庭攻击可能导致交易延迟、数据不一致、数据泄露，甚至损害系统完整性。

这会对性能产生可衡量的影响。例如，可以通过建模来展示交易延迟与攻击者数量之间的线性关系。总提交时间 (t_c) 等于复制时间 (t_r) 与交易延迟 (t_l) 之和：

交易提交时间可按如下方式计算：

$$t_c = t_r + t_l$$

而交易延迟则

$$t_l = RTT(1 + q)(1 + p)$$

这表明，攻击者数量增加时，交易延迟也会随之线性增加。

表 I. 符号说明

符号说明	描述
t_c	交易提交时间
t_r	交易请求时间
t_l	交易延迟
RTT	请求/响应消息的往返时间
p	拜占庭节点的百分比（= n/m）
q	攻击成功率
n	拜占庭节点数量
m	所有节点的数量

C. 消息重放攻击

前述一般性漏洞为更复杂的攻击创造了条件。尽管许多类型的攻击都构成显著威胁，但消息重放攻击对 RAFT 等共识协议而言是一种严重且常被忽视的威胁。消息重放攻击是一种网络攻击，恶意行为者会拦截合法的数据传输，捕获有效消息，并在之后恶意地重新传输该消息。在分布式共识的背景下，这种攻击利用协议对消息传递的依赖来欺骗节点，扰乱共识过程和系统状态（符号说明见表 I）。

在 RAFT 集群中，攻击者可以通过捕获先前发送的有效消息（如请求投票消息），并在之后将其重新注入网络，从而发起重放攻击。这可能导致一个或多个跟随者响应并投票，进而触发一场新的、不必要的选举，并造成领导者与跟随者分裂。

RAFT 上的消息重放攻击之所以能够成功，主要是因为该协议缺乏内在的重放保护机制，存在易受攻击的弱点。标准 RAFT消息不包含单调递增的序列号，也不包含任何唯一的基于会话的标识符，因而接收节点无法据此检测消息是否为过期的重复消息。缺少新鲜度检查使协议容易受到重放消息的操纵，导致共识崩溃。这可能造成系统不稳定、数据不一致，并损害分布式系统的完整性。

V. Related Work

关键基础设施/系统对分布式系统的依赖日益加深，这凸显了对强健共识机制的需求，此类机制应能抵御崩溃故障和恶意攻击。本节回顾相关研究，以比较并验证本文提出的方案；该方案旨在缓解针对默认 RAFT 实现及其他已解决此类问题的 RAFT 变体的重放/消息伪造攻击。

在分布式系统中实现容错性的理论基础，植根于状态机复制方法 [9], 要求所有副本在相同的初始状态下执行相同的操作序列 [1]。为实现这一目标，我们必须确保不存在恶意因素影响状态机以及交换消息的完整性。RAFT 共识协议 [3] 作为 Paxos 的一种实用替代方案应运而生，着重于易于理解和简洁，同时实现相同的容错特性（能够容忍崩溃故障）。然而，标准的 RAFT 实现假设运行环境为“故障即停止”，并未考虑拜占庭故障——在此类故障中，节点可能恶意行事并影响共识过程的完整性。这一区别至关重要，因为诸如 P-BFT [20] 之类的通用拜占庭协议通常会引入较高的计算开销来应对这一问题；另一方面，这也促使研究人员探索轻量级替代方案，以在有效实现安全缓解措施的同时维持性能基准。

标准 RAFT 设计假定系统处于“可信”环境中，节点均诚实但可能发生崩溃。这一假设使协议本质上容易遭受主动攻击，攻击者可利用其未经身份验证的明文消息结构。已有多项研究尝试识别、解决并形式化描述这些弱点：

1) 原始*RAFT*：Ongaro 和 Ousterhout [3] 的奠基性工作将 RAFT 引入为一种比 Paxos 更易理解的替代方案，重点是在崩溃故障和网络分区情况下维持一致性，并默认网络环境可信。然而，他们并未考虑重放攻击和消息伪造等安全攻击。

2) RaBFT：一种基于 Raft 改进的拜占庭容错共识算法 [10]。它实现了完整的拜占庭容错（在总计 3f+1 个节点中可容忍 f 个恶意节点），引入秘密共享以优化日志消息分发，并采用动态委员会角色来分散领导者压力。该算法改变了日志复制过程和选举逻辑，可防止恶意节点伪造消息；然而，它需要彻底重新设计 RAFT 共识引擎，引入复杂的多方计算步骤（秘密共享验证）和新角色（委员会），并带来大量额外开销。

3) ENGRAFT（飞地保护的 *RAFT*） 是另一个在共识层内运行的安全增强示例，专门用于防范节点自身的拜占庭故障 [4]。它利用可信执行环境（TEEs），例如英特尔 SGX 飞地，保护每个节点上的核心 RAFT 状态和逻辑。在 ENGRAFT 中，所有关键的共识操作，包括状态更新和日志操作，均在安全、隔离的硬件飞地内执行。这种设计能够有效防御内部威胁和拜占庭容错攻击，但也引入了对特定硬件的依赖，并提高了硬件、协议和平台方面的实现复杂度。

4) 应对基于 *RAFT* 的物联网区块链网络中的主动攻击 *[5]* 利用路径损耗：本文提出了一种利用路径损耗的物理层认证机制，以保护无线物联网中的 Raft 免受冒充攻击

环境。所提方案使用发送节点与接收节点之间信号的路径损耗作为独特的设备指纹，以验证发送方身份。该方案的弱点在于容易受到环境变化和多径的影响。

5) 信任与信誉管理系统 *[11]*：该方法根据节点的历史行为为其分配信用评分，以识别并隔离恶意节点。然而，此类系统存在“冷启动”问题，且收敛速度较慢；恶意节点可以长期表现诚实，以建立较高的信任度。此外，在大型分布式系统中，为每个节点存储和更新信誉评分所需的开销难以扩展。

6) P-BFT 与 *RAFT* 的结合：《构建可靠安全网络的新方法》[20] 提出了一种混合模型，将 RAFT 的简洁性和效率与 P-BFT 的强大安全性相结合。该组合旨在构建能够抵御拜占庭故障（恶意行为）的网络。尽管这种方法在保护共识过程免受内部攻击方面非常有效，但 BFT-RAFT 本质上需要对 RAFT 状态机进行重大修改，导致协议执行复杂性增加，开销也更高。

7) 零信任共识算法 *[16]*：这项值得关注的工作旨在弥合 RAFT 与现代安全概念之间的差距，提出了 VSSB-Raft，该协议采用“永不信任，始终验证”的零信任安全模型。它利用可验证秘密共享（VSS）机制和数字签名，确保任何单个节点都无法危害系统，从而实现高安全性。与传统 BFT协议相比，VSSB-Raft 展示了一种以资源高效的方式实现拜占庭容错的方案；然而，其全面的安全保障依赖于对 RAFT状态机进行重大修改，并集成复杂的密码学原语，导致其实现复杂度高于模块化传输层方案。

8) GPBFT（基于群组的实用拜占庭容错）*[17]*：作为对现有 BFT协议的增强方案，它采用双管理员短群签名机制。该方案允许节点使用短群签名验证消息，虽然功能强大，但仍有较高的准入门槛，增加了大量复杂性和通信开销，而且完全以复杂的 BFT协议取代了 RAFT。

9) PB-Raft [18]: 将拜占庭容错（BFT）能力集成到 RAFT 中。该方案分为两个层面：首先，引入加权 PageRank算法来评估节点的声誉和信任度；其次，利用 BLS（Boneh–Lynn–Shacham）阈值签名方案，确保日志条目的真实性和共识。该方案带来了两项主要的复杂性和潜在开销：领导者必须在提交日志之前收集达到阈值的签名；此外，基于 PageRank 的声誉评分需要在集群中持续进行复杂的计算和更新。

对相关工作的回顾揭示了当前 RAFT 安全领域中明显的权衡。尽管该领域的一些既有研究已较为成熟，能够全面抵御恶意攻击，但它们会带来显著开销，并且需要对核心共识逻辑进行复杂修改。这一分析证实了文献中存在一个重要空白：缺少一种模块化、低开销的安全方案，能够在传输层运行，专门且全面地防御针对基础 RAFT 的主动消息伪造和重放攻击，同时不改变共识协议本身的核心。因此，本研究旨在提出一种安全传输层机制来填补这一空白，在不改变 RAFT状态机的情况下提供必要的消息认证和新鲜性，具体内容将在下一节详述。

VI. Proposed Solution

本文讨论了 Raft 分布式共识协议、其机制及其重要性，并探讨了它的一些安全风险。

本节旨在缓解消息重放攻击带来的漏洞，并提出一种与 RAFT 集成的安全传输层方案，以模块化的方式保护集群节点之间的通信，同时保持原始RAFT不变。该方案结合现代密码技术，确保消息机密性、真实性，并通过某种校验和确保完整性，同时利用消息缓存缓解重放攻击。所提方案旨在对原始RAFT算法进行轻量级修改，在显著提升其安全属性的同时确保其效率。所提方案的架构如图 4 所示。

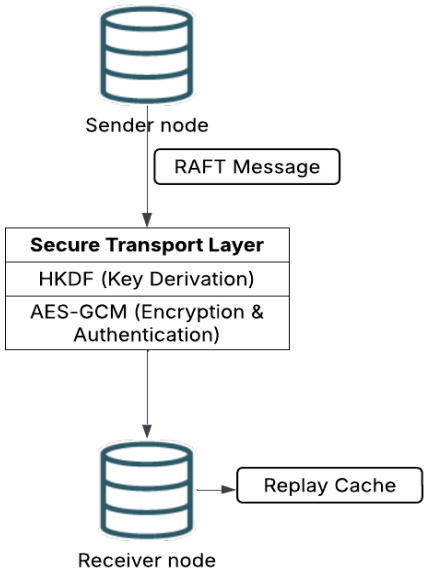

图 4. 所提方案基本流程。

所提方法包含三个关键组成部分：

A. 唯一加密密钥

所提方案使用基于 HMAC 的密钥派生函数（HKDF） [6] 为每条消息生成唯一的临时加密密钥，而不是依赖一个长期密钥来处理所有通信。HKDF 根据共享主密钥为每条消息派生一个新密钥。这是一项非常重要的安全最佳实践，因为即使攻击者能够攻破某条消息的密钥，也无法攻破其他任何消息，而泄露的密钥将毫无用处。

B. 认证加密

在所提方案中，通信期间使用 AES-GCM（伽罗瓦计数器模式下的高级加密标准（AES））密码算法对 RAFT 消息进行加密和身份验证。[7]

- AES：该算法部分负责加密消息内容，确保消息在传输过程中无人能够读取。为了在 AES 过程中实现最高级别的加密，将使用 GCM [7] 模式。
- GCM：该部分会生成认证标签，以确保消息未遭篡改，且确实来自持有正确密钥的有效发送方。这在 RAFT 中至关重要，因为节点必须确信日志条目或投票真实且未被篡改。

以下是解释该算法的伪代码：

函数 secure-envelope(明文, confidential, peer¯id, K_master, key_id):

// 生成每条消息唯一的标识符

随机数 ← random(12)

tx_id ← 随机数(16)

// 为此消息派生唯一密钥

K_tx ← HKDF-SHA256(K_master, 随机数, peer_id)

// 构造关联认证数据 (AAD)

aad ← combine(key_id, 随机数, tx_id, peer_id)

如果机密：

// 使用 AES-GCM 加密并进行身份验证

ct_合并后的← AES-GCM.encrypt(K¯tx, nonce, plaintext, aad)

密文 ← ct_合并后的[:-16]

标签 ← ct_合并后的[-16:]

否则：

// 仅进行身份验证（生成标签）

密文 ← 明文

标签 ← 计算_标签(K_tx, aad, 明文)

结束条件判断

// 返回最终的安全信封

返回 { key_id, 随机数, tx_id, 密文, 标签}

结束函数

AES-GCM 算法的架构和完整流程，从初始化向量开始，到生成附加到加密消息上的标签为止，如图 5 所示。

C. 重放缓存

重放缓存是防御重放攻击的重要额外防护层。重放缓存通过维护一个列表，记录所有最近处理过的消息的标识符（唯一消息 ID）。收到新消息时，系统会将其标识符与缓存中的记录进行比对。如果找到该 ID，系统会立即将该消息作为重放消息拒绝，从而阻止攻击得逞；否则，系统会接受该消息，并将其 ID 记录到重放缓存中。

以下伪代码说明了缓存检查过程：

数据结构：缓存 # 内部数据结构// 用于¯检查消息是否曾出现过的函数 function SEEN(peer¯id, tx_id): key ← combine(peer_id, tx_id) if key in cache:

return True // 检测到重放 else: return False // 新消息 end if end function // 用于记录新消息已出现过的函数 function REMEMBER(peer¯id, tx_id):key ← combine(peer_id, tx¯id) cache.add(key) end function.

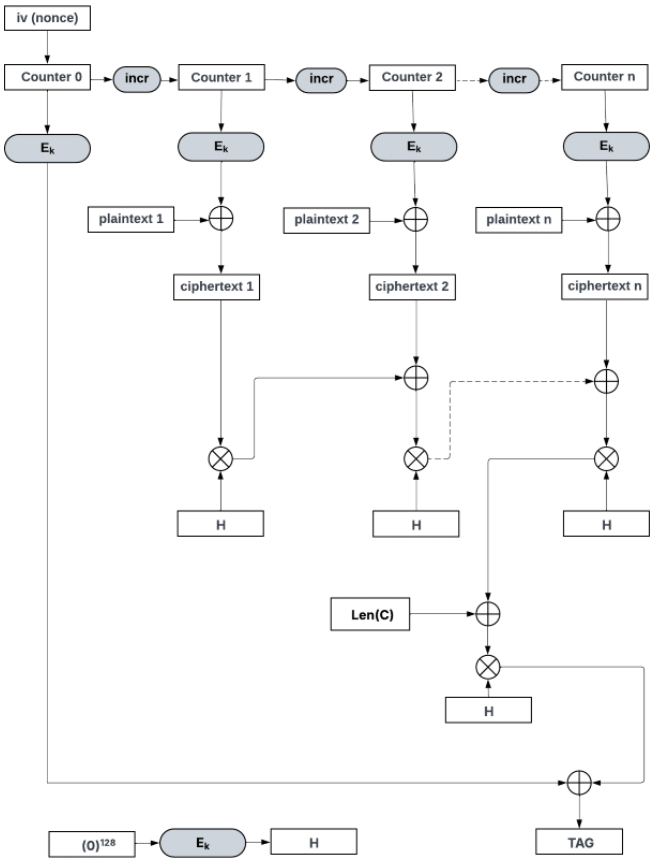

图 5. AES-GCM。

VII. 方法论ETHODOLOGY

本节概述了用于评估所提方案的方法和工具，该方案旨在缓解针对 RAFT 集群的重放攻击和消息伪造。主要目标是评估在实施所提安全方案前后，重放攻击对 RAFT

集群的影响，以证明 RAFT 的脆弱性以及所提方案的有效性。

A. 实验设置与架构

本研究的方法论与架构侧重于基于 Python 的原生 RAFT 实现，该实现基于 https://github.com/nikwl/raft-lite；物理实验室设置由 RAFT 集群中的三个节点构成，这三个节点均运行 Linux（Ubuntu 服务器）操作系统，并通过使用 IPv4 地址的以太网进行通信。攻击模拟器和所提方案均使用 Python 开发，如图 6 所示。

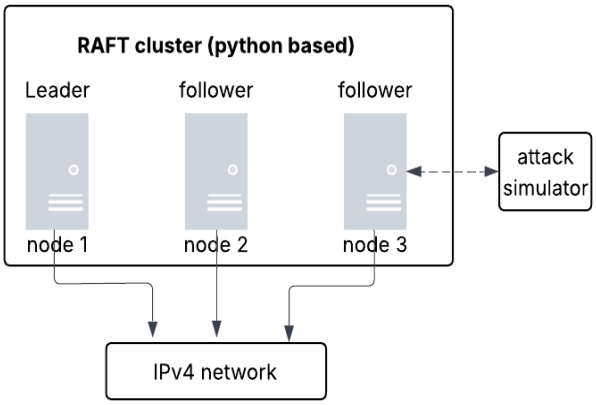

图 6. 物理实验室架构。

B. 测试流程

测试过程分为两个不同阶段，分别评估标准默认 RAFT 行为在实施所提方案前的基线漏洞，以及所提方案实施后的有效性。

1) 实施前测试（基线漏洞）：第一阶段重点是对默认 RAFT 集群实施模拟消息重放攻击，以建立可供后续衡量的安全基线。

a) 攻击模拟设置：使用自定义客户端脚本，通过 Python 的 socket 库与三节点 RAFT 集群中的一个跟随者节点建立 TCP 连接。

b) 攻击执行：使用专用攻击脚本（replaylast）和（entryattack）捕获一条有效的 RAFT 消息（例如心跳或日志条目），随后不作任何修改便将其重放，以模拟日志复制并引入恶意或过时的日志条目，从而构成明确的拜占庭消息伪造和重放攻击场景。

2) 实施后测试（所提方案的有效性）：第二阶段是在 RAFT 集群的全部三个节点上实施所提安全机制，并启用安全功能。随后再次启动相同的攻击模拟，并像第一阶段那样监控集群行为。

a) 测试流程：启用安全功能后，对修改后的集群执行了与实施前阶段完全相同的步骤和攻击脚本。

b) 方案结果：所提方案修改后的 RAFT 实现成功拒绝并丢弃了所有攻击消息，原因如下：

- 合法流量：在整个测试过程中，所有合法流量和当前消息均持续被接受并正确处理，证实了该方案的预期功能。
- 重放的合法消息：接收方检查后发现其唯一 ID 已存在于本地缓存中（未通过新鲜度检查），因此丢弃了先前捕获并随后重放的合法消息。
- 新伪造消息：由于未能通过针对前文引入并说明的标签进行的集成身份验证检查，新伪造的恶意消息被丢弃。

3) 性能评估：在应用所提方案的安全增强并完成测试与验证后，我们开展了性能测试，以评估相较于默认 RAFT 实现，所提方案引入的开销。测试中使用业界标准工具 raft-bench 测量了关键性能基准。

VIII. 结ESULTS

本节呈现安全和性能测试的客观结果，不作详细解读；详细解读将留待讨论部分。

A. 安全评估结果

安全评估证实了默认 RAFT 实现存在漏洞，也验证了所提方案的有效性：

1) 实施前测试（基线漏洞）：如预期，默认 RAFT 实现缺乏基本的重放保护，因此成功接受并处理了攻击消息。这一行为破坏了日志一致性，最终损害了共识状态。

2) 实施后测试（所提方案的有效性）：实施后测试阶段展示了所提安全传输层的有效性。所提方案修改后的 RAFT 实现成功拒绝并丢弃了所有攻击消息，原因如下：

a) 合法流量：在整个测试过程中，所有合法流量（当前消息）均继续被正确接受和处理，证实了该方案的针对性功能。

b) 重放的合法消息：接收方检查后发现，这些先前捕获并在之后重放的合法消息，其唯一 ID 已存在于本地缓存中，因此被丢弃（未通过新鲜度检查）。

c) 新伪造消息：新伪造的恶意消息因集成的

身份验证检查未通过而被丢弃，该检查针对前文引入并说明的标签。

3) 性能基准：性能测试评估了所提方案相较于默认 RAFT 实现的开销。结果汇总于表 II 中。

表 II. 性能分析

指标	默认 Raft	所提方案	变化 %
吞吐量	297.19	269.61	-9.28%
延迟	468.01	539.44	15.26%

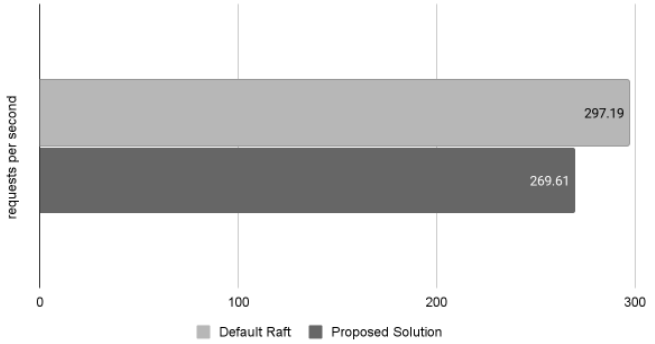

图 7. 吞 吐量性能分析：默认方案与 .所提方 案。

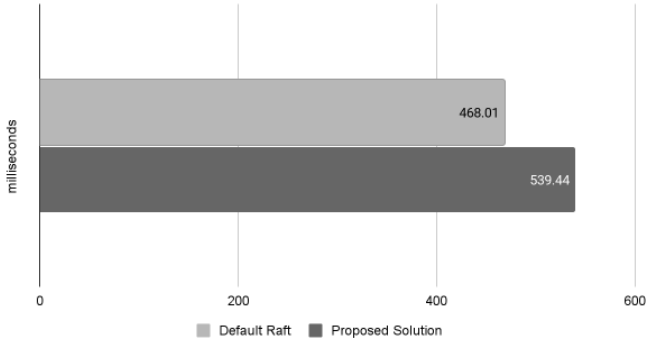

图 8. 默认方案与所提方案的延迟性能分析。

如图 7、图 8 和表 II 所示，结果表明，所提方案的吞吐量下降了 9.28%，延迟增加了 15.26%。这一性能代价直接源于为每条消息生成并验证安全加密消息和唯一标识符所需的计算开销，体现了典型的安全性与性能权衡。

IX. 讨论ISCUSSION

本研究结果表明，所提方案能够有效缓解针对 RAFT 的两种关键攻击：消息重放和消息伪造。实施前的基线测试证实，标准 RAFT 设计容易受到重放与伪造攻击，因为它依赖未经身份验证的消息，且缺少新鲜度检查。

实施后的分析表明，所提方案成功拒绝了所有重放的合法消息以及所有新伪造的恶意消息。这些高效的结果得益于密码学强制的消息认证与基于事务 ID 的重放缓存的结合。认证加密（AES-GCM）确保发送方的合法性以及消息内容的完整性，而重放缓存使用的唯一消息标识符则成功实现了对基于时序的重放攻击的缓解。

安全性与性能权衡：性能评估揭示了安全性与效率之间典型且可量化的权衡。与默认的非安全 RAFT 实现相比，所提方案使吞吐量下降了 9.28%，交易延迟增加了 15.26%。这一性能代价直接源于为每条发送和接收的消息生成唯一加密密钥（HKDF）并执行认证加密（ AES-GCM）所需的计算开销。

本研究的关键洞见在于，所获得的安全性——即防御伪造和重放等系统攻陷攻击——需要付出最低限度开销的代价，尤其是在金融系统或敏感数据账本等任务关键型环境中，数据完整性至关重要。这种经过评估的性能影响，是实现高水平数据完整性和真实性所需付出的可接受代价。

与替代方法的比较：该解决方案的模块化设计是其关键差异化优势。不同于其他先进的安全方案，例如 RaBFT [10], , 后者要求彻底重新设计核心 RAFT 共识引擎；或基于路径损耗的解决方案 [5], , 后者依赖专用的物理层测量；我们的方案完全运行于传输层。这种方法以较低的复杂性和对核心 RAFT 逻辑的最小影响，实现了缓解重放与伪造攻击这一主要目标。同时，它还改善了各组件的开发生命周期，不会影响其他组件，未来甚至无需替换整个模块即可换用性能更优的模块。

局限性：该方法在运行层面的主要局限是，重放缓存的大小可能会随着长期运行而增长。未来需要对其进行管理，以防止过度的内存消耗。

X. 结论

本文成功评估了 RAFT 分布式共识协议的安全漏洞，并识别出其易受主动消息重放与伪造攻击的严重弱点。为缓解这些威胁，本文提出了一种新颖的模块化安全传输层集成方案，该方案基于经认证的消息验证和新鲜度检查。与以往工作不同，后者要么重新设计 RAFT 核心协议、增加沉重负担，要么只针对特定的消息漏洞（如消息伪造），要么依赖专用硬件或物理层属性；本文提出的模块化安全传输层方案无需改变核心共识逻辑，也不会带来显著开销。

以下贡献凸显了这项工作的科学价值：

1) 漏洞验证：通过实际模拟证明，默认 RAFT 协议很容易遭到破坏，从而瓦解共识并损害数据完整性；攻击者可利用未经身份验证的 RAFT消息实现这一目的。

2) 模块化解决方案：通过采用 HKDF [6] 进行唯一密钥派生，并采用 AES-GCM [7] 同时实现加密和身份验证，可以开发并验证一种轻量级的基于传输层的安全增强方案，而无需更改默认 Raft 逻辑或增加繁重负担。

3) 重放攻击防护：引入唯一的事务标识符和重放缓存机制，可有效保护协议免受此前成功实施的重放攻击。

4) 消息伪造防护：相同的唯一事务标识符与计算所得的标签共同构成额外的防御层，确保发送方及消息本身的真实性。

5) 性能表征：衡量安全性与性能之间的权衡，并提供实验数据，表明在较低开销下即可实现显著的安全性提升（吞吐量下降9.28%，延迟增加15.26%）。

表 III 简要比较了所提方案与其他 RAFT 实现的性能。

表 III。 性能分析

item	原始RAFT	RaBFT	路径损耗（物理层）	所提方案
主要 *Goal*	崩溃容错（崩溃容错）和日志一致性	拜占庭容错（拜占庭容错）	针对冒充攻击的身份验证。	缓解重放/消息伪造攻击
缓解	None	秘密共享、动态委员会角色、修改后的选举/复制。	作为设备的路径损耗测量指纹。	认证加密与防重放缓存。
层	应用层	应用层/共识（协议重新设计）	物理层	传输
开销/复杂性	简单，易于理解的逻辑。	高。需要彻底重新设计协议，并进行复杂的多方计算。	中等。需要进行初始校准。	低，对核心逻辑。
弱点	易受攻击	高性能和高复杂性开销。	需要专用硬件，且易受环境变化的影响。	缓存文件大小增长。

本研究结果证实，缺少消息认证是标准 RAFT 中最大的安全弱点。我们的解决方案提供了一个适用性较高的框架，可用于增强 RAFT 实现，适合优先考虑数据完整性而非原始速度、且无法容忍完整拜占庭容错（BFT）协议复杂性的场景。

尽管该解决方案能有效抵御中间人重放与伪造攻击，但其依赖逐消息缓存，因此在长期运行期间会面临内存可扩展性和缓存管理方面的实际限制。

XI. 未UTURE 来ORK

未来研究将着眼于克服当前局限、拓展现有范围，并将这一模块化解决方案转变为可投入生产的框架。未来研究应涵盖以下领域：

- 缓存优化：实施并评估生存时间（TTL）和最近最少使用（LRU）淘汰策略，以确保重放缓存保持高效，并动态管理内存消耗。
- 安全密钥生命周期管理：开发一种安全、动态的流程，定期轮换主密钥，以防范服务器遭受长期攻击。
- 扩大攻击领域验证范围：拓展验证范围，测试该解决方案应对其他网络层漏洞的能力，例如拒绝服务（DoS）攻击。

REFERENCES

[1] Tel Gerard，《分布式算法导论》，第2版。剑桥大学出版社；2000年。[2] Maarten van Steen、Andrew S. Tanenbaum，《分布式系统简要介绍》，发表于 Springer，Computing（2016）98:967–1009，DOI 10.1007/s00607-016-0508-7，第4–6页。[3] D. Ongaro 和 J. Ousterhout，《寻找一种易于理解的共识算法》，2013年10月草稿。[4] Weili Wang、Sen Deng、Jianyu Niu、Michael K. Reiter、Yinqian Zhang，《ENGRAFT：面向拜占庭故障节点的飞地保护 Raft》，第8页。[5] H. M. Buttar、W. Aman、M. Rahman 和 Q. H. Abbasi，《应对基于 RAFT 的物联网区块链网络中的主动攻击》，格拉斯哥大学电子与纳米工程系，arXiv预印本 arXiv:2204.00838，2022年。[6] H. Krawczyk 和 P. Rogaway，《基于HMAC的提取与扩展密钥派生函数（HKDF）》，RFC 5869，2010年5月。[7] 美国国家标准与技术研究院，《分组密码工作模式建议：伽罗瓦/计数器模式（GCM）和GMAC》，NIST特别出版物 800-38D，马里兰州盖瑟斯堡，2007年11月。[8] Nathan Sealey、Adnan Aijaz、Ben Holden，《IOTA Tangle 2.0：迈向可扩展、去中心化、智能且自主的物联网生态系统》，arXiv:2209.04959v1，2022年。[9] SCHNEIDER, F. B.《使用状态机方法实现容错服务：教程》。ACM计算机调查 22, 4（1990年12月），299–319。[10] X. Chen、R. Liu 和 G. Wang，《RaBFT：一种基于raft的改进型拜占庭容错共识算法》，《超级计算杂志》，第80卷，第14期，第1-28页，2024年6月。[11] L. Mastromauro、D. Souto Andrade、M. Ozgur Ozmen 和 M. A. Kinsy，《安全

分布式系统中的数据复制：共识算法攻击与防御综述》，发表于IEEE Access，第13卷，第143631-143667页，2025年，doi: 10.1109/ACCESS.2025.3597260。[12] N. Santoro，《分布式算法的设计与分析》。新泽西州霍博肯：Wiley-Interscience，2007年。[13] D. Huang、X. Ma 和 S. Zhang，《私有区块链的 Raft 共识算法性能分析》，2018年，arXiv:1808.01081。[14] M. Zhang，《物联网信息安全管理中改进型Raft共识算法的实现》，《国际先进计算机科学与应用期刊》（IJACSA），第15卷，第6期，第153–161页，2024年。[15] A. J. J. Davis、M. Demirbas 和 L. Deng，《LeaseGuard：正确实现Raft租约》，2025年，arXiv:2512.15659。[16] W. Liu、Z. Wang、Z. Liu 和 Y. Wang，《VSSB-Raft：一种用于区块链的安全高效零信任共识算法》，《IEEE工业信息学汇刊》，2024年。[17] J. Hu、FenYan 和 Z. Wu，《BGRAFT：基于RSA加密算法的分组式抗拜占庭RAFT共识算法》，2024年人工智能与数字技术国际会议（ICAIDT），中国深圳，2024年，第279-283页，doi: 10.1109/ICAIDT62617.2024.00067。

[18] Shi, H.、Chen, Z.、Cheng, Y. 等。PB-Raft：一种基于加权PageRank和BLS门限签名的拜占庭容错共识算法。《Peer-to-Peer Netw. Appl.》18，26（2025）。https://doi.org/10.1007/s12083-024-01876-8[19] Oleiwi, Wed.（2025）。PPRCA：保护隐私的RAFT共识算法——分布式网络的下一代共识。巴比伦大学纯粹与应用科学期刊，223-238。10.29196/jubpas.v33i1.5657。[20] Bogdanov, A.、Shchegoleva, N.、Khvatov, V.、Dik, G.、Kiyamov, J.、Dik, A.（2023）。P-BFT与RAFT的结合：构建可靠安全网络的新方法。载于：Gervasi, O. 等，《计算科学及其应用–ICCSA 2023研讨会》。ICCSA 2023。《计算机科学讲义》，第14104卷。Springer，Cham。https://doi.org/10.1007/978-3-031-37105-9_38[21] Y. Li、Y. Fan、L. Zhang和J. Crowcroft，“无线网络中的RAFT共识可靠性：概率分析”，《IEEE物联网期刊》，第10卷，第14期，第12839-12853页，2023年7月15日，doi：10.1109/JIOT.2023.3257402。