北京信息科技大学

毕业设计（论文）

题　　目： 基于 eBPF 的内核 SM4 加密方法研究

学　　院： 计算机学院

专　　业： 计算机科学与技术

学生姓名： 赵帅阳 **班级/学号** 计科 2104/2021011930

指导老师/督导老师： 冯温迪

起止时间： 2025 年 2 月 17 日 至 2025 年 6 月 6 日

毕业设计（论文）任务书

学院 ：计算机学院　　专业：计算机科学与技术　　班级：计科 2104

<table>
<tr><td colspan="2">学生情况</td><td colspan="3">指导教师情况</td><td colspan="2">题目类型</td></tr>
<tr><td rowspan="2">姓 名</td><td rowspan="2">学 号</td><td rowspan="2">姓 名</td><td rowspan="2">职 称</td><td rowspan="2">单 位</td><td>理工专业</td><td>文、管、经专业</td></tr>
<tr><td>理论研究 □</td><td>理论研究 □</td></tr>
<tr><td rowspan="2">赵帅阳</td><td rowspan="2">2021011930</td><td rowspan="2">冯温迪</td><td rowspan="2">副教授</td><td rowspan="2">计算机学院</td><td>科研开发 ☑</td><td>应用研究 □</td></tr>
<tr><td>工程设计 □</td><td>调查研究 □</td></tr>
<tr><td>题目</td><td colspan="3">基于 eBPF 的内核 SM4 加密方法研究</td><td>是否实物型毕设</td><td>是□ 否☑</td></tr>
<tr><td>主要内容以及目标</td><td colspan="5">（毕业设计应完成的主要内容，设计任务达到的目标）
设计并实现基于 eBPF 和 XDP 技术的 SM4 加密方法，主要包括数据包的解析、SM4 加密算法的内核实现，并利用 eBPF 和 XDP 实现数据的加解密。设计任务包括以下几个模块：
（1）系统配置模块;
（2）数据包解析模块;
（3）数据加解密模块;</td></tr>
<tr><td>成果形式</td><td colspan="5">（毕业设计完成具体工作量；成果形式；验收方式）
（1）完成以上目标和主要内容的系统设计、程序编制和调试；
（2）提交软件原型完整的系统源代码与可执行代码；
（3）按学校要求撰写并完成毕业设计论文，以及对英文资料的翻译。</td></tr>
<tr><td>基本要求</td><td colspan="5">（对完成设计任务方面的具体要求：对理工专业应提出设计技术参数、数据及来源、调试所用仪器设备等）
（1）要求按照毕业设计的进度计划来开展毕业设计工作，并及时提交开题报告、毕业论文等相关文档；
（2）要求系统能够正确运行；
（3）论文编写要符合学校毕设工作手册中的相关规范。</td></tr>
<tr><td>实习调研要求</td><td colspan="5">（对部分有实习环节的专业，提出实习或调研的具体要求，包括调研提纲、实习时间、地点和具体内容要求；文、管、经专业提出对论文论点有关论据、数据和素材的搜集要求）
无</td></tr>
</table>

<table>
<tr><td>主要参考文献</td><td colspan="3">[1] Høiland-Jørgensen T, Brouer J D, Borkmann D, et al. The express data path: Fast programmable packet processing in the operating system kernel[C]//Proceedings of the 14th international conference on emerging networking experiments and technologies. 2018: 54-66.
[2] Vieira M A M, Castanho M S, Pacífico R D G, et al. Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications[J]. ACM Computing Surveys (CSUR), 2020, 53(1): 1-36.
[3] 李胡, 彭长根, 侯金秋. 流密码框架下的 SM4 专用认证加密算法[J]. Journal of Computer Engineering & Applications, 2024, 60(2).
[4] 魏鹏娟.国密分组算法 SM4 研究综述[J].陕西交通科教研究, 2023(1):27-31.
[5] 吕述望,苏波展,王鹏,等.SM4 分组密码算法综述[J].信息安全研究, 2016, 2(11):13.DOI:CNKI:SUN:XAQY.0.2016-11-005.</td></tr>
<tr><td>主要仪器设备或开发环境</td><td colspan="3">（根据毕业设计题目情况需要，各学院统一填写要求）
服务器：配备 Mellanox Connect-X 5 100 GbE 高性能网卡；
操作系统: Ubuntu 20.04 LTS；
开发语言：C、Shell；</td></tr>
<tr><td>毕业设计（论文）开始日期</td><td>2025 年 2 月 17 日</td><td>毕业设计（论文）完成日期</td><td>2025 年 6 月 6 日</td></tr>
<tr><td colspan="4">毕业设计（论文）进度计划（起止时间、工作内容）</td></tr>
<tr><td colspan="4">第 1 周　完成开题报告撰写及答辩演示文稿；
第 2-5 周　实现 SM4 加密算法的内核级代码，对 SM4 算法进行初步性能评估；
第 6-8 周　实现 eBPF 与 SM4 加密模块的集成，对集成后的系统进行功能和性能测试，根据测试结果调整和优化系统，完成中期检查。
第 9-11 周 完成毕业论文撰写；
第 12-13 周 完成论文定稿并制作答辩演示文稿；
第 14-15 周 完成毕业设计答辩。</td></tr>
<tr><td colspan="4">指导教师（签字）：冯涵迪　　2025 年 01 月 07 日
督导教师（签字）：　　年　月　日</td></tr>
<tr><td colspan="4">学院毕业设计（论文）领导小组审查意见：

组长（签字）：　　年　月　日</td></tr>
</table>

毕业设计（论文）原创性声明

本人郑重声明：所呈交的毕业设计（论文），题目为《基于 eBPF 的内核 SM4 加密方法研究》，是本人在导师指导下，进行研究工作所取得的成果。尽我所知，除了文中特别加以标注的内容外，本毕业设计（论文）的研究成果不包含任何他人创作的、已公开发表或者没有公开发表的作品的内容。对本毕业设计（论文）所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明并表示了谢意。本毕业设计（论文）原创性声明的法律责任由本人承担。

作者签名：赵帅阳

2025 年 05 月 12 日

毕业设计（论文）版权授权声明

本人完全了解北京信息科技大学关于收集、保存、使用本科生毕业设计（论文）的要求，按照学校要求提交毕业设计（论文）的印刷本和电子版本。学校有权保留毕业设计（论文）并向相关机构送交论文的电子版和纸质版，允许论文被查阅和借阅，可以采用影印、缩印或扫描等复制手段保存、汇编毕业设计（论文）。学校有权适当复制、公布论文的全部或部分内容。

指导教师签名：冯温迪　　作者签名：赵帅阳

2025 年 05 月 12 日　　2025 年 05 月 12 日

摘要

数字时代，网络安全威胁日益复杂多样。在高性能网络环境下，传统的用户态加密技术因频繁的上下文切换和数据拷贝导致严重的性能瓶颈，而专用硬件加密网关虽然性能优越，却面临部署灵活性不足和成本高昂等问题。

本研究提出了一种基于 eBPF 与 XDP 框架的内核态 SM4 加密方法，实现了高性能、协议无关的透明加密传输。研究采用内核态无拷贝架构设计，将加密逻辑下沉至网卡驱动层，有效避免了用户态与内核态之间的频繁上下文切换与数据拷贝开销。针对 eBPF 虚拟机的严格限制，本研究通过循环展开、S 盒静态映射等创新方法，成功解决了 SM4 算法在指令数限制、栈空间受限等受限环境中的实现难题；同时设计了基于 BPF Map 的高效密钥分发与更新机制，确保加密操作的安全性与动态性。

实验结果表明，与传统用户态加密方案相比，本系统在往返时延上平均降低 30%~45%，在吞吐量上平均提升 10 倍以上，尤其在大数据包场景下性能优势更为显著。该研究不仅验证了国密算法在内核态高性能场景下的可行性，更为构建自主可控的网络安全基础设施提供了技术支持，对推动国产密码技术在高性能安全通信领域的应用具有重要意义。

关键词：eBPF；XDP；SM4 加密；内核态；零拷贝

ABSTRACT

In the realm of digital era, cybersecurity threats are becoming increasingly complex and diverse. In high-performance network environments, traditional user-space encryption techniques cause significant performance bottlenecks due to frequent context switching and data copying, while dedicated hardware encryption gateways, despite their superior performance, face challenges such as limited deployment flexibility and high costs.

This research proposes a kernel-space SM4 encryption method based on the eBPF and XDP frameworks, achieving high-performance, protocol-agnostic transparent encryption transmission. The study adopts a kernel-space zero-copy architecture design that pushes encryption logic down to the network driver layer, effectively avoiding frequent context switching and data copying overhead between user and kernel spaces. To address the strict limitations of the eBPF virtual machine, this research successfully overcomes the implementation challenges of the SM4 algorithm in constrained environments with instruction count limitations and limited stack space through innovative methods such as loop unrolling and static S-box mapping. Meanwhile, an efficient key distribution and update mechanism based on BPF Maps is designed to ensure the security and dynamism of encryption operations.

Experimental results demonstrate that compared to traditional user-space encryption solutions, this system reduces round-trip latency by an average of 30%~45% and increases throughput by an average of 10 times, with even more significant performance advantages in large packet scenarios. This research not only validates the feasibility of national cryptographic algorithms in high-performance kernel-space scenarios but also provides technical support for building autonomous and controllable network security infrastructure, holding significant implications for advancing the application of domestic cryptographic technology in high-performance secure communications.

Keywords: BPF; XDP; SM4 Encryption; Kernel Space; Zero-Copy

目录

第一章 概述

1.1 研究背景

本小节介绍本研究的研究背景。通过数字化时代网络安全的挑战引出数据加密的重要性，再结合国家法需求强调国产加密算法的战略价值，最后讨论 eBPF 下的内核加密新范式。

1.1.1 数字化时代的网络安全挑战

随着 5G、云计算、物联网等新一代数字技术的快速发展普及，数字基础设施正在进行广泛重构。这些技术的应用虽然推动了数字化进程，但伴随而来的网络安全风险也愈加复杂和频繁。从个体层面的个人隐私保护，到企业层面的商业机密维护；从政府层面的数据安全管理，到金融交易领域的风险防控，几乎每一个环节都暴露于潜在网络安全威胁之下。据 IBM《2024 年数据泄露成本报告》显示，全球每起数据泄露事件造成的损失已高达到 488 万美元[1]；与此同时，Cloudflare 记录的 2024 年 DDoS 攻击峰值流量更是突破了 5.6Tbps，创下历史新高[2]；而新型高级持续性威胁（Advanced Persistent Threat，APT）攻击也正将攻击目标对准金融、能源等关键基础设施[3]。值得注意的是，攻击者正利用 AI 技术实现自动化漏洞挖掘与攻击路径规划，使得传统基于规则库的防御手段逐渐失效[4]，数据安全面临“加密泛化”与“威胁升级”的双重挑战。

我国网络环境同样呈现快速发展与高压并存的态势。第 55 次《中国互连网络发展状况统计报告》指出：截至 2024 年 12 月，我国网民规模达 11.08 亿，互联网普及率高达 78.6%，兼顾发展与安全成为政策制定的核心方向[5]。中国国务院颁布的《网络数据安全管理条例》明确指出“保障网络数据安全，维护国家安全和公共利益”，强调构建自主可控网络安全体系的紧迫性和重要性[6]。中国工业和信息化部制定的《工业和信息化领域数据安全管理办法（试行）》对工业和信息化领域的数据存储和传输提出了明确的加密规范要求[7]。

1.1.2 国产加密算法的战略价值

数据加密技术作为在链路传输过程中保障数据生命周期安全的核心技术，其战略意义愈发凸显。该技术不仅具备防范恶意程序窃取敏感信息的强大能力，而且能够满足金融、政务等关键领域对于数据传输实时加密的严格要求[8]。数据加密技术通过对数据进行加密处理，确保数据在传输过程中的机密性和完整性，有效防止数据被窃取或篡改。此外，加密算法在身份验证、数字签名等方面的应用，也为防止未授权访问和数据伪造提供了有力保障[9]。然而，加密技术的广泛部署也引发了新的挑战：据 Google Transparency Report，2024 年 HTTPS 流量占比已超过 95%，加密流量占比的上升导致传统基于深度包检测（Deep Packet Inspection，DPI）的安全审计手段失效[10]，亟需结合加密算法与协议栈深度集成的透明化处理方案。

网络数据加密的实践需协同保障安全性（Security）、强化可控性（Controllability），并优化效率性（Efficiency），三者共同构成数字主权时代的核心密码学范式[11]。加密数据的安全性是保障个人隐私、企业机密甚至国家安全的基础。一旦加密数据被破解，导致敏感信息泄露，将引发严重信任危机，造成不可估量的损失。此外，提升加密算法的可控性同样重要。掌握自主可控的加密算法，能避免因外部算法潜在风险导致的国家信息安全漏洞，使中国在国际网络空间中拥有更高自主权和话语权，保障国家信息安全。另外，降低加密带来的额外开销，关系到国家安全措施的可持续性。加密效率的优化是技术落

地的“最后一公里”，合理的资源利用和成本控制，有利于促使加密技术推广应用于各领域。

为了应对这些挑战，选用一种合适的加密方法尤为重要。当前广泛应用的国际通用加密算法，虽然满足数据安全的需求，但是在设计之初未充分考虑特定国家的安全需求。以 RSA 公钥加密算法为例，它通过复杂的数学变换和密钥操作，将数据转化为高度混乱的密文，从而极大地提升了破解难度。然而，正是这种依赖于特定数学原理的设计，使得其在某些情况下可能缺乏可控性，甚至存在潜在的安全风险或后门隐患[12] 。2023 年 NIST 标准中的 CRYSTALS-Kyber 后量子算法因潜在侧信道漏洞引发争议[13] ，凸显了算法可控性的战略价值。鉴于国际通用加密算法存在上述局限性，特别是其难以与中国安全战略实现深度融合，国产加密算法成为了解决问题的关键。中国自主研发的 SM4 加密算法属于分组对称加密算法范畴，具备加解密效率高、算法实现简易等显著优势[14] ，不仅能够有效满足数据安全的基本需求，而且能够深度契合国家网络空间安全战略的要求，为中国构建坚实可靠的网络安全防护体系提供有力的技术支持与保障。特别是作为中国首个国际标准对称加密算法，SM4 加密算法凭借其高安全性和良好的国产化适配优势，已广泛应用于政务云、5G 通信等关键领域[15] 。

1.1.3 eBPF 技术赋能内核态加密

传统的网络安全防护机制，以及现有的基于用户态的加密技术，在应对海量数据处理与实时流量过滤等复杂场景时，正面临性能瓶颈与延迟过高的双重挑战。传统的网络安全防护手段，如防火墙、入侵检测系统（Intrusion Detection System，IDS）、入侵防御系统（Intrusion Prevention System，IPS）等，虽能在一定程度上保护网络安全，但在面对日益复杂且隐蔽的网络攻击时存在明显的功能局限性[16] 。具体而言，防火墙依赖静态规则集以过滤网络流量，难以应对新型攻击，而且由于只能处理明文流量，无法有效处理加密流量；IDS/IPS 系统则容易产生误报，很难确保数据的安全性。现有的基于用户态加密技术，通过软件层来实现加解密功能，其性能具有局限性。这种方式在面对大数据量和高并发流量时，会因用户态与内核态之间的频繁上下文切换而导致显著的性能开销[17] ，难以满足现代网络应用对高性能、低延迟的严格要求。

为突破用户态-内核态上下文切换的性能损耗，近年来基于内核层的高性能框架成为研究热点，其中，扩展伯克利数据包过滤器（extended Berkeley Packet Filter，eBPF）技术凭借其沙箱化、可编程、零拷贝的特性，允许开发者在操作系统内核中动态注入安全逻辑。eBPF 技术允许开发者通过编写小型程序来扩展内核功能，而无需修改内核源代码或重启系统，具有极高的灵活性和[18] 。通过在内核层面直接处理数据包，eBPF 技术能够显著提高网络吞吐量并降低延迟，尤其是其支持的 XDP（eXpress Data Path）框架，通过在网卡驱动层直接处理数据包，可实现纳秒级响应与线速过滤，为构建高性能网络安全防线提供了全新范式[19] 。Cilium 项目利用 eBPF/XDP 实现容器网络的微隔离策略，将策略执行延迟从毫秒级降至微秒级[20] ，证明了该技术在实时安全防护中的潜力。然而，现有 eBPF 应用多聚焦于流量过滤，尚未解决加密算法在内核态的资源受限、校验严格等核心问题[21] 。

1.2 研究意义

本小节介绍本研究的研究意义。本研究在理论层面构建透明加密新范式，在技术层面解决 eBPF 虚拟机环境下的算法适配问题，在国家层面支撑自主可控网络安全体系的底层技术自动化。

1.2.1 构建协议无关的透明加密范式

传统网络加密技术通常遵循“用户态协议栈处理+加密库调用”的分层架构，导致数据在内核态与

用户态之间频繁拷贝，加解密操作与网络协议栈深度耦合，难以实现端到端性能优化。本研究通过 eBPF/XDP 框架与 SM4 算法的协同设计，提出一种“加密逻辑下沉至网卡驱动层”的新型架构，从理论上验证了内核态零拷贝加密的可行性，为构建“协议无关、算法可插拔、零拷贝处理”的透明化加密体系提供了新的设计范式。

1.2.2 解决 eBPF 虚拟机环境下的算法适配问题

虽然利用 eBPF 技术能弥补加密带来的性能开销，但传统的 eBPF 应用主要集中在网络数据包处理、性能优化和内核监控方面，较少涉及到加密算法的应用和优化。如果将 SM4 等加密算法直接在 eBPF 中实现，不仅会面临资源、循环与指令集受限等技术限制，还需要应对严格的验证器规则以及缺乏成熟经验等多重考验。本研究通过静态循环展开（Loop Unrolling）、查找表优化以及位运算等方法，首次在 eBPF 受限环境中完整实现 SM4 算法，解决了算法逻辑与执行环境适配性的核心技术挑战。此外，通过 XDP 层零拷贝数据平面与 SM4 结合，可显著降低加解密延迟，为高性能场景下的国密算法部署提供了可复用的优化路径。

1.2.3 支撑自主可控网络安全体系的底层技术自主化

当前国密算法的实现多依赖用户态软件库或专用硬件加速卡，存在供应链风险与部署灵活性不足的缺陷。本研究通过将 SM4 算法深度集成至 Linux 内核网络协议栈，实现了“通用 CPU+开源软件栈”的国密合规化方案，可有效规避外部技术依赖。该成果符合《国家网络空间安全战略》中“推进密码技术全面应用”的政策导向[22]，为金融、政务、工业互联网等关键领域提供了高性能、低成本的国密合规落地实践，助力我国在网络空间主权博弈中掌握技术主动权。

1.3 主要研究内容

本研究提出一种基于 eBPF 的内核态无拷贝加密框架，以解决传统用户态加密框架带来的性能瓶颈问题。本研究设计了 SM4 算法在 eBPF 虚拟机上的适配方法，以达到在受限平台上实现国产加密算法的目的；探索密钥分发机制，以达到加密密钥的实时动态过更新；进行性能评估，以验证加密操作的高效性和安全性。最后，本研究弥补了现有加密技术在性能和安全性上的不足，为加密算法在 eBPF 领域的应用提供了新的思路和方法。

1.3.1 基于 eBPF 的内核态无拷贝加密框架设计

为了提升加密的效率，本研究采用分层解耦的架构，设计了基于 eBPF 的内核态无拷贝加密框架。

数据平面使用 XDP 程序进行数据包的捕获与预处理，XDP 可直接在网络驱动程序的接收路径执行，从而实现对数据流的高效拦截与处理。通过 XDP 的高效性，本系统能在数据到达网络栈之前，完成加密操作。控制平面通过 BPF Map 管理密钥、加密策略、参数配置等信息。密钥在内核空间中进行存储和管理，支持密钥的动态加载和更新。加密引擎中即时编译（Just-In-Time Compilation，JIT）的 SM4 内核模块执行加解密。JIT 编译能够将 SM4 算法动态转化为机器代码，进一步提升执行效率，减少内存消耗。最终通过 BCC 工具链实现程序动态加载，支持热更新而无需重启系统。

该框架的创新点在于，将 eBPF 与加密算法结合，实现内核态的加密处理；使用动态加载的 eBPF 程序，支持灵活的加密策略调整；减少加密过程中的上下文切换，显著提升数据处理速度。

1.3.2 SM4 算法在 eBPF 虚拟机上的适配

由于 eBPF 虚拟机资源受限，因此直接在 eBPF 中实现 SM4 算法存在一定挑战。本研究针对这一问题，提出了 SM4 算法在 eBPF 虚拟机上的优化适配方法，以保证其高效、稳定运行。

在算法结构方面，对 SM4 算法进行了精简和优化，减少了冗余计算与内存使用。使用更高效的内存访问模式和减少不必要的算法迭代，以降低内存占用和执行时延。在代码优化方面，通过改进 SM4 的循环结构，降低其复杂度，减少 eBPF 虚拟机中每个指令的执行时间，使得加密操作能够在 eBPF 的执行限制下高效完成。另外，采用内存预分配来优化内存管理，避免频繁的内存分配和释放操作，提高程序的内存使用效率，降低内存碎片。通过 eBPF 的钩子机制，优化后的 SM4 算法在数据包流经内核栈时能够实时加密，确保数据的即时保护。

1.3.3 密钥分发与更新

在现代加密系统中，密钥的分发至关重要，尤其是在高效加密系统中，密钥的分发与更新机制需要保证其动态性与安全性。本研究针对基于 eBPF 的内核态加密框架，设计了高效的密钥分发与更新机制，确保加密操作中密钥的安全性和实时更新。

密钥通过 TCP 连接从密钥管理系统（Key Management System，KMS）发送至客户端。客户端通过指定的服务器 IP 和端口连接 KMS，接收当前有效的密钥数据。密钥包括 SM4 加密所需的密钥本身、密钥 ID、生成时间戳以及过期时间戳等。KMS 负责根据预定的周期更新密钥。每当密钥接近过期时，KMS 会生成新的密钥并将其分发给所有已连接的客户端。客户端收到新密钥后，立即将其更新到 BPF Map 中，以确保加密操作使用的是最新的密钥。通过 eBPF 的灵活性，密钥更新无需重启系统或重载内核模块。密钥更新通过内核空间中的 BPF Map 动态加载，无需中断服务，从而保证了系统的高可用性。

1.3.4 性能评估与优化

在内核态实现加密操作后，需要对其性能进行评估，并探索相应的优化策略。本研究将设计一种综合性能评估体系，针对加密操作的吞吐量、延迟进行测量与分析。基于评估结果，提出未来优化策略。

1.4 论文的组织结构

本论文共分为六章，对基于 XDP 的 SM4 加密方法进行了系统性研究与实现：

第一章为概述，介绍了研究背景、意义及主要内容。阐述了数字化时代网络安全挑战、国产加密算法 SM4 的战略价值以及 eBPF 技术在内核态加密中的应用潜力，明确了本研究旨在构建协议无关的透明加密范式，解决 eBPF 环境下的算法适配问题，支撑自主可控网络安全底层技术。

第二章为相关技术背景及研究现状，系统回顾链路传输技术（XDP、DPDK、RDMA）与数据加密技术（SM4、AES、RSA、SM2）相关知识，分析了硬件加密网关与 VPN 技术的优缺点，明确本研究相较现有方法的创新点与技术突破方向。

第三章为系统设计，全面描述本研究提出的内核态 SM4 加密系统架构与关键模块设计。包括数据包捕获与解析、加密与解密逻辑、密钥分发机制及转发处理流程，强调 eBPF/XDP 如何在内核态高效完成数据加解密并实现透明链路传输。。

第四章为挑战与应对措施，针对在 eBPF 虚拟机中实现 SM4 加密算法面临的技术限制（如指令数限制、栈空间受限、Verifier 验证机制等），逐一提出对应的优化策略与工程实践，包括循环展开、S 盒映射、

轮密钥预计算与协议逻辑重构等。

第五章为系统测试，通过密钥分发、前端功能和性能评估等测试，验证了系统功能的完整性和性能优势。结果表明 XDP 模式在各种负载下均表现出显著优势，尤其在大数据包场景下，吞吐量平均提升可达 10 倍以上，往返时延最多减少 45%，印证了本研究方案的技术先进性。

第六章为总结与展望，归纳了本研究的主要成果，并针对未来可能的研究方向提出了展望，包括硬件加速与异构计算、多线程扩展、状态管理优化以及密钥体系完善等方面，为后续研究指明了方向。

第二章 背景及国内外研究现状

2.1 背景知识

2.1.1 链路传输技术

现代网络架构中，链路传输技术作为数据通信的核心支撑，其性能直接影响着网络系统的吞吐量和响应效率。随着互联网流量的激增和数据中心需求的提升，传统的链路传输技术已无法满足高带宽、低延迟和可扩展性的需求。在此背景下，XDP、DPDK 与 RDMA 三种新型传输技术凭借其创新架构，成为构建高性能网络的关键解决方案。

（1）快速数据通路技术

快速数据通路技术（eXpress Data Path，XDP）作为 Linux 内核原生支持的高性能网络数据处理框架，通过 eBPF 虚拟机在网卡驱动层实现数据包处理逻辑的灵活注入。该技术将数据处理点前移至网络协议栈的最底层——即网卡接收数据包后、进入内核协议栈前的阶段，使得数据包可在抵达内核前完成过滤、转发或修改操作。例如，在 DDoS 攻击防御场景中，XDP 程序能够以纳秒级延迟实时识别并丢弃恶意流量，而无需触发内核中断或内存拷贝。XDP 提供三种运行模式：原生模式（依赖网卡驱动支持）、卸载模式（利用智能网卡硬件加速）和通用模式（软件模拟实现），其中原生模式可实现接近线速的处理性能。由于其高性能和灵活性，XDP 被广泛应用于需要快速网络响应的场景，如流量分析和负载均衡等[23] 。

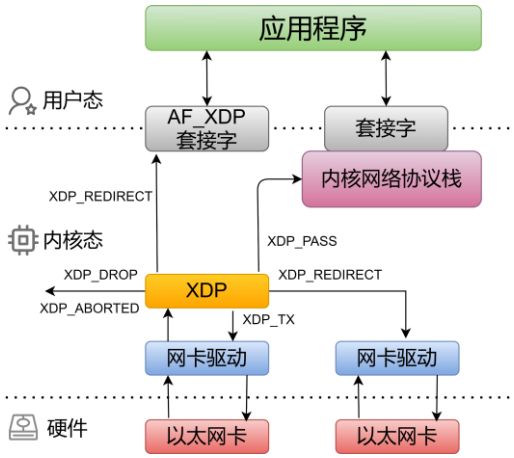

图 2-1 XDP 架构示意图

XDP 的技术优势体现在与 Linux 内核的深度协同，具有极低的延迟和高吞吐率，且不需要额外的硬件支持。它通过 AF_XDP 协议族实现用户态与内核态的无缝交互，允许将处理后的数据包直接重定向到用户态内存队列（User-space Memory Queue，UMEM），既保留了内核协议栈的完整功能，又实现了用户态程序对网络数据的零拷贝访问。例如，Cloudflare 利用 XDP 技术构建了 L4Drop 防御系统，通过动态加载 eBPF 程序在驱动层直接丢弃 DDoS 攻击流量，实现了每秒处理千万级数据包的能力，且 CPU 占用率极低[24] 。然而，XDP 的适用性受限于网卡驱动支持度，部分老旧设备仅支持通用模式 [25] 。此外，对于复杂协议的处理，编写高效的 BPF 程序具有一定难度[26] 。

（2）Data Plane Development Kit（DPDK）

DPDK 作为由 Intel 主导开发的用户态数据平面开发套件，其核心目标是通过绕过内核协议栈、零拷贝技术和轮询模式驱动实现网络数据的高吞吐处理[27]。DPDK 的架构基于用户态 I/O（User-space Input/Output，UIO）机制，允许应用程序直接访问网卡硬件寄存器，并通过大页内存优化内存管理，减少虚拟地址转换开销。例如，在 10Gbps 网络环境下，对于 64 字节小包，DPDK 的单核处理能力可达 2000 万包/秒，较传统内核协议栈提升 20 倍[28]。

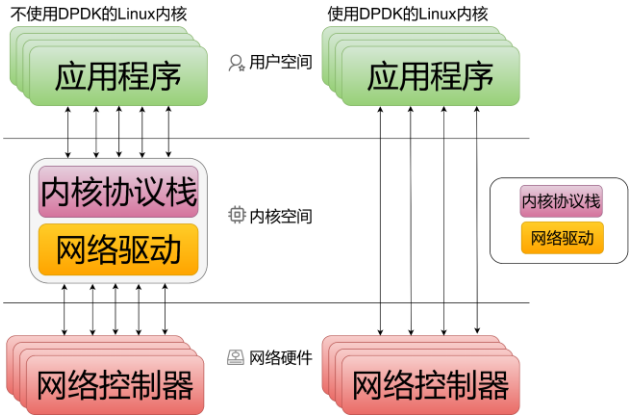

图 2-2 传统内核网络处理与 DPDK 用户态加速架构对比示意图

DPDK 的核心优势在于用户空间编程，具有极低的延迟和高吞吐量，且支持多种硬件和网络接口卡（Network Interface Card，NIC）。DPDK 的主要应用场景包括网络功能虚拟化、软件定义网络（Software-Defined Networking，SDN）和云计算数据中心。例如，Open vSwitch with DPDK（OVS-DPDK）通过用户态协议栈替代传统内核虚拟交换机，将转发时延从微秒级降至纳秒级，提升了虚拟化网络的性能[29]。

DPDK 可以充分发挥现代多核 CPU 的优势，支持多线程和大规模数据包的并行处理。然而，其局限性同样明显：用户态协议栈与内核生态完全割裂，导致 iptables、tcpdump 等工具无法直接使用；持续轮询机制造成 CPU 空转，在负载波动场景中能效比较低；对专有硬件和操作系统化境有较高要求[30]。

（3）Remote Direct Memory Access（RDMA）

RDMA 通过网卡硬件卸载实现跨节点内存直接访问，彻底消除数据在用户态与内核态间的复制开销[31]，其核心创新在于将传输协议栈（如流量控制、重传机制）下沉至网卡硬件。RDMA 通过消除数据在用户态与内核态之间的复制操作，将端到端通信延迟降至亚微秒级，并支持高达 200Gbps 的带宽。RDMA 的三种主流实现协议包括 InfiniBand、RDMA over Converged Ethernet（RoCE）和 iWARP，其中 RoCEv2 基于 UDP/IP 协议栈，兼容标准以太网设备，成为当前数据中心部署的主流方案[32]。

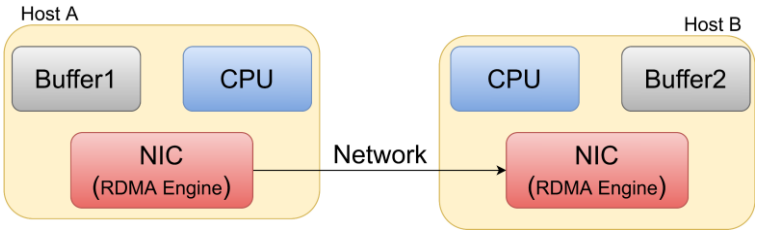

图 2-3 RDMA 零拷贝跨主机直接内存访问流程示意图

在应用层面，RDMA 被广泛用于分布式存储和高性能计算（High-Performance Computing，HPC）。例如，阿里云的 PolarFS 分布式文件系统通过 RDMA 实现了存储节点间的内存直接访问，将 I/O 延迟从毫秒级降低至微秒级[33]。此外，RDMA 还被用于 AI 训练中的 GPU 间通信，如 NVIDIA 的 GPUDirect RDMA 通过绕过 CPU 直接传输显存数据，显著提升了训练效率[34]。然而，RDMA 的部署复杂度较高，需依赖无损网络及专用硬件支持，且与普通以太网设备的兼容性有限，使得 RDMA 在普遍的 IP 网络环境中的应用受到限制。

尽管 XDP、DPDK 和 RDMA 都提供了高性能的数据传输能力，但它们的应用场景和适用条件各不相同。一方面，XDP 由于与 Linux 内核的紧密集成，能够兼顾灵活性与性能，尤其适用于需要灵活定制的网络应用（如流量过滤、负载均衡等）。DPDK 和 RDMA 分别在用户空间和远程内存访问方面提供了更为专业化的解决方案，其中 DPDK 适合用于大规模的并行数据包处理，而 RDMA 更加适用于分布式存储系统和高性能计算环境。另一方面 DPDK 全部在用户空间运行，需要特定的驱动支持，且难以直接利用内核生态，灵活性有限。RDMA 适用于分布式存储等场景，在一般 IP 网络环境中，部署 RDMA 需要专用硬件支持，且与常规以太网兼容性差，应用场景有限。XDP 能够与内核资源无缝协作，且在多种网络环境中具有较强的通用性，这使得它成为本研究中链路传输技术的最佳选择。

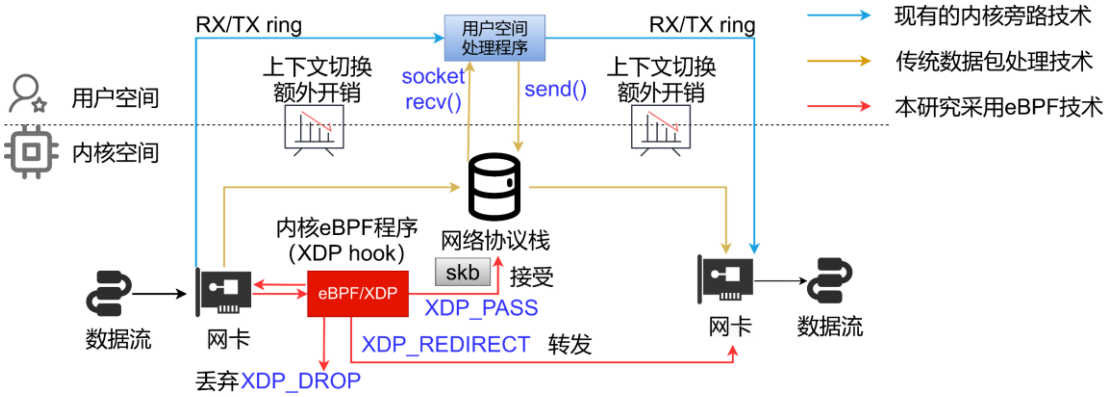

图 2-4 XDP、DPDK、用户态下上下文切换额外开销示意图

2.1.2 数据加密技术

数据加密技术是保障信息在存储和传输过程中机密性、完整性和不可抵赖性的核心手段。本研究主要关注对称加密和非对称加密两大类算法。

（1）对称加密算法

对称加密算法使用同一密钥完成加密和解密，因其计算开销相对较小，常用于实时通信场景。

Advanced Encryption Standard（AES）是由美国国家标准与技术研究院（National Institute of Standards and Technology，NIST）于 2001 年发布的对称加密标准[35]，旨在替代原有 Data Encryption Standard（DES）算法。AES 采用分组加密方式，分组长度为 128 位，支持 128、192 和 256 位的密钥长度，分别对应 10、12、14 轮加密迭代。其结构基于代换-置换网络（Substitution-Permutation Network，SPN），通过多个加密轮次增强数据的混淆性和扩散性，有效抵御差分攻击和线性分析。AES 具有高效性和强大的安全性，广泛应用于政府、金融、通信等多个领域，成为现代加密技术的基石之一[36]。

在算法实现层面，每轮加密包含四个关键操作：字节代换（SubBytes）通过 16×16 的 S 盒完成非线

性字节替换，打破明文统计特性；行移位（ShiftRows）对状态矩阵的每行进行循环左移，增强行间扩散；列混淆（MixColumns）采用有限域 GF（2^8）上的矩阵乘法，实现列内字节的线性混合；轮密钥加（AddRoundKey）则将当前轮次子密钥与状态矩阵异或，注入密钥相关性。以 AES-128 为例，密钥扩展算法通过 Rijndael 密钥调度生成 11 组 128 位子密钥，每组密钥由前一组经字节循环移位、S 盒代换和轮常数异或产生，确保密钥材料的不可预测性。

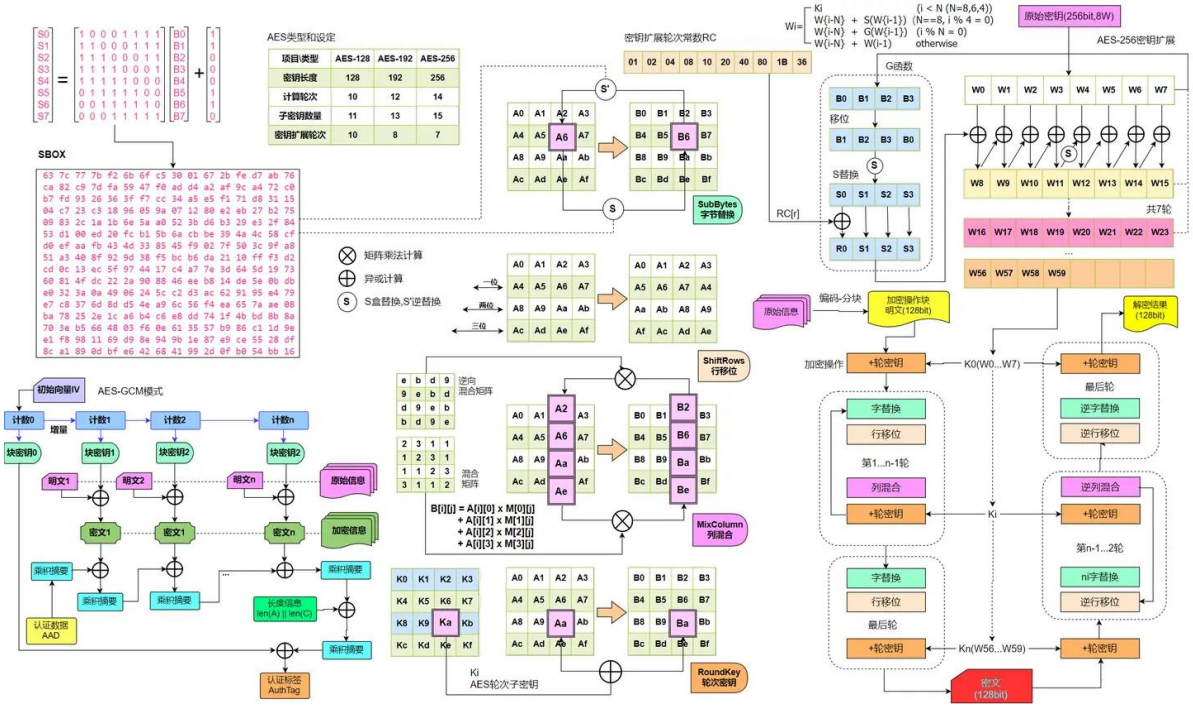

图 2-5 AES 算法全景图

图来源：稀土掘金社区

硬件加速技术显著提升了 AES 的实际性能。Intel AES-NI 指令集通过专用硬件电路实现轮函数并行化，在现代处理器上能够实现数十 Gbps 的吞吐量，满足大规模数据加密场景的实时性需求[37] 。然而，AES 也存在一定的局限性。首先，其对侧信道攻击（如功耗分析、缓存时序分析）较为敏感[38] ，特别是在硬件实现不完善或缺乏恒定时间编程策略的环境中，可能导致密钥泄露风险。其次，在后量子密码学背景下，尽管 AES 并未直接受到量子算法如 Shor 算法的影响，但 Grover 算法对对称密钥系统提出了挑战，其搜索复杂度下降一半，理论上降低了 AES-128 的安全强度[38] 。因此，NIST 推荐在对抗量子计算风险时优先采用 AES-256，以维持 128 位的量子安全强度。

SM4 是中国国家密码管理局于 2012 年发布、2016 年正式成为国家标准（GB/T 32907-2016）的商用对称加密算法[40] ，是我国密码自主化战略的核心技术之一。作为分组密码算法，SM4 采用 128 位分组长度和 128 位密钥长度，通过 32 轮非平衡 Feistel 网络结构实现数据加密。其设计目标是在保证安全性的前提下，适配国产硬件平台的高效实现需求。

SM4 算法核心流程中，每轮加密通过轮函数 F 完成非线性变换，该函数由 S 盒代换与线性变换 L 组合而成。S 盒采用固定的 8 位输入/输出映射表，通过有限域运算实现非线性混淆；线性变换 L 则通过循环左移和异或操作增强扩散性。密钥扩展算法通过逆向 Feistel 结构生成 32 轮子密钥，进一步增强

密钥材料的不可预测性。

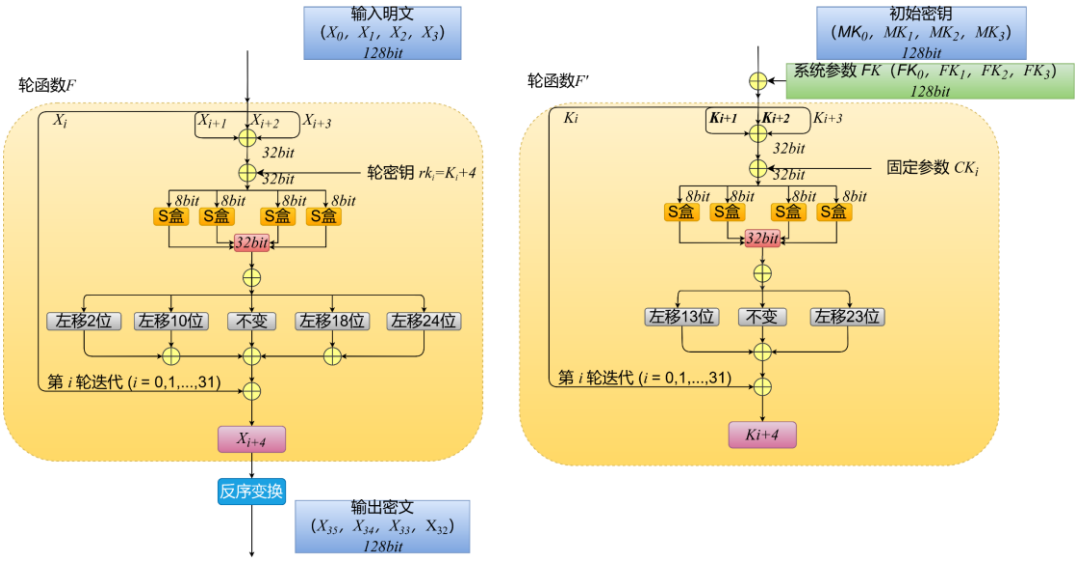

图 2-6 SM4 核心算法图

在安全性层面，SM4 历经多轮国家密码管理局的严格审查，可抵御差分攻击、线性攻击及侧信道攻击。中国密码学会 2015 年发布的《SM4 分组密码算法安全性分析报告》指出，其全轮差分概率低于 2^{-128}，线性壳概率低于 2^{-114}，理论破解需超过 2^{100} 次操作，安全强度与 AES-128 相当[41] 。为应对侧信道威胁，SM4 支持掩码技术和随机化执行路径，符合《GM/T 0062-2018 侧信道攻击检测规范》要求。在国产飞腾 FT-2000/4 处理器上，SM4-CTR 模式的软件优化实现可达 3.5 Gbps 吞吐量，而搭载专用指令集的龙芯 3A5000 处理器可进一步提升至 10 Gbps 以上，显著优于无硬件加速的 AES 实现[42] 。尽管如此，SM4 的国际化应用仍面临挑战，需通过协议转换网关与 AES 生态互通，且非国产硬件平台（如 Intel/AMD）缺乏原生指令支持，导致软件性能受限。

（2）非对称加密算法

非对称加密使用公钥加密、私钥解密，解决了对称加密的密钥分发难题，主要用于密钥交换、数字签名和身份认证等场景。

RSA 是由 Ron Rivest、Adi Shamir 和 Leonard Adleman 于 1977 年提出的非对称加密算法[43] ，其安全性基于大整数分解难题——即对于两个大素数 p 和 q 的乘积 $n = p \times q$，在未知 p 和 q 的情况下，计算 n 的素因子在计算上不可行。RSA 的核心流程包括密钥生成、加密与解密三个阶段。密钥生成阶段需选择两个大素数 p 和 q，计算模数 $n = p \times q$ 和欧拉函数 $\phi(n) = (p-1)(q-1)$，随后选取与 $\phi(n)$ 互质的公钥指数 e，并计算私钥指数 d 满足 $e \times d \equiv 1 \, mod \, \phi(n)$。公钥为 (n, e)，私钥为 (n, d)。加密时，明文 m 被转换为密文 $c = m^e \, mod \, n$，解密则通过 $m = c^d \, mod \, n$ 恢复原文。其数学基础依赖欧拉定理，确保 $m^{e \times d} \equiv m \, mod \, n$ 成立。

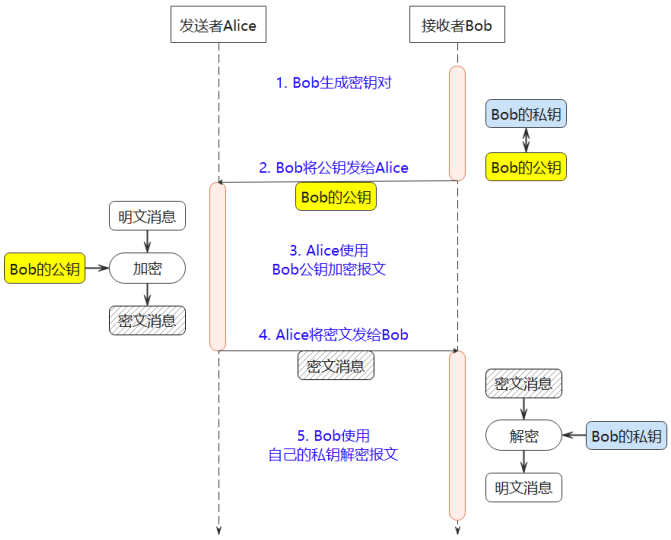

图 2-7 RSA 交互流程图

在实际部署中，RSA 的密钥长度直接影响安全性。NIST 建议当前使用 2048 位及以上密钥，而 3072 位密钥可抵御未来十年内的算力增长[44] 。RSA 的典型应用涵盖数字签名、密钥交换与身份认证。在 TLS 协议中，RSA 用于握手阶段的密钥封装和证书签名；PGP 电子邮件加密依赖 RSA 实现端到端数据保密；区块链领域，比特币早期地址生成基于 RSA 衍生算法。然而，RSA 面临量子计算威胁——Shor 算法可在多项式时间内破解大数分解问题，2048 位 RSA 在量子计算机下仅需数小时即可攻破[45] 。因此，NIST 在后量子密码迁移路线中建议逐步淘汰 RSA-2048。

SM2 是中国国家密码管理局于 2010 年发布、2016 年成为国家标准（GB/T 32918-2016）的椭圆曲线公钥密码算法[46] ，是我国商用密码体系的核心非对称加密标准。其设计基于椭圆曲线离散对数问题（Elliptic Curve Discrete Logarithm Problem，ECDLP），采用国密标准化的椭圆曲线参数，包括素数域F_p的椭圆曲线方程 $y^2 = x^3 + ax + b$，其中 p 为 256 位大素数，基点 G 的阶 n 为 256 位素数，确保算法在有限域上的计算安全性。SM2 支持公钥加密、数字签名与密钥交换三大功能，其加密流程采用基于椭圆曲线的集成加密方案（Elliptic Curve-Based Integrated Encryption Scheme，ECIES），通过密钥派生函数（Key Derivation Function，KDF）和消息认证码（Message Authentication Code，MAC）实现数据机密性与完整性保护。数字签名机制基于 SM3 哈希算法与椭圆曲线运算，签名过程生成 (r, s) 对，验证时通过重构方程确认签名的合法性，可抵御伪造攻击与重放攻击。

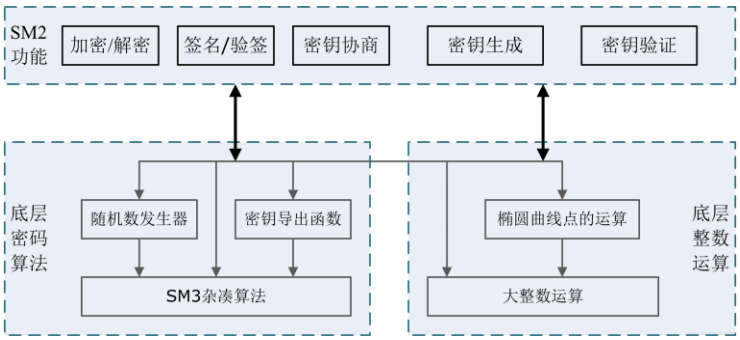

图 2-8 SM2 整体结构图

综上所述，对称加密算法因其高效性和低计算开销，通常适用于大规模数据加密或实时通信场景，在构建高速加密链路时具有显著优势；而非对称加密算法则凭借其密钥分发的便捷性和良好的安全特性，广泛应用于密钥交换、数字签名与身份认证等关键环节，是现代公钥密码体系的核心技术。本研究在整体架构中采用国密 SM4 算法作为链路加密的对称加密方案，兼顾了安全性与性能的平衡，契合我国自主可控的密码体系发展方向。

2.2 国内外研究现状及其存在的问题

当前国内外在链路加密传输领域已有丰富的研究，常见的两类方案是硬件加密网关（Hardware Encryption Gateway，HEG）和虚拟专用网络（Virtual Private Network，VPN）技术，但各有优劣。

2.2.1 硬件加密网关

硬件加密网关通常指使用专用硬件设备来执行加密操作的网络设备，例如专用加密通信网关、加密交换机或路由器。它们可以集成加速芯片（如 ASIC 或 FPGA），在物理层或数据链路层提供快速的加密处理。采用硬件加速的优点在于加密吞吐量高，可以达到数十 Gbps 级别，适合大流量场景[47]。

现代硬件加密方案通过建立多条加密隧道，将并行加密任务分散到多个 CPU 或加速核，实现多核并行处理，从而克服了单通道加密的瓶颈。例如，Aviatrix 的“高性能加密模式”通过在两个网关之间建立多条 IPsec 隧道，使各个 CPU 核心都能执行加密任务，大幅提升了整体吞吐量[48]。使用硬件加密网关时，多个 CPU 核心协作加速加密处理，使得 IPsec 加密性能可以轻松突破 10Gbps 甚至 25Gbps 以上。

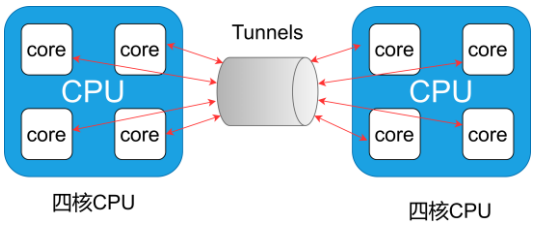

图 2-9 硬件网关多加密隧道示意图

尽管硬件加密网关性能强劲，但其缺点也很明显：首先，此类设备价格昂贵，部署成本较高；其次，它依赖于特定硬件和固定算法，灵活性差，难以随着需求变化进行动态扩展。再者，当网络拓扑或安全策略变化时，硬件网关的升级和维护复杂度较高。此外，多隧道的加速方式需要维护大量的安全关联和状态，对网关的控制面也提出更高要求。在链路加密传输系统中，如果采用纯硬件网关方案，虽然可以保障高吞吐量，但通常需要牺牲部署灵活性和成本[49]，从长远来看难以满足快速迭代的需求。

2.2.2 虚拟专用网络技术

虚拟专用网络通过在公用网络上建立逻辑上的加密隧道，为分支机构或远程用户提供安全访问[50]。典型实现包括互联网协议安全（Internet Protocol Security，IPsec ）VPN 和安全套接层（Secure Sockets Layer，SSL）VPN[51]。VPN 的优点在于部署灵活、易于管理、支持跨越公网和多种接入方式。IPsec VPN 作为国际通用方案，通过隧道模式加密整个 IP 数据报，提供了较高的安全性；SSL VPN 则通常在

传输层或应用层提供隧道加密，便于穿透防火墙和互联网环境。

然而，VPN 方案也存在明显不足。首先，VPN 本质上是面向连接的隧道，需要在通信双方维护完整的安全关联，包括密钥和会话状态，这在并发会话较多的场景下管理复杂[52] 。其次，传统 VPN 加密会引入较大的性能开销：例如对较小数据包而言，IPsec VPN 头部加密开销非常明显。当有效载荷仅 160 字节时，IPsec 隧道模式会将包大小从 188 字节增至 272 字节，带来约 50%的额外开销[53] ；对于更大的报文，开销虽相对减小，但仍不可忽视。这种额外开销导致实际可用带宽明显下降，尤其在高负载和低延迟应用下效果更为突出。

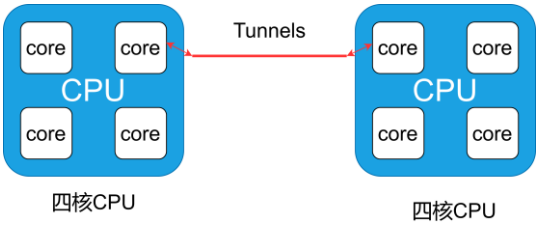

图 2-10 IPsec 隧道模式示意图

图示为单条 IPsec 隧道的典型工作方式。可以看到，若只建立一条隧道，那么流经该隧道的数据只能分配到单个 CPU 核心进行处理，成为吞吐量瓶颈[54] 。即使主机拥有多个核心，但单隧道模式下网络接口只能将数据交付到一个核心，导致性能无法跨核扩展。虽然可以通过多隧道来部分改善，但 VPN 系统本身的复杂度和负载管理开销也随之增加。总之，VPN 技术维护了状态和加密隧道的通用性，但在性能扩展性和成本方面存在显著限制。

综上所述，硬件加密网关虽然具备较高的吞吐能力，但在灵活性和部署成本方面存在一定局限；而 VPN 方案虽便于部署和跨网络访问，但其性能和跨核扩展性相对有限。为了在软件层面设计一种兼顾安全性与性能的数据加密传输机制，本研究提出基于 eBPF/XDP 的实现方案。该方案充分利用 Linux 内核的可编程特性，将 SM4 加密算法集成至网络数据路径中，以实现高性能、安全的数据处理能力。

第三章 系统设计与实现

3.1 系统概览

本系统基于内核态无拷贝加密框架实现了对 UDP 报文的透明化 SM4 加解密。内核态负责加密流与解密流。加密端流程包括数据包捕获、报文解析、密钥接收、加密处理、转发或重定向；解密端流程包括数据包捕获、报文解析、密钥接收、解密处理、转发或重定向。用户态负责密钥管理、策略配置与统计采集，形成完整闭环。该架构实现了“协议无关、算法可插拔、零拷贝处理”的目标，兼顾高性能与灵活可控。

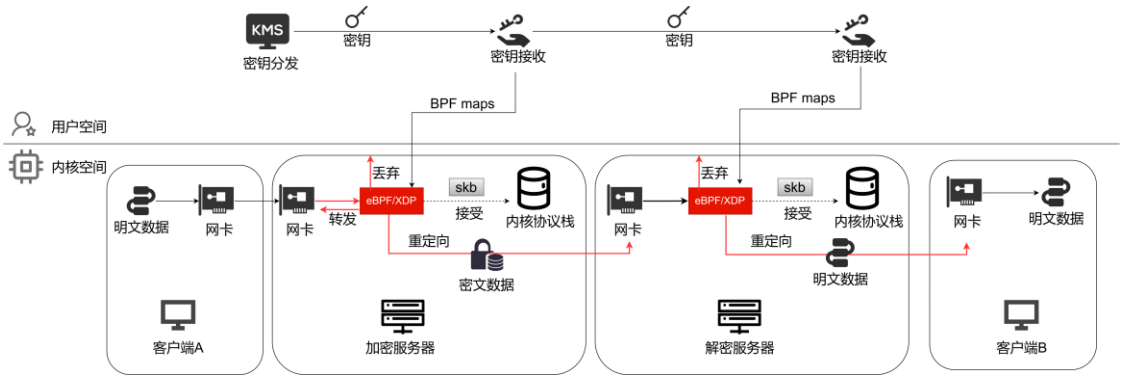

图 3-1 基于 XDP 的内核 SM4 加解密系统流程图

本系统流程如图 3-1 所示：发送明文的客户端 A 与加密服务器在一个安全域，解密服务器与接收端 B 在另一个安全域。A 向 B 发送信息，携带有明文的数据包从发送端通过网络链路传输至加密服务器网卡，网卡驱动中的 XDP 挂钩拦截到达的数据包，并交由 eBPF 程序处理。eBPF 程序首先判断报文类型，对于需加密的明文报文，提取其有效载荷并调用 SM4 加密模块对以 16 字节为单位的数据进行加密，然后将携带有密文的数据包转发到解密服务器网卡。解密服务器对于入站的密文报文，调用 SM4 解密模块回复得到明文，然后将解密后的数据包转发到接收端。最后接受端 B 会收到来自发送端 A 发送的信息。期间加解密服务器用户端定时更新从 KMS 分发的密钥并更新到 BPF Map 中供加解密模块使用。整个过程对 A 和 B 是透明的，只在链路传输的过程中实现对报文的加解密。

3.2 模块设计

3.2.1 数据包发送模块

该模块负责将用户发送的信息格式化后写进 UDP 数据包，然后将明文数据包发送到加密服务器。

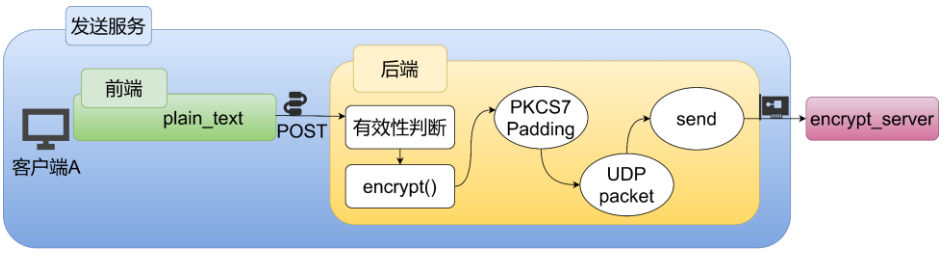

图 3-2 数据包发送模块示意图

（1）明文获取

用户在前端页面的输入框组件中输入待发送信息后，点击发送按钮，触发后端请求。后端接收到前端传来的 JSON 格式的明文内容后，首先需要进行内容有效性验证。如果接收到的内容为空，后端会向前端发送错误提示信息，提醒用户重新输入。若明文内容不为空，系统则继续处理。

（2）PKCS7 填充

为了适应 SM4 加解密模块的要求，后端首先将 JSON 格式的明文转化为 UTF-8 编码。由于 SM4 算法是基于 16 字节（128 位）为一组来加解密的，因此需要对 UTF-8 字节码进行 PKCS7 填充。首先计算明文内容的字节数与 16 字节的最小倍数之间的差值，即缺失的字节数。然后通过填充缺失的字节数，将数据的长度扩展到 16 字节的整数倍。填充字节的内容为缺失字节数的数值，这样可以方便解密时正确去除填充。

（3）UDP 数据包封装

在填充后的明文数据准备好后，系统将其写入 UDP 数据包的 payload 部分。UDP 数据包的封装不仅仅包括加密的负载数据，还需要添加必要的网络协议头信息，确保数据能够正确地发送和接收。具体过程包括填充 UDP 数据包的头部信息，包括源 IP 地址、目的 IP 地址、源端口、目的端口等；填充数据包的 payload 部分，即待加密的明文数据。

（4）UDP 数据包发送

将构建好的 UDP 数据包通过网络发送到加密服务器，进行后续处理。选用 UDP 作为发包载体的原因是因为 UDP 的无连接特性，这使得数据加密传输简单高效，适合大规模数据流。

3.2.2 数据捕获解析模块

该模块部署于系统的网络接入点，并利用 XDP 技术提供的最早期钩子实时拦截所有进出网络的数据包，解析并根据报文的头信息进行初步的处理。当网络接口接收到数据包时，XDP 程序会立即获取该数据包的起始和结束位置，进行基本的边界检查，以确保数据包在有效范围内。接下来，程序会根据数据包的协议类型进行判断，决定是否对该数据包进行进一步处理。

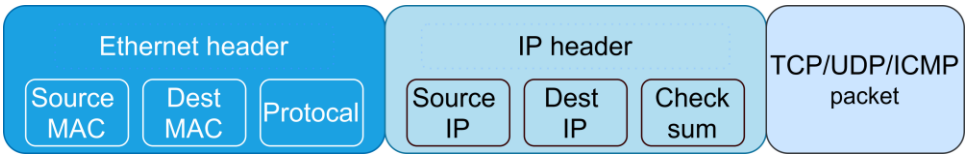

图 3-3 数据包结构示意图

（1）以太网头部解析

该模块首先获取以太网头部的指针，并通过 $ethhdr$ 结构体进行解析，检查是否为有效的以太网帧。判断该数据包的协议类型是否为 $IPv4$（ETH_P_IP），如果不是，则直接跳过该数据包，调用 XDP_PASS，继续接收其他数据包。

（2）IP 头部解析

如果是 IPv4 数据包，程序进一步解析 IP 头部，通过 $iphdr$ 结构体获取源 IP 和目标 IP 等信息。程序会检查 IP 数据包的协议字段，确认该数据包是否为 UDP 协议。如果不是 UDP 协议，则继续跳过该数据包。

（3）UDP 头部解析

如果是 UDP 数据包，程序会解析 UDP 头部，获取源端口和目标端口等信息。然后，程序计算 UDP 负载的长度，并提取负载部分，作为后续加解密操作的输入。

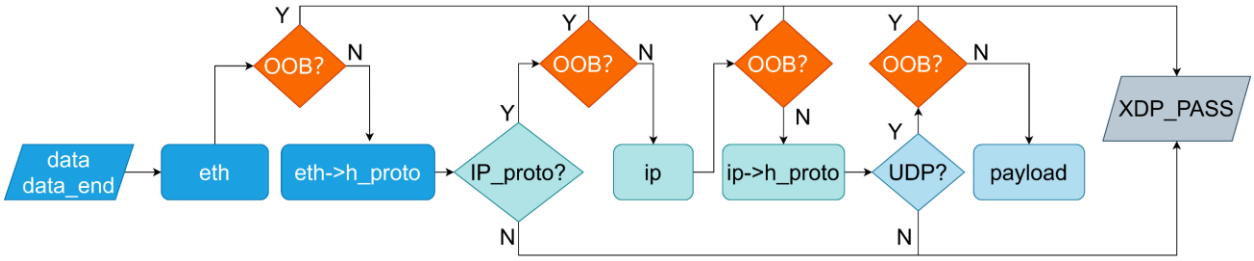

图 3-4 UDP 数据包解析逻辑示意图

（4）特定数据包过滤

在处理过程中，模块会进行特定源地址和端口的过滤。例如，只处理来自某特定源 IP 和特定 UDP 源端口的数据包。其他不符合条件的数据包将会被丢弃（XDP_PASS），不会进一步处理，这样处理可以减少不必要的负载、优化资源使用。

3.2.3 加密处理模块

SM4 是一种对称分组密码算法，它使用 128 位的密钥和 128 位的数据块进行加密。加密过程包括 32 轮的迭代，每轮使用一个轮密钥进行数据转换。SM4 采用的是 Feistel 结构，每一轮的输入数据经过轮函数处理后输出，最终输出的加密结果即为密文。

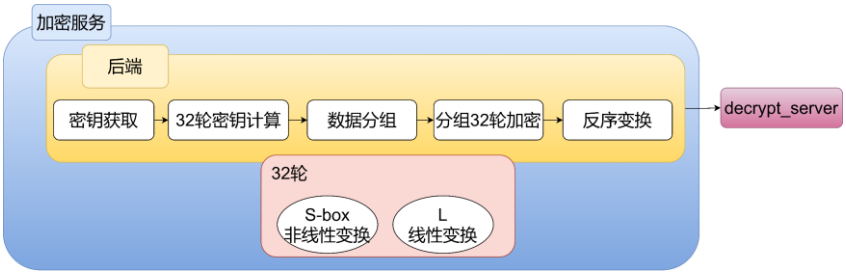

图 3-5 XDP 加密处理模块示意图

（1）算法初始化

从初始密钥中生成 32 个轮密钥（Round Key，RK）。每轮迭代中，数据分为左右两个部分。根据 SM4 的轮函数，数据经过一个置换操作，将数据按照特定规则进行重新排列。通过 SM4 定义的 S-Box 对数据进行替代，增加加密的复杂度，防止简单的暴力破解。通过一系列数学运算，如按位与、异或、加法等操作，使得数据的每个部分在经过多轮加密后都能彼此影响，增强加密的安全性。

（2）密钥获取

SM4 的 128 位密钥存入 BPF Map 中，系统会在初始化时加载这些密钥，在加密操作前，根据需要通过 eBPF 程序访问获取加密密钥。

（3）分组加密

SM4 算法是基于分组加密的原理，即每次处理固定大小的数据块（128 位）。加密过程会对输入的数据流进行分块处理，确保每个数据块的加密独立性。首先将输入数据分割成固定大小的 128 位块，每个数据块作为一个加密单元进行加密。每个 128 位的数据块通过 SM4 算法的 32 轮迭代加密，输出对应的密文块。加密后的密文块将按照顺序拼接在一起，构成最终的加密数据。

（4）密文写回

在内核层，eBPF 程序会将加密后的数据包直接写入到网络接口的发送队列中，或者返回给相应的用户态应用进行后续处理。

3.2.4 转发和重定向模块

在完成加密处理后，网络转发模块负责将报文送出网络或交付到目标主机。加密后的报文需要通过网络接口进行发送，而解密后的报文则需要交由本地主机的网络栈继续处理。

（1）转发处理

若目标为当前接口对应的主机，即直接响应客户端，该模块会采用 XDP_TX 操作，将报文直接回送至当前接收接口，实现高速应答或伪回环通信，确保数据包可以在不经过内核栈的情况下被快速转发至目标地址。

（2）重定向处理

若需要通过网关转发到其他主机，模块将利用 $bpf\_fib\_lookup()$ 进行 FIB 路由查找，获取下一跳接口和目标 MAC 地址。查找成功后，通过 $bpf\_redirect()$ 将报文发送至目标接口，实现跨接口转发。

（3）IP 和 MAC 地址的更新

在数据包转发之前，网络模块可能需要修改目标 IP 地址、端口号和 MAC 地址，将报文中的 $eth->h\_dest$ 和 $eth->h\_source$ 更新为目标/源 MAC 地址。回环测试中，交换源/目的 IP 和 UDP 端口，实现逻辑对称。这保证了报文在网络中传递时，路由和链路层信息是准确的。

（4）校验和更新

由于 IP 和 UDP 报文头可能发生改变，将原有校验和清零，并重新计算 $ip->check$ 和 $udp->check$。

（5）发送数据包

通过不同的返回值，决定了数据包的后续处理行为。当数据包不需要被拦截或特殊处理时，使用 XDP_PASS 将其交给内核进一步处理；当检测到不符合条件的恶意流量或异常数据包时，使用 XDP_DROP 直接丢弃它，避免浪费系统资源或遭遇攻击；如果 XDP 程序在处理过程中遇到无法恢复的错误，比如内存分配失败、访问无效内存等，使用 XDP_ABORTED 来指示出现问题；进行包转发时可以使用 XDP_TX 让数据包通过发送队列离开当前网络接口；当需要对数据包进行流量重定向时（例如，将流量转发到另一台服务器或虚拟网卡），可以使用 XDP_REDIRECT。通过此返回值，可以控制流量的转发路径，甚至可以将其发送到特定的内核网络设备。

3.2.5 解密处理模块

SM4 是一种对称加密算法，其加密和解密过程共享相同的核心算法结构，唯一的区别在于解密时使用的轮密钥是加密时轮密钥的逆序。通过这一机制，解密过程能够还原加密后的数据。

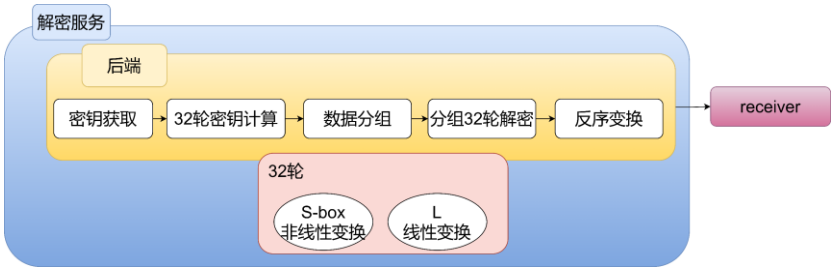

图 3-6 XDP 解密处理模块示意图

（1）算法初始化

和加密过程相同，从初始密钥中生成 32 个轮密钥（Round Key，RK），在加密过程中，轮密钥按顺序从$RK[0]$到$RK[31]$依次使用；而在解密过程中，轮密钥的使用顺序是倒序的，即从$RK[31]$到$RK[0]$

（2）密钥获取

密钥通过 BPF 映射（$sm4\_key\_map$）进行存储。由用户空间应用加载到 BPF 映射中，然后通过 eBPF 程序在加密和解密操作中使用。在解密过程中，通过 $bpf\_map\_lookup\_elem$ 获取密钥数据。

（3）分组解密

解密过程会对输入的数据流进行分块处理，确保每个数据块的解密独立性。首先将输入数据分割成固定大小的 128 位块，每个数据块作为一个解密单元进行解密。每个 128 位的数据块通过 SM4 算法的 32 轮迭代解密，输出对应的明文块。

（4）明文写回

在内核层，eBPF 程序会将解密后的数据包直接写入到网络接口的发送队列中，或者返回给相应的用户态应用进行后续处理。

3.2.6 密钥分发模块

该模块专门负责加密通信中所需密钥材料的生成、分发和定期更新，确保通信过程中的数据机密性与安全性。系统采用一种简洁高效的密钥分发机制，在通信双方之间以固定时间间隔自动轮换密钥，以提升抗重放攻击和密钥泄露风险的能力。

（1）密钥生成

密钥管理系统（Key Management System，KMS）中的密钥调度器负责定期生成对称加解密密钥。该过程采用符合国家商用密码标准的安全随机数生成器（Cryptographically Secure Pseudo-Random Number Generator ，CSPRNG），基于 $/dev/urandom$ 或 Linux 内核的 $getrandom()$ 系统调用，以确保密钥具备高度不可预测性。KMS 可配置每隔固定时间周期生成一组新的密钥材料，生成后将被组织为带有版本号（Key ID）、生成时间戳和有效期元数据的密钥结构体，支持将来密钥轮换、撤销和审计。

（2）密钥分发

为降低系统复杂度，该系统设计采用“推送式定时分发”机制，由 KMS 主动将新密钥通过本地 UDP 安全通道推送至本机运行的加解密客户端，避免使用集中式网络分发引入的额外复杂性与风险。

（3）密钥存储

客户端接收到密钥后，立即将其写入一个只读可访问的 BPF Map，通常为 $BPF\_MAP\_TYPE\_ARRAY$，Map 中按照版本号（或轮换序号）存储不同密钥条目，同时记录当前“活动密钥索引”。该 BPF Map 需

以 $BPF\_F\_RDONLY\_PROG$ 方式打开，使其仅可由 XDP 程序读取，而用户空间进程无权读取或枚举 Map 内容。

（4）XDP 程序中的密钥使用

XDP 程序在数据包处理路径中会根据当前密钥版本索引，从 BPF Map 中高效读取相应密钥内容，用于 SM4 加密或解密运算。密钥读取采用原子操作，避免轮换过程中发生中间态异常。由于 BPF Map 位于内核内存中，这一机制可以实现纳秒级别的密钥读取延迟，确保在高吞吐量网络负载下也不引入性能瓶颈。XDP 程序支持检测 Map 更新事件，自动切换当前活跃密钥，无需重启程序或进行用户态交互。

（5）密钥生命周期

每组密钥自 KMS 推送后立即生效，并设定固定的有效周期。到期后密钥将自动失效。

第四章 挑战与应对措施

4.1 挑战

4.1.1 BPF 指令数与栈空间受限

eBPF 的运行环境受到虚拟机架构的严格限制，eBPF 虚拟机对每个程序的指令数限制通常在 4096 条以内，而栈空间仅为 512 B。这一机制虽然保障了系统的安全性与稳定性，但也对实现诸如 SM4 这类复杂加密算法带来了诸多挑战。SM4 完整实现需要至少数千行展开后的指令和多个 32 轮密钥、16 字节缓冲等数据。若直接在内核态部署，会因指令数过多而被 Verifier 拒绝，或者因栈帧超出 512 B 而导致加载失败。

4.1.2 查表操作难以在 eBPF 中高效实现

SM4 的非线性变换依赖 S 盒查找，传统用户态可随意访问，但是 eBPF 不支持从全局段直接做可变偏移的读取，动态索引查表在 Verifier 眼中存在越界风险，必须显式展开或拆分，只能用静态常量数组并通过偏移访问。SM4 的非线性变换将 32 位输入拆分，各自映射到 S 盒，这些操作在 eBPF 中会被拆分为多条边界检查指令，显著增加指令数目。

4.1.3 轮密钥存储与访问开销

SM4 轮密钥共 32 个 32 位值。源码在每次接收到包时，先从 BPF map 读主密钥 MK，再在内核中调用 $getRK$ 计算所有 RK 并缓存在栈数组 $RK[32]$。此过程涉及 32 次写栈，每次都要做栈边界和 map 映射检查，处理一个数据包的密钥扩展延迟较高。若把 RK 保留在寄存器数组，也会因寄存器数量受限（最多 11 个通用寄存器可用于存放临时变量）而无法完整容纳。

4.1.4 Verifier 对循环与指针运算严格检查

eBPF Verifier 要求所有循环必须能在编译期确定最大迭代次数，并对每次指针偏移作安全证明。对数据负载部分进行分块的循环和对每块内部的循环若不进行展开，Verifier 会直接拒绝执行；加了指令展开后，又导致指令数骤增。对于涉及指针的运算，Verifier 需要确保每次访问在可访问界限范围之内，必须在循环展开后对所有偏移显式检查。

4.1.5 协议栈缺失导致通信链路重构复杂

XDP 程序仅在链路层处理原始以太网帧，无完整内核 UDP/IP 协议栈。源码必须自行维护并更新以太网头、IP 头、UDP 头的长度字段和校验和，并在加密后交换源/目的 MAC、IP 和端口，才能将加密包正确回送。任何字段计算或字节序处理错误，都会导致数据被下层丢弃。重构过程逻辑分散在多处，给调试与性能优化带来额外挑战。

4.2 应对措施

4.2.1 循环展开与代码拆分

针对 SM4 核心的 32 轮加解密循环，使用 $\#pragma\ unroll$ 将其展开为 32 段线性代码，并将对

payload 的最多 100 组处理也拆分为固定上界的多段逻辑。这样，Verifier 能够静态验证每条路径无动态循环，且在拆分后每段函数内指令数维持在可接受范围内。同时，将原本在单一函数中的大数组拆分成多个小段寄存器变量或分别处理，避免单次栈使用超过 512 B。

4.2.2 S 盒静态映射

将 S 盒常量放在只读数据段中，避免占用栈空间；对 SM4 算法的相关函数进行内联展开，将一次 32 位 S 盒变换拆为四次对常量区的直接内存读取，借助 eBPF 的只读数据段访问特性，消除对动态下标的担忧。

4.2.3 轮密钥预计算与寄存器缓存

因为密钥是动态变化的，轮密钥的计算不可避免，为降低对栈的依赖，将 RK 数组分段加载到可用寄存器中，利用循环展开，使子密钥生成与加密迭代并行展开在同一函数体中，既保证了内核态实时计算轮密钥，又避免了大量栈操作。

4.2.4 Verifier 友好编码

对所有指针偏移操作前，显式进行边界判断，如果越界则终止操作。对数据负载进行填充后必须重新加载计算首尾指针，并在每个分支后立即验证。循环拆分后，对于每个固定偏移都做同样检查，确保 Verifier 在静态分析时能够证明内存安全。

4.2.5 协议逻辑独立实现与模块化封装

将以太网、IP、UDP 头的重写和校验独立为一套模块化函数：包括 MAC 交换、IP 源/目的地址重写、UDP 端口互换，以及对应的校验和重计算。XDP 主处理流程仅调用这些小函数，从而避免在一个巨大函数中混杂过多协议细节。

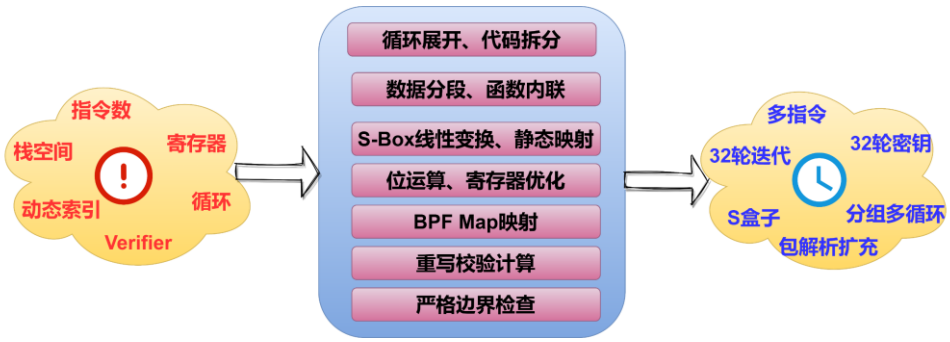

图 4-1 基于 eBPF 的 SM4 实现挑战与对应措施示意图

第五章 系统测试

5.1 功能测试

在系统实现后，需要对其功能和稳定性进行测试。本系统测试了一下几个方面：

（1）系统的功能完整性测试，确保系统能够完整运行。

（2）系统功能测试，给定输入，确保系统能够正确接收明文输入，并输出加密后和解密后的结果。

（3）密钥分发模块测试，启动密钥分发服务器和密钥接收客户端，对比分发的密钥内容以及 BPF Map 中存储的内容，调试输出加密解密过程过程中采用的密钥，确保对应的加解密过程中使用相同的密钥。

（4）系统可靠性测试，确保系统在连续运行的环境下能够不出现重大错误导致系统崩溃。

5.1.1 测试环境

本系统测试环境如下：测试平台由三台虚拟机组成，操作系统均为 Ubuntu 22.04 LTS。Flask 前后端部署在虚拟机 A 上，通过 IP 地址 192.168.58.131 向加解密服务器发送 UDP 数据包，其中，加密服务器 B 的 IP 地址为 192.168.58.129，解密服务器的 IP 地址为 192.168.58.128。同时 KMS 密钥分发服务器也部署在虚拟机 A 上。在启动该系统前，KMS 已经独立启动。当加解密服务器需要工作时，会从 BPF Map 中读取事先获取的密钥。

5.1.2 密钥分发测试

启动加密服务器和解密服务器，通过运行 deploy.sh 脚本自动化加载运行加解密服务器，通过输出的调试输信息确保接收密钥的 BPF Map 挂载成功。

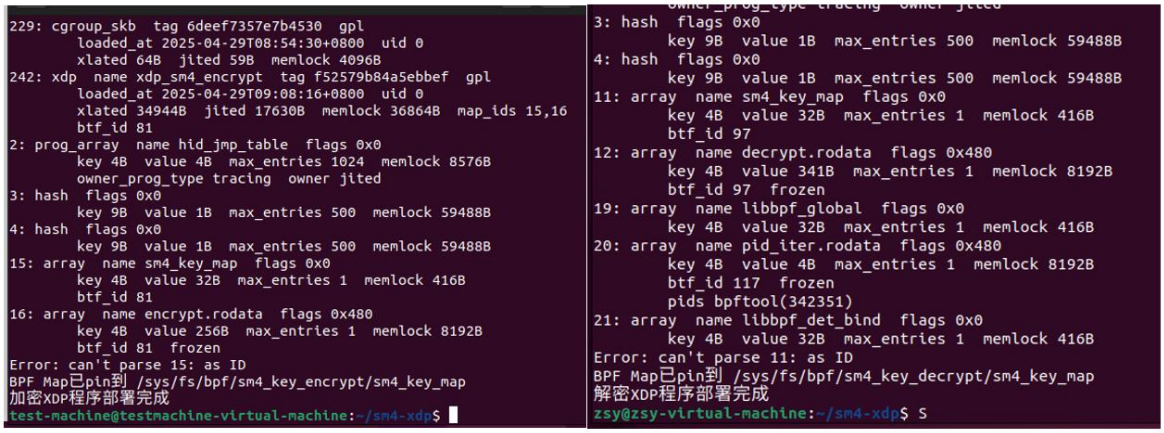

图 5-1 加解密服务器部署示意图

启动密钥分发客户端，选择 $-r$ 参数进行随机密钥生成，通过输出的调试输信息观察到密钥分发系统正常运行，生成了第一个随机密钥。当启动密钥接收客户端后，会看到 KMS 与另外两个接收密钥的加密服务器、解密服务器建立了连接，该连接是稳定的 TCP 连接，确保密钥正常分发与接收。

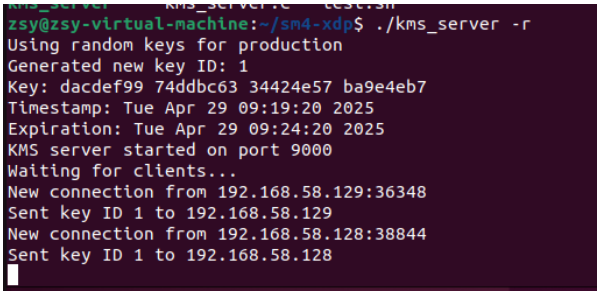

图 5-2 密钥分发服务部署示意图

启动密钥接收客户端，通过 *./encrypt_key_client* 192.168.58.132 */sys/fs/bpf/sm4_key_encrypt /sm4_key_map* 30和*./ decrypt _key_client* 192.168.58.132 */sys/fs/bpf/sm4_key_decrypt/sm4_key_map* 30 启动加密服务器和解密服务器上的密钥接收客户端，这两个客户端会同时接收 KMS 分发的密钥并保存在 BPF Map 中供 XDP 程序调用。通过输出的调试输信息可以看到两者都成功与 KMS 建立连接并收到了相同的密钥，密钥接收端的轮询频率是 30 秒。

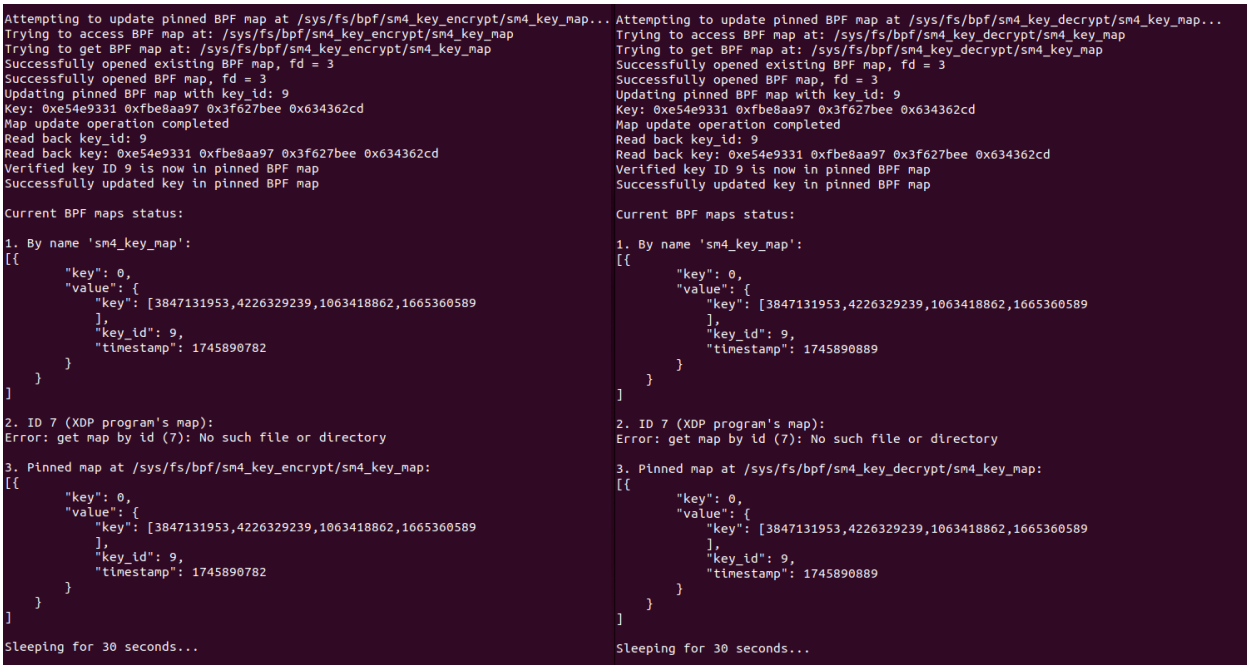

图 5-3 密钥接收服务部署示意图

5.1.3 前端测试

在 Pycharm 中运行 Flask 系统。可以在浏览器中输入网络地址进行过访问。下面是前端的控制面板模块和加解密部分模拟模块，模拟了发送方向接收方发送信息的流程，发送方的明文信息会先经过加密服务器进行加密，加密后的密文信息会经过公共链路传输到解密服务器进行解密，解密后的明文会发送到接收方。发送方和加密服务器在一个安全域中，解密服务器和接收方在一个安全域中，两个安全域之间是由网关连接的公共开放链路。

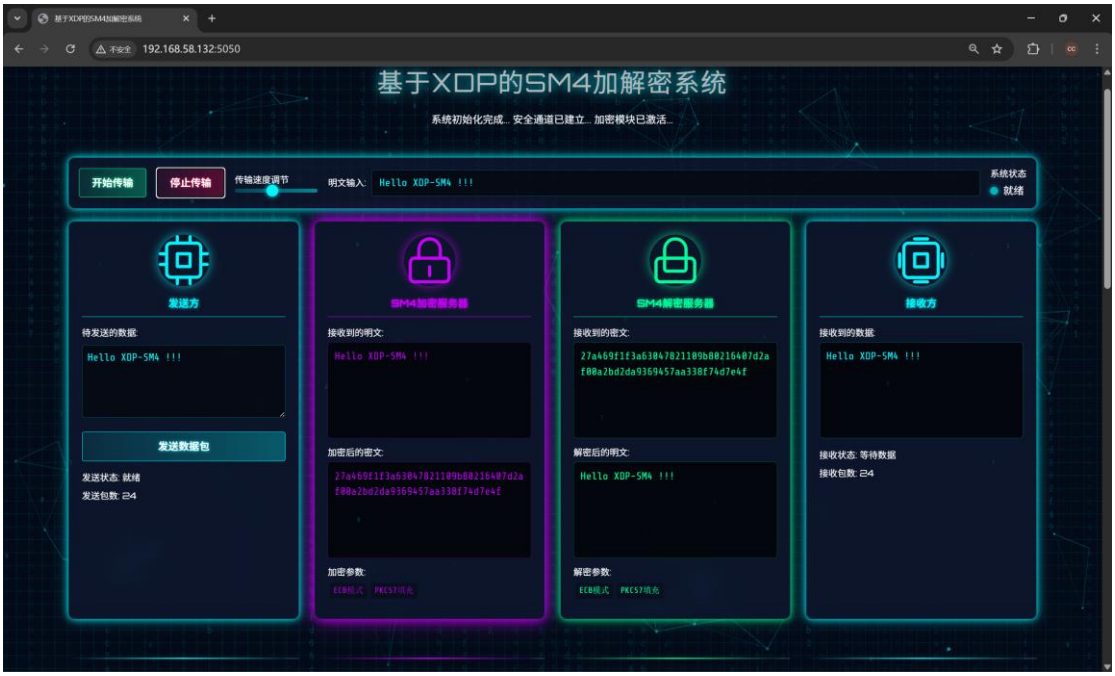

图 5-4 加解密模块测试示意图

可以在控制面板的明文输入框中输入要发送的明文，该控制面板控制明文的发送与停止，点击开始传输后，会持续发送明文信息，点击停止传输后会停止发送。又或者在发送方区域的待发送数据框中输入要发送的信息，然后点击发送数据包发送一条信息。

在加解密模拟模块下面是可视化模块，以动画的形式展示了密钥分发的流程以及数据包传输过程中发生的变化，并有简单的数据统计。

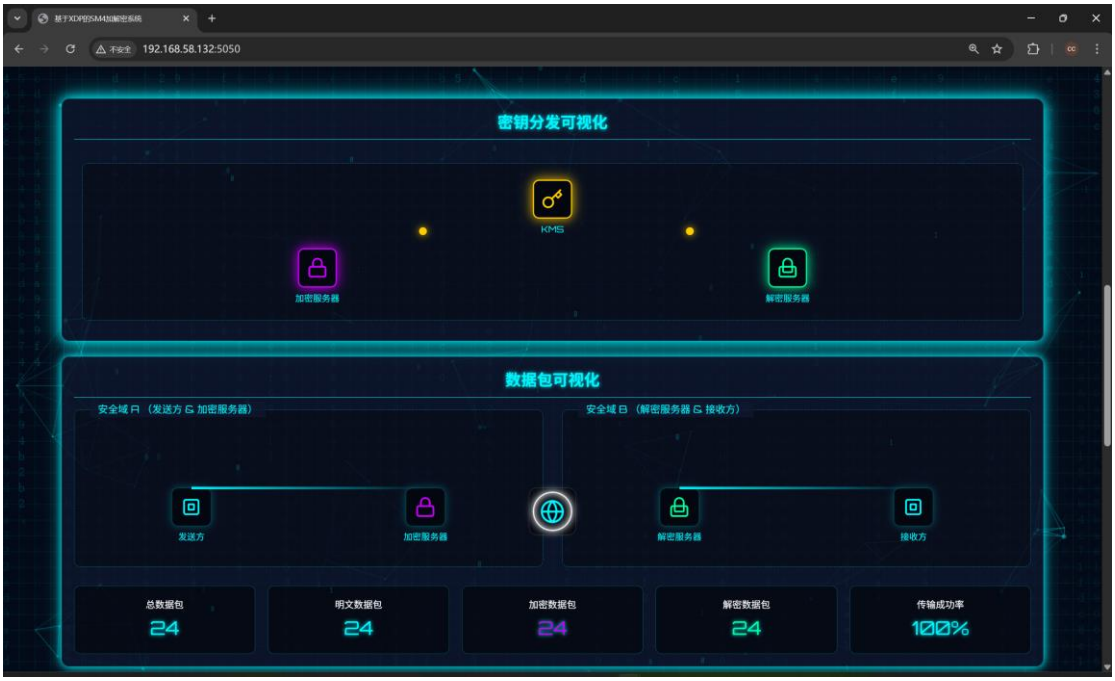

图 5-5 密钥分发与数据包可视化示意图

下面的模块是详细的性能指标数据展示：每成功从发送方到接收方发送一条信息都会统计一组实时数据，包括系统吞吐率、整个过程中的延迟、加解密处理时间以及加解密服务器的资源使用情况。

图 5-6 系统性能指标统计示意图

5.1.4 后端测试

系统运行中可会有意料外的错误，为此本系统也实现对重要数据内容的记录，在运行过程中会进行日志记录，内容如下：

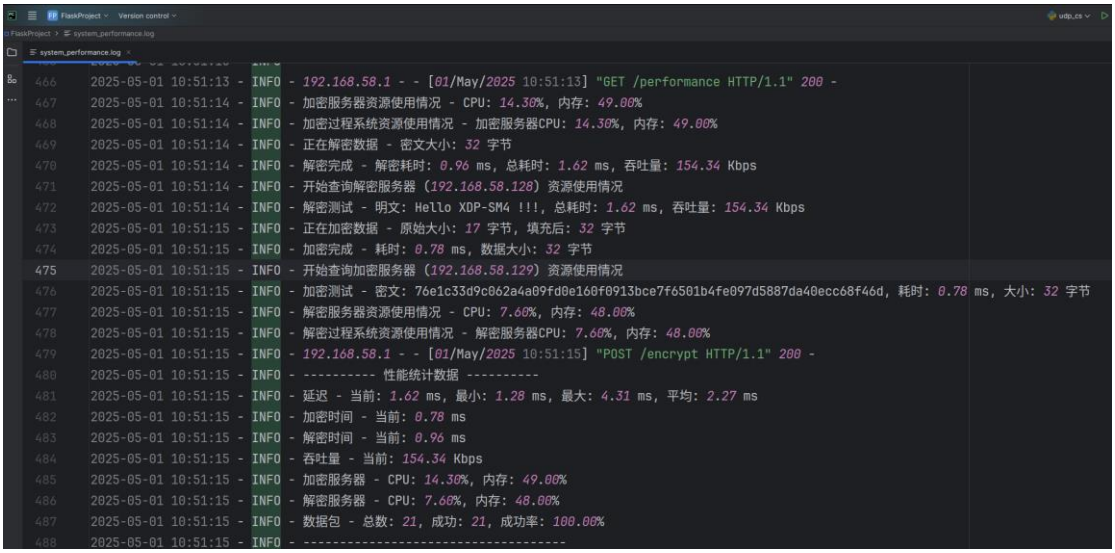

图 5-7 后端日志记录示意图

对于每一对数据，都应该保持密钥一致，可以在服务器上通过 *sudo cat /sys/kernel/debug/*

$tracing/trace\_pipe$ 查看调试信息：

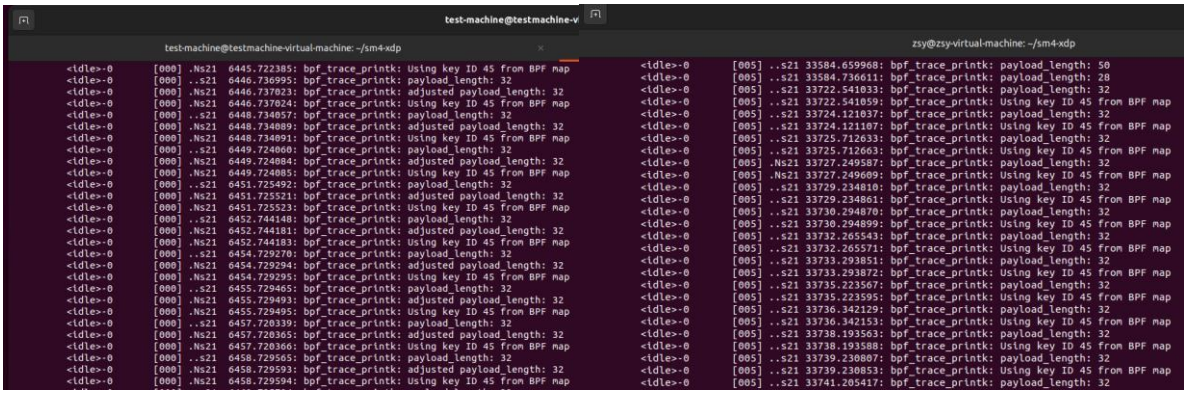

图 5-8 加解密密钥追踪示意图

5.2 性能测试

5.2.1 测试环境

本测环境由两台 Ubuntu 22.04 LTS 虚拟机组成。每台虚拟机均配置了一块 Mellanox ConnectX-5 100 GbE 高性能网络接口卡。实验部署了本研究设计的加密传输系统，客户端的 IP 地址为 192.168.58.171，加密服务端的 IP 为 192.168.58.170，其中 XDP 程序挂载在 ens2np0 网卡上，用户空间程序负责加密和转发。在测试中，通过使用自研测量程序，对固定报文负载下的单包往返时延进行采样，并借助 iperf3 生成 UDP 数据流并测量整体网络吞吐量和丢包率。

5.2.2 性能指标

系统性能评估主要通过时延、吞吐量和丢包率三个关键指标来衡量网络加密系统的表现。

（1）往返时延

往返时延指的是数据包从源端发送到目的端并返回源端所需要的总时间，通常以微秒为单位。该指标反映了加密处理对数据包传输延迟的影响，尤其是在高负载和加密操作频繁的网络环境中，往返时延的增加可能导致系统的响应速度变慢，进而影响用户体验。

（2）吞吐量

吞吐量是指单位时间内成功传输的数据量，通常以比特每秒（bps）、千比特每秒（Kbps）或兆比特每秒（Mbps）为单位。它是衡量网络加密系统性能效率的关键指标，越高的吞吐量表示网络系统可以在单位时间内处理更多的数据，从而提供更高效的加密服务。

（3）丢包率

丢包率（Packet Loss Rate）是指在网络传输过程中，未能成功到达目的端的数据包占发送总包数的比例，通常以百分比表示。它反映了网络或系统在高负载和复杂处理条件下的可靠性和稳健性。

5.2.3 结果分析

（1）往返时延

实验数据表明，随着数据包负载长度的增加，平均往返时延整体呈现出非线性递增趋势。在相同负载条件下，用户态加密的平均时延明显高于 XDP 模式。这主要是由于用户态需要在内核态和用户态之

间频繁进行上下文切换，增加了 CPU 负载和内存访问时间。

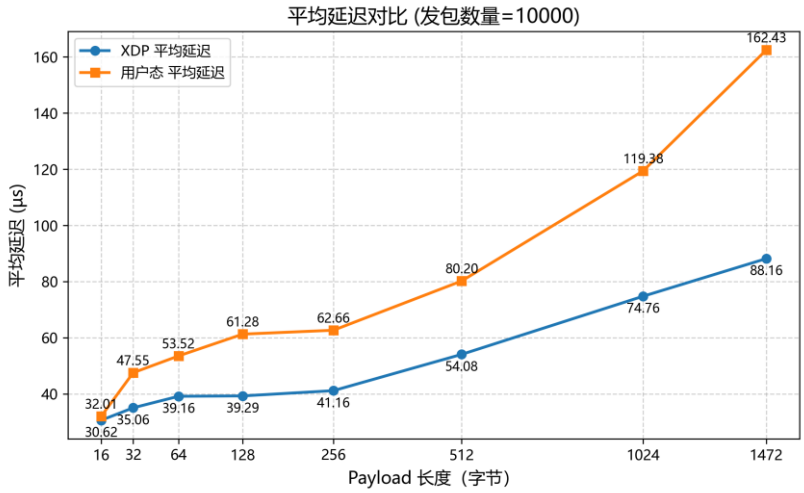

图 5-9 平均时延数据结果图

从图 5-9 中可见，在发包数量固定为 10000 的条件下，XDP 方案在各报文长度下的平均往返延迟始终显著优于用户态处理方式。随着 Payload 长度从 16 B 增加到 1472 B，XDP 的平均延迟由约 30.62 μs 增加到 88.16 μs，而用户态方案的延迟则从 32.01 μs 增至 162.43 μs。两者之间的延迟差值从最小的 1.39 μs（16 B）增长至 74.27 μs（1472 B）。整体来看，XDP 方案在各报文长度下均显著降低了平均延迟，相较于用户态处理，延迟降低率从 4.34%（16B）稳步提升至 45.72%（1472B），在典型负载范围（如 512B～1472B）内，延迟降低幅度普遍在 30%～45%。随着报文长度增大，XDP 的优势不仅未减弱，反而在高负载场景下更显著，表现出良好的可扩展性和稳定性。该结果表明：基于 XDP 的内核态加密处理架构在保障安全性的同时，能够显著降低时延开销，非常适合用于对时延敏感的网络安全应用场景。

（2）吞吐量

在吞吐量测试中，系统同样展现出随着报文大小增加而显著提升的非线性增长特性；与此同时，XDP 模式始终优于用户态处理，且优势随着报文变大愈加明显。

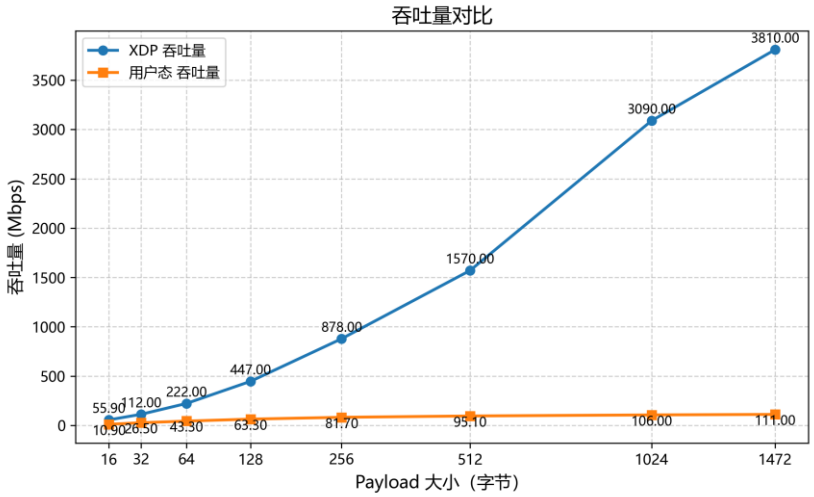

图 5-10 吞吐量数据结果图

从图 5-10 中可见，随着 Payload 从 16 B 增加至 1472 B，XDP 模式与用户态处理均表现出吞吐量的持续提升，但增长速度与上限差异明显。其中，XDP 吞吐量从 55.9 Mbps 快速提升至 3810 Mbps，增长速度显著；而用户态处理在初期增长较快，但自 256 B 起趋于饱和，仅由 81.7 Mbps 增至 111 Mbps，增长显著受限。

从处理性能对比来看，XDP 在所有报文长度下均显著优于用户态处理，且随着报文增大优势愈发明显。在小报文（16–128 B）场景中，XDP 吞吐量为用户态的约 4.2–7.1 倍；中等报文（256–512 B）的吞吐量提升至 10–16 倍以上；而在大报文（1024–1472 B）场景下，XDP 吞吐量最高达 3810 Mbps，是用户态的 34.3 倍。整体来看，XDP 相比用户态平均提升约 14 倍吞吐量，充分验证了其在低延迟、高并发网络环境下的强大适应能力与效率优势。

（3）丢包率

通过使用 iperf3 工具测试不同负载长度下的丢包率，实验对比了 XDP 与传统用户态处理模型的性能差异。结果显示，用户态的丢包率显著高于 XDP 模式，且随着报文负载从 16 B 增长至 1472 B，用户态丢包率由 70% 急剧上升至 98%，性能持续恶化。而 XDP 模式则表现出较强的稳定性，其丢包率始终控制在低水平，从 2.9%增加至 13%，总体增幅平缓，表现出更好的扩展性与可靠性。。

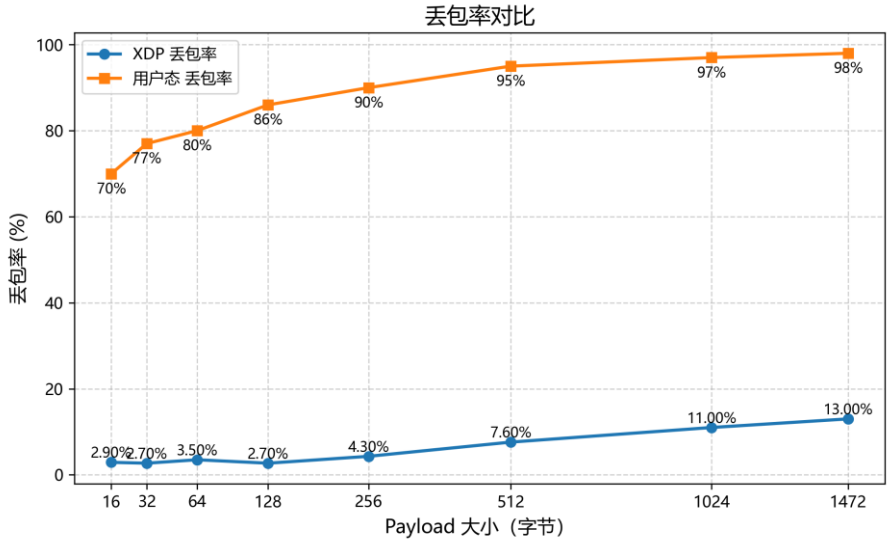

图 5-11 丢包率数据结果图

如图 5-11 所示，在最小报文（16 B）下，XDP 相较于用户态将丢包率降低了 67.1 个百分点；在最大报文（1472 B）下，丢包率差距扩大至 85 个百分点。通过进一步计算，用户态处理的平均丢包率约为 87.88%，而 XDP 平均丢包率仅为 6.23%，即整体平均丢包率下降约 81.8%。这充分验证了在高并发网络转发场景下，XDP 模型凭借其内核级执行优势，能有效避免上下文切换带来的延迟和拥塞，显著提升网络数据可靠性。

第六章 总结及展望

6.1 总结

本研究针对现有数据加密传输系统在性能与安全性平衡方面的挑战，提出了一种基于 eBPF/XDP 框架与 SM4 国产加密算法结合的高性能内核态透明加密解决方案。全文工作总结如下：

本研究提出了基于内核态的无拷贝加密框架，通过 XDP 技术将加密逻辑下沉至网卡驱动层，消除了传统方案中的上下文切换与数据拷贝开销。本研究成功解决了 SM4 算法在 eBPF 受限环境中的实现挑战，通过循环展开、代码拆分、S 盒静态映射等技术，克服了指令数限制、栈空间受限等技术障碍。设计了低开销、安全可靠的密钥分发系统，支持密钥的动态生成、分发与更新，通过 BPF Map 实现内核态与用户态的高效密钥同步。实验表明，相较于传统用户态加密方案，本系统在往返时延上平均降低 30%~45%，在吞吐量上平均提升 10 倍以上。

本研究成功实现了国产 SM4 加密算法与 Linux 内核 eBPF/XDP 技术的深度融合，验证了国密算法在内核态高性能场景下的可行性，为推动国产密码技术在高性能安全通信领域的应用提供了实践参考。

6.2 展望

虽然本研究已在 eBPF/XDP 环境下实现了 SM4 加密算法的高效部署，但仍有多个方面值得深入探索和完善：

（1）算法与硬件协同优化：

未来可探索 SM4 算法与国产硬件平台（如龙芯、飞腾处理器）的协同优化路径，研究利用硬件加速指令集（如龙芯的 SM4 指令集）提升内核态加密性能。同时，可探索与智能网卡和可编程数据平面结合，将加密运算进一步下沉至网络接口硬件，实现“零 CPU 开销”的加密处理。

（2）多核并行与高吞吐扩展：

当前系统尚未充分发挥多核处理器的并行计算能力。未来可引入 XDP 中的多队列机制，结合 RSS（Receive-Side Scaling）技术，实现多核心并行处理加解密流量，线性扩展系统吞吐量。特别是在高速网络（25/40/100Gbps）环境下，需要设计细粒度的负载均衡机制，确保加密任务能够均匀分布在各 CPU 核心上。

（3）协议栈与状态管理增强：

本研究主要针对 UDP 报文实现了加密处理，未来可扩展支持 TCP、SCTP 等有状态协议，这需要解决连接跟踪、流重组、顺序保证等复杂问题。可考虑结合 eBPF 中的 *socket maps* 和 *socket redirect* 功能，实现针对有状态连接的高效加密处理，同时保持协议语义的正确性。

（4）密钥体系与安全增强：

现有的密钥管理机制可进一步完善，引入完整的公钥基础设施（Public Key Infrastructure，PKI）体系，支持基于数字证书的身份认证与密钥协商。同时，可探索将可信执行环境（如 Intel SGX、ARM TrustZone）与 eBPF 结合，为密钥材料提供更强大的保护。针对侧信道攻击风险，需研究基于恒定时间的 SM4 实现，并评估在 eBPF 环境下的安全性表现。

（6）动态配置与策略控制：

增强系统的动态配置能力，支持基于流量特征、安全策略的细粒度加密处理。通过 eBPF Map 实时

更新策略规则，对不同的流量采用不同的加密强度或处理模式，在安全性与性能之间实现更精细的平衡。

（7）国产密码算法全家族支持：

扩展对 SM2、SM3 等其他国密算法的支持，构建完整的国密算法内核态应用体系。探索如何在 eBPF 环境中高效实现 SM2 椭圆曲线算法，为内核态密钥交换和数字签名提供支持，进一步增强系统的安全性和功能完备性。

通过以上方向的深入研究，可进一步提升基于 eBPF/XDP 的网络加密系统的性能、安全性和适用性，为构建自主可控、高性能的网络安全基础设施提供更加坚实的技术支撑。

结束语

落笔至此，四年本科学业即将画上句点，求知之旅亦将步入新的征程。回望这四年时光，犹如白驹过隙，令人唏嘘不已。在这段弥足珍贵的学习历程中，有太多恩师、挚友与亲人的身影值得我铭记于心。

衷心感谢我的导师冯温迪老师在论文研究过程中给予的悉心指点与耐心引导。每次讨论中，冯老师深入浅出的分析和审慎缜密的思维，不仅为本文的完成提供了学术支撑，更为我今后的科研之路点亮了明灯。正是因老师循循善诱的督促与鼓励，我才能克服研究中的重重阻碍，最终完成这篇凝聚心血的毕业论文。

诚挚致谢大学期间所有任课教师的辛勤教诲与倾囊相授。诸位教师在专业知识的海洋中为我导航，在技术难题面前为我解惑。每一位老师独特的教学风格和学术造诣，共同构筑起我知识体系的坚实基石，使我在计算机领域能够逐渐形成自己的见解与思考。

特别感谢实验室的学长学姐，以及所有在科研路上与我并肩奋斗的同窗伙伴。是大家的支持与陪伴让我的科研道路不再孤单。在项目合作中，大家的思维碰撞不断启发我前行的灵感，深夜调试代码时的相互鼓励更是支撑我渡过一个又一个艰难时刻。每一次阶段性成果的实现，每一段小小突破的喜悦，都因大家的分享而更加难忘。我们不仅是科研路上的同行者，更是人生旅途中彼此珍贵的伙伴。

同时感谢我的室友们，在这四年朝夕相处的时光里，你们用乐观和坚持感染我、激励我，引导我在困境中依然积极向上、奋发图强。无论是平日生活的点滴琐碎，还是心灵深处的成长互助，都是我大学时光中最温暖的底色与最亮丽的风景。

难以言表的感谢属于我的家人，是他们默默的付出与无条件的爱，给了我追求梦想的勇气与底气。无论面对怎样的选择与挑战，家人始终是我坚实的后盾与温暖的港湾。感谢家人对我的理解与包容，让我能够全身心投入到学业中。

此刻回望自己走过的路，有汗水也有收获，有迷茫也有突破。从青涩的大一新生到如今即将踏入社会的毕业生，每一次挑灯夜战的钻研，每一次失败后的重整旗鼓，每一次灵光乍现的欣喜，都是成长路上不可或缺的一部分。这段旅程教会了我学术研究的严谨与创新，也让我领悟到坚持不懈的价值与意义。

站在人生新阶段的起点，我满怀感恩，也满怀期待。星辰大海，征途漫漫。站在毕业的门槛，我既怀念过去，更期待未来。愿用所学回馈社会，以专业之力续写精彩人生。青春不散场，山水有相逢。

参考文献

[1] IBM Corporation. Cost of a Data Breach Report 2024 [EB/OL]. (2024-07-30) [2024-12-24]. https://www.ibm.com/reports/data-breach.

[2] YOACHIMIK O, PACHECO J. Record-breaking 5.6 Tbps DDoS attack and global DDoS trends for 2024 Q4[EB/OL]. (2025-01-21) [2025-02-03]. https://blog.cloudflare.com/ddos-threat-report-for-2024-q4/.

[3] Cybersecurity and Infrastructure Security Agency (CISA), Federal Bureau of Investigation (FBI). Enhanced Monitoring to Detect APT Activity Targeting Outlook Online[R]. Washington, D.C.: CISA, 2023.

[4] He K, Kim D D, Asghar M R. Adversarial machine learning for network intrusion detection systems: A comprehensive survey[J]. IEEE Communications Surveys & Tutorials, 2023, 25(1): 538-566.

[5] 中国互联网络信息中心. 第 55 次《中国互联网络发展状况统计报告》[EB/OL]. (2025-01-17) [2025-02-03]. https://cnnic.cn/n4/2025/0117/c88-11229.html.

[6] 国务院. 网络数据安全管理条例[EB/OL]. (2024-09-30) [2024-12-24]. http://www.gov.cn/zhengce/content/202409/content_6977766.htm.

[7] 工业和信息化部. 工业和信息化部关于印发《工业和信息化领域数据安全管理办法（试行）》的通知[EB/OL]. (2022-12-13) [2024-12-24]. https://www.gov.cn/zhengce/zhengceku/2022-12/14/content_5731918.htm.

[8] Kureshi R R, Mishra B K. A comparative study of data encryption techniques for data security in the IoT device[M]//Internet of Things and Its Applications: Select Proceedings of ICIA 2020. Singapore: Springer Nature Singapore, 2022: 451-460.

[9] Wang Y. Computer Network Data Security Encryption Technology[C]//2024 International Conference on Electrical Drives, Power Electronics & Engineering (EDPEE). IEEE, 2024: 365-369.

[10] Google. HTTPS encryption on the web: Percentage of page loads over HTTPS[EB/OL]. (2024-12) [2025-05-03]. https://transparencyreport.google.com/https/overview?hl=en.

[11] Cai Y, Song W. A Federated Learning Framework Using a Secure, Controllable and Efficient Multi-Key Homomorphic Encryption Scheme[C]//International Conference on Database Systems for Advanced Applications. Singapore: Springer Nature Singapore, 2024: 491-500.

[12] Sharma A. Balancing National Security and Personal Privacy: Legal Implications of Encryption Backdoors in Global Cybersecurity Policy[J]. Legal Studies in Digital Age, 2022, 1(1): 53-67.

[13] National Institute of Standards and Technology. Post-Quantum Cryptography: CRYSTALS-Kyber Standard[EB/OL]. (2023-07-05) [2024-12-24]. https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions.

[14] 李胡, 彭长根, 侯金秋. 流密码框架下的 SM4 专用认证加密算法[J]. Journal of Computer Engineering & Applications, 2024, 60(2).

[15] 白荣华,魏强,郭瑞,等.政务信息系统商用密码集约化平台设计与实现[J].信息安全研究,2023,9(05):461-468.

[16] Trisolino A. Analysis of Security Configuration for IDS/IPS[D]. Politecnico di Torino, 2023.

[17] Kulsum A S, Abhisha B H. Implementation and Visualization of Context Switching[J]. Grenze International Journal of Engineering & Technology (GIJET), 2024, 10.

[18] Panaite D C. Evaluating the performance of eBPF-based security software in a virtualized 5G cluster[D]. Politecnico di Torino, 2024.

[19] Vieira M A M, Castanho M S, Pacífico R D G, et al. Fast packet processing with ebpf and xdp: Concepts, code, challenges, and applications[J]. ACM Computing Surveys (CSUR), 2020, 53(1): 1-36.
[20] Cilium. CNI Performance Benchmark[EB/OL]. Cilium Documentation. [2025-02-03]. https://docs.cilium.io/en/latest/operations/performance/benchmark.html.
[21] Deokar M, Men J, Castanheira L, et al. An Empirical Study on the Challenges of eBPF Application Development[C]//Proceedings of the ACM SIGCOMM 2024 Workshop on eBPF and Kernel Extensions. 2024: 1-8.
[22] 中中华人民共和国国家互联网信息办公室. 国家网络空间安全战略[EB/OL]. 北京: 中华人民共和国国家互联网信息办公室, (2016-12-27) [2025-02-03]. https://www.cac.gov.cn/2016-12/27/c_1120195926.htm.
[23] Attiq M O, Mushtaq M T, Yousafzai A, et al. XDP-ML: A Game-Changer in Intrusion Detection Systems for Modern Cybersecurity[J].
[24] Fabre A. L4Drop: XDP DDoS Mitigations[EB/OL]. (2018-11-28) [2025-02-03]. https://blog.cloudflare.com/l4drop-xdp-ebpf-based-ddos-mitigations/.
[25] Bianchi G, Bonola M, Pontarelli S, et al. Creating Complex Network Services with eBPF: Experience and Lessons Learned[J]. IEEE Communications Magazine, 2019, 57(3): 98-104.
[26] Vieira M A M, et al. Fast Packet Processing with eBPF and XDP: Concepts, Code, Challenges, and Applications[J]. ACM Computing Surveys, 2019, 53(1): 16:1–16:36.
[27] Söderman F. Performance Evaluation of AF_XDP and DPDK in Multi-Buffer Packet Processing Applications[J]. 2024.
[28] Intel®. Benchmark the performance of single core forwarding with XXV710 and 82599/500 Series 10G[EB/OL]. [2025-02-03]. https://doc.dpdk.org/dts/test_plans/nic_single_core_perf_test_plan.html.
[29] Traynor K. Save power with OVS-DPDK PMD thread load-based sleeping[EB/OL]. Red Hat Developer, (2023-10-16) [2025-02-03]. https://developers.redhat.com/articles/2023/10/16/save-power-ovs-dpdk-pmd-thread-load-based-sleeping.
[30] DPDK and energy efficiency[EB/OL]. NANOG Mailing List, (2021-03-04) [2025-02-03]. https://mailman.nanog.org/pipermail/nanog/2021-March/212472.html.
[31] Gangidi A, Miao R, Zheng S, et al. Rdma over ethernet for distributed training at meta scale[C]//Proceedings of the ACM SIGCOMM 2024 Conference. 2024: 57-70.
[32] Hu J, Shen H, Liu X, et al. RDMA transports in datacenter networks: survey[J]. IEEE Network, 2024.
[33] 阿里云开发者社区. 基于阿里云 eRDMA 的 GPU 实例大幅提升多机训练性能[EB/OL]. (2023-03-23) [2025-02-03]. https://developer.aliyun.com/article/1179757.
[34] NVIDIA 技术博客. 使用 NVIDIA DOCA GPUNetIO 解锁 GPU 加速的 RDMA[EB/OL]. (2024-06-13) [2025-02-03]. https://developer.nvidia.com/zh-cn/blog/unlocking-gpu-accelerated-rdma-with-nvidia-doca-gpunetio/.
[35] Sarkar B, Saha A, Dutta D, et al. A Survey on the Advanced Encryption Standard (AES): A Pillar of Modern Cryptography[J]. 2024.
[36] Shakor M Y, Khaleel M I, Safran M, et al. Dynamic AES encryption and blockchain key management: a novel solution for cloud data security[J]. IEEE Access, 2024, 12: 26334-26343.
[37] Ghosal A K, Sardar A, Chowdhury D R. Differential fault analysis attack-tolerant hardware implementation of AES[J]. The Journal of Supercomputing, 2024, 80(4): 4648-4681.
[38] Dong L, Ye X, Zhuang L, et al. AES software and hardware system co-design for resisting side

channel attacks[J]. Expert Systems, 2024, 41(10): e13664.
[39] Grassl M, Langenberg B, Roetteler M, Steinwandt R. On the practical cost of Grover for AES key recovery. NIST PQC Standardization Conference. 2024.
[40] 国家密码管理局. SM4 分组密码算法［S］. 北京: 中国标准出版社, 2016. GB/T 32907–2016.
[41] 中国密码学会. SM4 分组密码算法安全性分析报告［R］. 北京, 2015.
[42] 飞腾信息技术有限公司. 飞腾处理器密码算法优化白皮书［R］. 天津, 2022.
[43] Santosh Kumar R, Prakash K, Krishna S R M. Cryptanalysis of RSA with composed decryption exponent with few most significant bits of one of the primes[J]. Journal of Computer Virology and Hacking Techniques, 2024, 20(1): 195-202.
[44] Barker W, Souppaya M, Newhouse W. Migration to Post-Quantum Cryptography. NIST, 2021. [EB/OL]. https://csrc.nist.gov/pubs/pd/2021/08/04/migration-to-postquantum-cryptography/final.
[45] Gidney C, Ekerå M. How to factor 2048-bit RSA integers in 8 hours using 20 million noisy qubits. arXiv preprint arXiv:1905.09749, 2019. [EB/OL]. https://arxiv.org/abs/1905.09749.
[46] Wang X, Zeng Y, Hu B, et al. Analysis and verification of SM2 cryptographic certificates[C]//6th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM 2024). IET, 2024, 2024: 208-213.
[47] Rozlomii P D I, Yarmilko P D A, Naumenko S, et al. Modern encryption methods in IoT: hardware solutions and cryptographic libraries for data protection[J]. Авторський колектив, 2024: 28.
[48] Barone M. Toward High-Speed Tunneling Technologies: A New WireGuard Parallel Architecture for Linear Throughput Scaling[D]. Politecnico di Torino, 2024.
[49] Akter S, Khalil K, Bayoumi M. A survey on hardware security: Current trends and challenges[J]. IEEe Access, 2023, 11: 77543-77565.
[50] Xue D, Ramesh R, Jain A, et al. OpenVPN is open to VPN fingerprinting[J]. Communications of the ACM, 2025, 68(1): 79-87.
[51] Chen Y, Li Q, Tian L, et al. Navigating the VPN Landscape: A Comparative Study of L2TP, IPSec, and MPLS VPN Technologies[C]//2024 4th International Conference on Electronic Information Engineering and Computer Science (EIECS). IEEE, 2024: 614-617.
[52] Stallings W. Cryptography and Network Security: Principles and Practice [M]. 7th ed. Pearson Education, 2016.
[53] TechTarget. Managing VPN bandwidth requirements, speed and overhead [EB/OL]. [2025-02-03]. https://www.techtarget.com/searchnetworking/answer/What-is-the-bandwidth-utilized-on-Internet-VPN.
[54] Zhang Y., et al. A Generic High-Performance Architecture for VPN Gateways[J]. Electronics, 2024, 13(11): 2031.