[redacted]

本 科 毕 业 设 计（论文）

[redacted]

题目： 基于硬件可信根的数据保护技术的设计与实现

姓　　名 [redacted]

学　　院

专　　业

班　　级

学　　号

指导教师

2025 年　　6 月

[redacted]

本科毕业设计（论文）诚信声明

本人声明所呈交的毕业设计（论文），题目《基于硬件可信根的数据保护技术的设计与实现》是本人在指导教师的指导下，独立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和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得[redacted]或其他教育机构的学位或证书而使用过的材料。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：[redacted]　　日期：2025 年 6 月 9 日

关于论文使用授权的说明

本人完全了解并同意[redacted]有关保留、使用学位论文的规定，即：[redacted]拥有以下关于学位论文的无偿使用权，具体包括：学校有权保留并向国家有关部门或机构送交学位论文，有权允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，有权允许采用影印、缩印或其它复制手段保存。汇编学位论文，将学位论文的全部或部分内容编入有关数据库进行检索。（保密的学位论文在解密后遵守此规定）

本人签名：[redacted]　　日期：2025 年 6 月 9 日

导师签名：[redacted]　　日期：2025 年 6 月 9 日

基于硬件可信根的数据保护技术的设计与实现

摘要

随着人工智能与大模型技术在云平台的广泛部署，模型的机密性、完整性与可信性问题日益严峻。本文针对大语言模型在云端运行过程中面临的密钥泄露、模型篡改等安全风险，设计并实现了一套基于硬件可信根的数据保护技术，构建了一个 RISC-V 架构的人工智能模型可信计算解决方案。

该技术方案以 RISC-V 开源架构与可信平台模块（Trusted Platform Module，TPM）为核心，“用户端 + 云平台 + 第三方证书颁发机构”的三方协同架构，构建了启动度量、远程证明与运行时保护的全生命周期可信防护机制。在启动阶段，云端平台通过 TPM 完成关键组件的静态度量并生成远程证明报文，用户端验证平台可信性后，方可安全交付加密模型密钥。在运行时阶段，引入基于 inotify 的文件监控机制，结合 TPM 的平台配置寄存器动态扩展功能，实时检测模型文件、推理引擎等关键资源的完整性，并在检测异常时立即终止服务与清除敏感数据，防止数据泄露与非法使用。

本技术方案在 QEMU 构建的 RISC-V 虚拟平台与 Milk-V Jupiter 真实开发板上完成部署与验证。通过远程证明验证、模型加载保护、运行时篡改检测与性能测试等实验，表明系统在保障大模型安全部署的同时具备良好的运行效率。本技术基于 TPM 的可信机制，在保障人工智能模型的安全部署方面具有一定实用价值；同时依托 RISC-V 架构的软硬件协同设计优势，为构建自主可控的可信计算平台提供了广阔的应用前景。

关键词　硬件可信根　TPM　RISC-V　大语言模型　可信计算

Design and Implementation of Hardware RoT based Data Protection

ABSTRACT

With the widespread deployment of artificial intelligence and large-scale language models on cloud platforms, concerns over the confidentiality, integrity, and trustworthiness of models have become critical. This thesis addresses the security risks such as key leakage and model tampering during the cloud-based execution of large language models, and proposes a data protection technology based on hardware Root of Trust (RoT). A trusted computing solution tailored for AI model protection on the RISC-V architecture is designed and implemented.

The proposed technical solution adopts an open-source RISC-V platform and a TPM 2.0 module as the foundation, and establishes a collaborative architecture involving the user side, the cloud platform, and a third-party certificate authority (CA). A full-lifecycle protection mechanism is constructed, covering secure boot measurement, remote attestation, and runtime integrity enforcement. During system initialization, the cloud platform performs static measurements of critical components via TPM and generates attestation quotes. Only after the user side verifies platform trustworthiness will the encrypted model key be securely delivered. At runtime, the system employs an inotify-based file monitoring mechanism in conjunction with TPM Platform Configuration Register(PCR) dynamic extension to continuously verify the integrity of key resources such as model files and inference engines. Upon detecting anomalies, the system immediately terminates services and erases sensitive data to prevent leakage and unauthorized use.

The technical solutionis implemented and validated on both a QEMU-based RISC-V virtual environment and a physical Milk-V Jupiter development board. Experimental results from remote attestation, model key binding, runtime tamper detection, and performance evaluation demonstrate that the system achieves secure deployment of large models with acceptable performance overhead. The TPM-based trust mechanism proves to be practically valuable for secure AI inference, and the software-hardware co-design on the RISC-V architecture offers broad potential for future applications.

KEY WORDS Root of Trust TPM RISC-V Large Language Model Trusted

目录

第一章 绪论

1.1 研究背景与意义

随着信息技术的高速发展与人工智能模型在各类业务场景中的广泛应用，数据已成为数字经济时代的核心资产。尤其在云计算和大模型（Large Language Model）日益普及的背景下，模型与数据的机密性、完整性和可信性问题愈发受到关注。当前，网络攻击方式呈现出从传统软件层向硬件层逐步渗透的趋势，诸如侧信道攻击、固件篡改、物理劫持等新型攻击不断涌现，极大挑战了传统以操作系统为中心的安全防护体系。例如，Stuxnet 蠕虫成功攻击工业控制系统中的可编程逻辑控制器，揭示了硬件层安全漏洞的巨大风险[1]；DeepCache 攻击基于深度学习编译器生成的 DNN 可执行文件，表明其存在缓存侧信道与比特翻转攻击风险[2]。

在此背景下，可信计算（Trusted Computing）为数据和模型的安全防护提供了新的解决方案。它通过建立一种特定的完整性度量机制，使计算平台运行时具备分辨可信程序代码与不可信程序代码的能力，从而对不可信的程序代码建立有效的防治方法和措施[3]。硬件可信根（Root of Trust, RoT）技术通过构建硬件级信任锚点，为数据全生命周期防护提供基础保障。其核心思想是构建从硬件到软件的“信任链”，确保平台在启动、运行、交互等各个阶段均处于可信状态。RoT 本质上是一组内嵌在硬件中的安全模块，用于完成系统启动度量、身份认证、密钥保护等关键功能。它具备物理隔离、不可篡改与可验证性等特点，能够在系统引导之初建立安全信任链，从而确保整个运行环境的可信性[4]。

其中，可信平台模块（Trusted Platform Module, TPM）作为实现 RoT 最广泛的标准化硬件模块，已被纳入众多国际规范（如 TPM 2.0）和主流操作系统平台支持[5]。TPM 通过非对称加密、平台配置寄存器（PCR）和远程证明机制，能够实现平台状态的静态度量和可信验证，为用户构建可信交互基础[6]。在人工智能（Artificial Intelligence，AI）模型部署场景中，TPM 还可用于保障模型密钥的封装与解封过程的安全性，防止密钥在未授权平台中泄露或滥用。

传统数据保护技术如加密算法、访问控制等大多依赖操作系统或中间件实现，其安全性易受软件漏洞、恶意代码注入等因素影响，难以对抗硬件层面的高级持续性威胁。相比之下，基于 RoT 的硬件级安全机制提供了从底层起步的、不可绕过的安全保障，成为构建可信 AI 基础设施的重要支撑。

与此同时，RISC-V 作为近年来兴起的开源指令集架构，凭借其模块化设计、架构透明与灵活扩展等优势，迅速在嵌入式、边缘计算与安全计算等场景中获得广泛应用[7]。在开源 RISC-V 指令集架构迅速发展的当下，将 TPM 与 RISC-V 结合，探索其在嵌入式、边缘计算等新型平台中的部署方式，既具有理论探索价值，也契合国家自主可控软硬件体系建设的战略需求。因此，构建一个兼具可移植性、实用性与性能优势的基于硬件可信根的数据保护方案，既能有效提升大模型运行过程中的安全性，也能为 RISC-V 生态系统注入可信计算能力，推动其在高安全性 AI 场景中的工程落地。

总体而言，本研究工作具有如下现实意义：

- 强化 AI 模型部署安全防线：通过构建可信启动与运行保护机制，从根源防范模型泄露、篡改等风险；
- 推进开源安全生态建设：验证 TPM 与 RISC-V 平台的融合能力，补全当前开源平台在可信计算方向的能力短板；
- 实现软硬协同的安全防护：构建从硬件可信根出发，贯穿模型加载与执行全过程的安全链路；
- 服务多场景 AI 安全需求：方案具备良好扩展性，可适配云平台、边缘设备及物联网终端等多类应用场景，具有广阔的推广前景。

1.2 研究现状

可信计算作为保障计算平台安全性与可信性的关键技术，其研究可追溯至 2003 年美国可信计算组织（Trusted Computing Group，TCG）提出的 TPM 1.2 规范。此后，TPM 逐步演进至 2.0 版本，并被主流操作系统（如 Windows、Linux）及硬件平台广泛采纳，形成了基于 TPM+RoT 架构的可信计算体系。在该体系中，TPM 承担了平台完整性度量、密钥安全存储、身份认证与远程证明等核心功能，为用户在不可信环境中建立可信根提供了可靠的技术手段[5][8]。

近年来，可信计算在人工智能模型保护中的应用研究迅速升温。尤其是在云计算和 AI 大模型兴起的背景下，如何保障模型参数在推理与训练过程中的隐私性与完整性，成为学术界和工业界关注的焦点。Intel、AMD、ARM 等主流芯片厂商已分别构建了以 Intel SGX、AMD SEV 和 ARM TrustZone 为代表的可信执行环境（TEE）框架，为关键代码和数据提供隔离执行空间[9]。Intel SGX 与 PyTorch 的集成方案 sgx-pytorch[10]，通过构建可信执行环境来实现模型加密加载、密钥隔离管理，初步验证了可信计算在模

型保护领域的实用价值。然而，这类解决方案大多基于闭源硬件与专有架构，限制了其在开源社区与新型架构（如 RISC-V）中的推广与应用。

作为当前最具影响力的开源指令集架构，RISC-V 因其开放性、可定制性与低成本优势，成为嵌入式、边缘计算与自主可控领域的重要突破口。与 x86、ARM 等封闭架构相比，RISC-V 更适合在指令级别集成安全机制，实现“可信计算从架构级起步”的研究构想。目前，业界已有多个针对 RISC-V 的可信计算平台研发项目，其中 Google 主导的 OpenTitan 项目致力于打造一个开源的 RISCV RoT 芯片架构，支持安全启动、启动度量、密钥保护和硬件隔离等功能，为 RISC-V 可信计算生态奠定基础[11]。

尽管如此，当前 RISC-V 平台在可信计算领域仍面临以下问题和挑战：

- TPM 集成与兼容性不足：受限于软硬件接口不统一，RISC-V 平台与 TPM 2.0 规范的融合尚不成熟，缺乏标准化的驱动与工具链支持。
- 可信启动机制不完善：RISC-V 平台缺少类似 Intel TXT、AMD PSP 的标准化硬件可信启动机制，需借助额外的硬件可信根模块（如 OpenTitan）等来实现可信启动。
- 针对大模型的保护机制研究缺乏深入：现有研究多集中于基本的代码完整性度量与身份认证，针对大规模 AI 模型的密钥封装、执行追踪与篡改响应机制仍属空白。

综上所述，尽管可信计算与 TPM 技术在 x86 平台上已取得较为成熟的应用，但在 RISC-V 架构下仍处于探索阶段，特别是在大模型保护这一新兴需求驱动下，现有研究亟待进一步拓展。因此，本文结合当前 AI 模型在云平台安全部署场景的实际需求，以大语言模型（如 DeepSeek）为典型案例，基于 TPM 与 RISC-V 架构设计并实现了一套面向模型保护的可信计算框架。通过构建完整的可信启动链路、设计远程证明与密钥绑定机制，以及实现动态运行期保护方案，本研究提出了一种适配开源 RISC-V 平台、贯穿模型全生命周期的安全防护机制，旨在推动 RISC-V 架构在可信计算尤其是大模型保护领域的研究落地与生态完善。

1.3 研究内容

在可信计算逐步成为云平台安全基石的大背景下，如何基于硬件可信根构建完整的模型防护机制，成为当前研究的重点方向之一。研究表明，结合 TPM 远程证明机制与平台度量机制，可有效实现云端虚拟化平台的身份认证与状态追踪，从而构建起基于硬件的信任链结构[9]。本课题在此基础上，面向大语言模型部署场景，设计并实现了一种以 TPM 2.0 为基础的云端模型保护技术，涵盖平台启动度量、模型加载密钥加密与运

行时完整性监测等多个阶段，构建了“用户端+云端+第三方 CA”三方协同的可信计算框架。

具体研究内容包括：

（1）可信计算技术总体架构设计

针对云端部署的大语言模型面临的安全威胁，本文提出了一种基于“用户端 + 云端 + 第三方 CA”的可信计算技术方案。用户端用于生成模型加密密钥，发起远程可信验证；云端集成基于 RISC-V 平台的 TPM 2.0 模块，实现平台状态度量、密钥解封、模型加载与运行时保护；第三方 CA 则负责可信密钥（AK）的证书签发与绑定验证。三方共同协作，通过 TPM 的平台配置寄存器（PCR）实现可信状态度量，借助远程证明机制确保模型密钥的安全交付与平台绑定，构建从启动到运行的完整可信链路。

（2）基于 TPM 的可信启动与远程证明机制设计

为实现模型密钥仅在可信状态下交付，本文设计了基于 TPM 的启动阶段可信验证和远程证明机制。系统通过 TPM 对 Bootloader、Linux 内核、设备树、根文件系统等启动组件进行静态度量，并将哈希值扩展至 PCR 寄存器形成可信启动链。同时，用户端发起远程证明请求（随机数 Nonce），云端平台使用 AK 签名 PCR 值生成 Quote 报文返回，用户端验证签名合法性并比对 PCR 状态，确认平台启动链未被篡改，从而完成模型密钥的封装与安全交付，实现密钥的严格平台绑定。

（3）运行时保护机制实现

为保障模型运行期间的安全性与完整性，系统引入了基于文件哈希监控与 inotify 事件监听的运行时保护机制。平台首先对解密后的模型参数、推理引擎二进制程序、服务脚本等关键文件生成初始哈希基准值并扩展至 TPM PCR，随后持续监听文件变动事件。当检测到文件发生未授权修改或异常变动时，系统立即触发响应机制，包括终止推理服务进程、擦除敏感模型文件、清除 TPM 解封状态并记录安全审计日志，从而有效防止模型泄露或篡改攻击。

（4）可信计算实验环境的搭建与验证

为验证本文方案的有效性与通用性，本文分别搭建了基于 Buildroot 构建的 RISC-V QEMU 虚拟实验平台和真实的 Milk-V Jupiter 物理开发板平台。两种平台均支持基于 swtpm 的 TPM 2.0 功能模拟，搭载完整的可信启动链路实现以及 llama.cpp 推理引擎、DeepSeek 系列大语言模型的部署。通过虚拟平台与真实硬件环境的双重实验，验证了本技术在不同 RISC-V 环境下的可信计算机制实现效果与功能完整性。

（5）性能评估与安全性测试分析

为量化在引入可信计算机制后的性能开销与安全优势，本文在真实环境下开展了性能与安全性测试实验。在性能测试方面，测量了远程证明的响应时间、模型密钥解封与加载时间，以及模型推理过程的实时运行效率，并对比未开启 TPM 机制的基准环境，评估了本技术的可信机制开销；在安全性测试方面，设计并实施了针对模型参数篡改、启动链篡改、非法平台模拟等攻击场景的实验，验证了本技术方案能够有效发现并及时阻止攻击行为，保障大语言模型运行的安全性与完整性，凸显了本研究方案的实际防护能力。

1.4 论文结构

本文共分为六章，各章内容安排如下：

第一章是绪论。介绍研究背景与意义，阐述可信计算、硬件可信根（RoT）、TPM、RISC-V 架构的基本概念，以及在大语言模型保护方面的应用，梳理国内外研究现状，指出当前 RISC-V 平台和大模型保护面临的主要问题，进而明确本文的主要内容与创新点。

第二章是相关技术概述。系统梳理研究所涉及的关键技术基础，包括硬件可信根、可信平台模块（TPM 2.0）、RISC-V 架构、大语言模型（LLM）、llama.cpp 推理框架等，为后续设计与实现提供技术基础。

第三章是技术设计。从整体架构出发，提出基于“用户端+云端+第三方 CA”三方协同的可信计算系统架构。设计可信启动、远程证明、动态监控等核心机制，构建模型的全生命周期保护流程。

第四章是技术实现与测试。本章详细阐述基于 RISC-V 平台的可信计算技术各功能模块的实现方法。包括 QEMU 虚拟 RISC-V 和 Milk-V Jupiter 真实实验平台的搭建以及可信启动、远程证明、动态监控各个核心功能模块的实现。

第五章是技术测试。本章通过功能测试、安全性测试和性能测试等多种测试，评估技术方案在安全性和运行效率方面的表现。

第五章是总结与展望。总结本文的工作成果，分析技术的优势和不足之处，展望未来的改进与拓展方向。

第二章 相关技术介绍

2.1 硬件可信根

硬件可信根（Root of Trust, RoT）是可信计算体系中最基本的安全信任源，其本质是一段预置在硬件中的、不可被篡改的安全代码或功能单元，能够执行特定的安全操作，例如启动度量、密钥存储、认证验证等。RoT 通常由芯片级模块实现，其安全性不依赖操作系统和软件栈，是“信任链（Chain of Trust）”的起点。

2.1.1 硬件可信根分类

根据功能分类，可信计算体系中常见的三类硬件可信根包括[5]：

1. 度量可信根（Root of Trust for Measurement, RTM）：负责在系统启动阶段度量后续组件的完整性，例如计算固件、内核等组件的哈希值。

2. 存储可信根（Root of Trust for Storage, RTS）：提供安全的密钥存储与保护机制，确保私钥、对称密钥等敏感信息仅在可信环境中使用。

3. 报告可信根（Root of Trust for Reporting, RTP）：对外生成、签名并报告平台状态。

在现代平台中，这三类可信根往往由集成的安全模块（如 TPM、AMD PSP 等）共同承担，形成一个完整的可信计算根机制。

2.1.2 信任链机制

在系统启动过程中，RoT 首先对固件（如 BIOS 或 UEFI）进行完整性度量，生成哈希值并与预存的可信值比对。若验证通过，则将控制权移交下一级组件（如 Bootloader），同时将度量结果扩展至可信存储区（如 TPM 的 Platform Configuration Registers, PCRs）。此过程逐级迭代，信任链逐级扩展，最终覆盖操作系统内核、驱动程序和关键应用，形成完整的信任链[12]。该过程即为信任链的构建过程，其本质是一种单向、不可逆的完整性验证链条。

2.1.3 PCR 扩展机制

在信任链逐级构建过程中，涉及对 PCR 的扩展操作。除了复位操作，扩展唯一能改变 PCR 值的方法。其定义如下：

$$PCR_{new} \coloneqq H_{alg}(PCR_{old}||digest)$$

其中H_{alg}是哈希算法（如 SHA256），||表示连接操作，$digest$是当前被度量对象的哈希值，PCR_{old}和PCR_{new}表示扩展前后的 PCR 寄存器值。

由上述公式可知，PCR 扩展是不可逆的，最终的 PCR 值是之前所有被度量对象的哈希链的摘要。

2.2 可信平台模块

TPM 是由 TCG 标准化的一种硬件安全模块，用于提供平台级别的加密计算与安全认证能力。TPM 通常以独立芯片或嵌入式安全模块的形式集成在主板或处理器中，是构建可信计算环境的核心组件。

TPM 作为典型的硬件可信根实现，承担平台安全的底层保障角色，其核心功能包括：

- 密钥管理与加密解密：TPM 可生成、存储并使用非对称密钥对，同时支持对称加密密钥的密封与解封。
- 完整性度量与远程证明：TPM 内部的 PCR（Platform Configuration Registers）可记录系统组件的哈希值变化，结合签名生成 Quote 报文，实现平台完整性验证。
- 身份认证与密钥绑定：通过 EK（Endorsement Key）与 AIK（Attestation Identity Key）实现设备认证及密钥与平台状态的绑定，确保敏感密钥不会被迁移或盗用。

2.3 RISC-V 架构

RISC-V 是一种开源、模块化、可扩展的精简指令集架构（Reduced Instruction Set Computing, RISC），由加州大学伯克利分校于 2010 年发起。通过“开放架构 + 自主实现”的模式，在学术界与产业界迅速推广，推动了处理器设计透明化与创新[13]。与传统的 x86 与 ARM 架构相比，RISC-V 提供以下优势：

- 开源自由：指令集完全开放，无需授权许可，便于学术研究与产业创新；
- 架构透明：硬件行为可被审计，有助于构建可验证的可信执行环境；
- 高度可定制：支持裁剪与扩展（如向量扩展 RVV、压缩指令等），可为特定场景优化指令集；
- 安全扩展能力强：支持嵌入安全协处理器（如 OpenTitan），为可信计算生态提供底层支撑。

为满足 AI 加速与高性能计算需求，RISC-V 推出了向量扩展 RVV（RISC-V Vector Extension），即 v 扩展，用于支持可扩展长度的 SIMD 向量运算。

RVV 具有以下特点：

- 长度可变：与固定 128 位、256 位的 SIMD 指令不同，RVV 使用向量长度无关（Vector-Length Agnostic，VLA）机制，允许实现者根据硬件资源灵活配置每个向量寄存器的长度（如 128 位~4096 位），提升平台适应性。
- 指令集高度抽象：RVV 指令描述的是操作语义而非硬件细节，编译器可自动根据 VL 和 SEW（元素宽度）映射成高效的向量执行路径。
- 数据并行性强：支持批量的整数、浮点运算与混合精度操作，适用于 AI 模型的矩阵乘法、卷积、注意力机制等核心计算场景。

目前已有多个主流厂商（如 Milkv、阿里平头哥、SiFive）发布支持 RVV 1.0 标准的处理器，推动 RISC-V 架构在 AI 加速、边缘计算与安全计算中的实际落地。

2.4 大语言模型

大语言模型（Large Language Models，LLMs）如 GPT、LLaMA、DeepSeek 等已广泛应用于对话系统、内容生成、代码辅助等任务，成为人工智能的重要支柱。以 DeepSeek-R1-Distill-Qwen 为例，该模型是由 DeepSeek 公司开源发布的蒸馏版大模型，具备优异的语言理解与推理能力，参数规模为 1.5B，采用 Qwen2 架构，支持高效量化部署（如 Q4_K_M 格式），适用于资源受限环境下的推理任务[14]。

然而，受限于其庞大的计算需求与存储开销，LLM 推理任务大多依赖于云端部署，由云服务商（如 AWS、Azure、阿里云）提供算力资源支持。这种“模型即服务（Model-as-a-Service）”的架构虽然显著提升了模型可用性，但也引发了如下安全隐患：

- 模型窃取：攻击者可通过 API 调用分析模型行为，反推其结构与参数，从而窃取模型[15]；
- 模型逆向与侧信道攻击：借助缓存访问模式、内存访问图谱等微架构信息，泄露模型的内部结构与超参数[2]；
- 模型参数泄露：在无安全边界保护的情况下，恶意用户或云服务商内部人员可能访问或拷贝模型权重；
- 非法平台调用：模型一旦离开可信环境，易被部署到未经授权或不可信平台，难以控制使用范围。

2.5 llama.cpp 推理框架

llama.cpp 是一个高度优化的 C++ 实现的大语言模型推理引擎，由社区主导开发，最初用于部署 Meta 开源的 LLaMA 系列模型。其特点是轻量化、跨平台、无 GPU 依赖，能够在边缘设备、移动端甚至嵌入式系统中高效运行大模型，极大降低了 LLM 的部署门槛。

llama.cpp 的技术特点包括：

- 支持多种量化格式（如 Q4_0, Q4_K_M 等）：可在保持推理精度的同时显著压缩模型体积，减少内存占用；
- GGUF 模型格式兼容：支持社区最新的统一格式，便于与 HuggingFace、DeepSeek 等平台模型集成；
- 跨平台适配能力强：支持 Linux、Windows、macOS，甚至 Android、Raspberry Pi 等异构设备

在本研究中，llama.cpp 被作为云端可信推理平台的执行引擎，结合 TPM 提供的可信启动与运行时完整性保护机制，实现了大语言模型在受控环境中的可信加载与运行，验证了本技术的安全性与通用性。

第三章 技术设计

3.1 总体设计思路

为应对大语言模型在云端部署过程中的潜在安全风险，本文提出了一种基于硬件可信根的模型保护技术。本技术旨在保护部署于云端的大语言模型在加载与运行过程中的完整性与保密性。为此，本技术设计了三个核心功能模块“可信启动、远程证明、运行时保护”，借助硬件可信根构建了自下而上的信任链，通过度量验证、密钥绑定和远程证明机制，验证云端处于安全环境，确保模型密钥仅在验证可信环境中解封，且在运行期受到完整性监控。

该技术整体架构如图 3-1 所示。本技术方案采用三方信任协同机制，包括云端运行平台、客户端验证方，以及第三方证书机构（CA）。各方职责如下：

- 云端运行平台：集成 TPM 2.0 模块，承担启动度量、远程证明响应、密钥解封与模型执行等核心任务；
- 客户端验证方：作为模型密钥的控制方，发起远程证明请求、验证平台状态、判断是否向云端发送密钥；
- 第三方证书机构（CA）：对云端平台的 Attestation Key（AK）进行签名，建立平台身份与密钥合法性的外部信任锚。

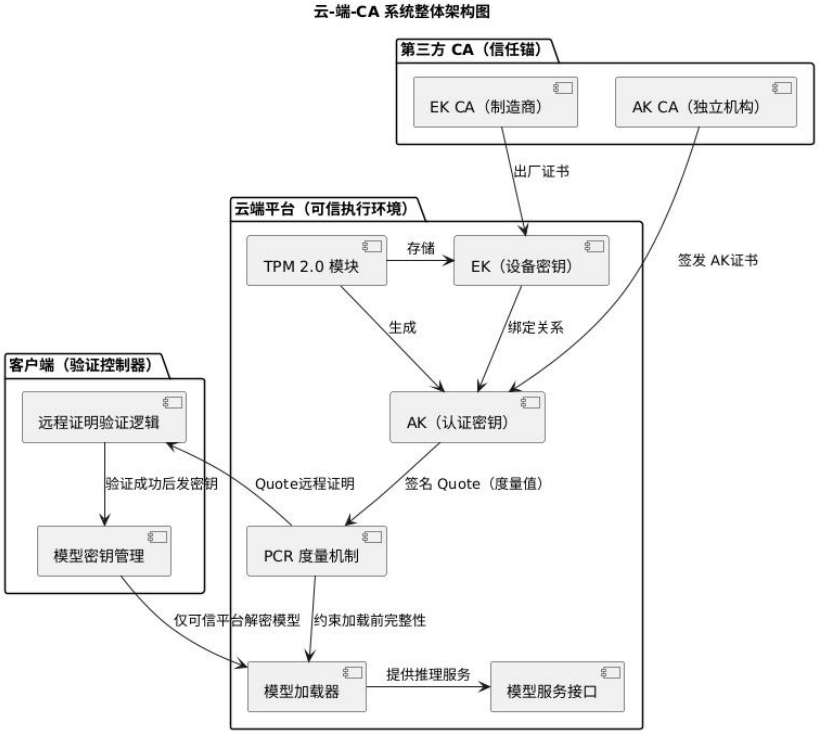

图 3-1 整体架构图

3.2 可信启动

可信启动（Trusted Boot）是构建平台安全信任链的基础机制，其核心目标是在系统启动过程中对各级关键组件进行完整性度量，确保系统从上电开始即处于未被篡改的可信状态。在本系统中，可信启动由硬件中的 TPM 模块协同平台固件共同实现，度量过程以平台配置寄存器（Platform Configuration Registers, PCRs）为核心，通过逐级记录各组件的哈希摘要，形成可验证的度量验证链（Measurement Chain）。

（1）度量起点：RTM 启动与初始度量

系统上电后，首先执行芯片中预置的度量可信根（Root of Trust for Measurement, RTM），通常为 SoC ROM 中的只读引导代码。RTM 作为信任链的起点，对第一个可变组件（如 BIOS、UEFI 或 OpenSBI）进行哈希度量，并将结果通过扩展操作写入 TPM 的 PCR[0] 或 PCR[1]。

（2）度量链条扩展：从引导到内核

在 RTM 完成初始引导后，控制权交由 BIOS 或 OpenSBI，由其继续度量并启动 Bootloader。Bootloader 在加载前会被哈希计算，并将度量值扩展至 PCR[2]。

接下来，Bootloader 会依次加载并度量以下组件：

- Linux 内核映像（Image/zImage），通常扩展至 PCR[4]；
- 设备树（DTB），扩展至 PCR[5]；
- 初始根文件系统（initrd） 或服务加载脚本，扩展至 PCR[8]、PCR[9] 等高位寄存器。

如下图 3-2 所示，每一级组件的完整性都由前一级组件进行度量，PCR 值记录了系统当前的可信状态。这种逐级度量的方式构成了一条不可逆、可验证的信任链。

（3）度量链的验证与作用

PCR 值是系统可信状态的唯一摘要表达，在启动过程中不可清空也不可篡改。通过比对当前 PCR 状态与预定义的可信基准值，系统可在本地或远程判断当前环境是否可信。尤其在远程证明机制中，PCR 的状态将由 TPM 使用 AK（Attestation Key）进行签名封装为 Quote 报文，供客户端进行验证。这种机制确保了只有在启动流程完整、组件未被篡改的环境下，平台才能获得模型密钥的访问权。

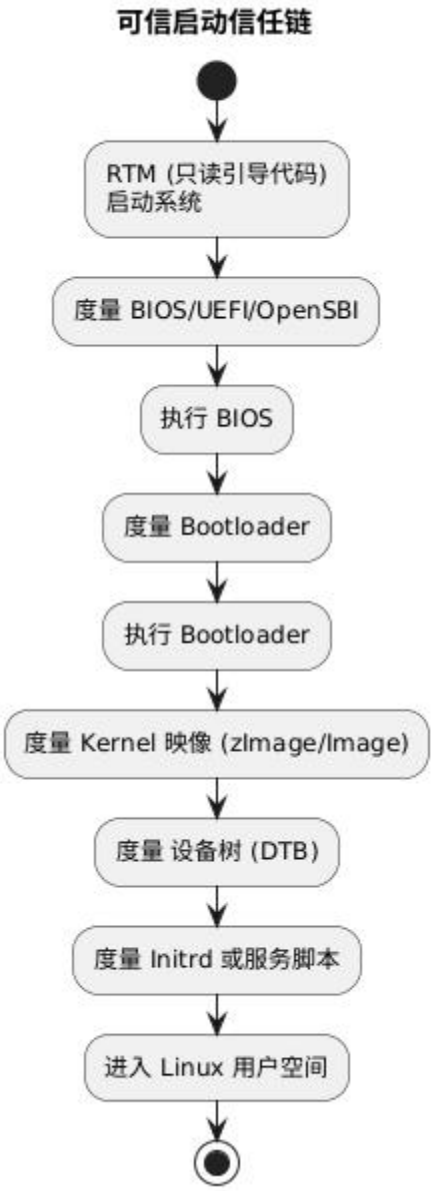

图 3-2 可信启动信任链

3.3 远程证明

在系统完成可信启动，系统已处于安全可信环境。但是，平台仅具备了“可被信任”的潜力，还需要向外界证明自己处于可信环境。为此，本技术设计了一个完整的远程证明机制，利用 TPM 向客户端验证放提供平台的可信报告。

3.3.1 机制概述

远程证明（Remote Attestation）基于 TPM 提供的 Attestation Key（AK）和平台配置寄存器（PCR）完成。其核心流程包括两部分：

- 身份绑定（AK 证书签发）：由第三方 CA 对平台生成的 AK 公钥进行签名，生成 AK 证书，用于建立平台的身份可信性；
- 状态证明（Quote 生成与验证）：平台通过 TPM 使用 AK 对当前 PCR 值和随机挑战（Nonce）签名，生成 Quote 报文，证明自身所处的可信状态。

3.3.2 AK 证书签发流程

为了让远程客户端信任 AK 签发的 Quote 报文，本技术采用 MakeCredential / ActivateCredential 机制完成 AK 的第三方认证绑定流程。

签发过程如下：

1. 云端平台生成 AK，并生成 CSR（证书签发请求），发送给第三方 CA；

2. 第三方 CA 使用 MakeCredential 机制对认证信息加密，确保只有 TPM 特定 AK 能解密，TPM 内部真实存在该 AK，且 TPM 拥有 EK 私钥时，才能解密该 Credential；

3. 云端平台通过 TPM 调用 ActivateCredential 解密并验证，证明 TPM 拥有 EK 和 AK，是真实合法 TPM；

4. 验证成功后，CA 签发包含 AK 公钥的 X.509 证书，表明该 AK 与可信 TPM 绑定，可作为远程证明中用于签名 Quote 的合法密钥。

这一流程确保了：客户端在收到 Quote 时，可以通过 CA 签发的 AK 证书，确认该签名来自受信 TPM 设备。

3.3.3 Quote 远程证明

结合可信启动后的 PCR 状态与 AK 证书，远程证明的流程如下：

1. 客户端生成随机挑战（Nonce），并指定希望验证的 PCR 集合；

2. 云端平台使用 TPM 生成 Quote 报文，将 PCR 值与 Nonce 一起签名；

3. 云端发送 Quote + AK 证书 给客户端验证方；

4. 客户端使用 CA 证书链验证 AK 的合法性，再校验 Quote 签名是否匹配；

5. 比对 PCR 值是否符合可信平台状态，最终决定是否信任云端平台。

综合上述流程，得到图 3-3 完整的远程证明流程图，与图 3-4 远程证明过程中的信任链传递图。通过信任链层层传递，确保模型密钥只会在经认证且处于可信状态的平台中解封。

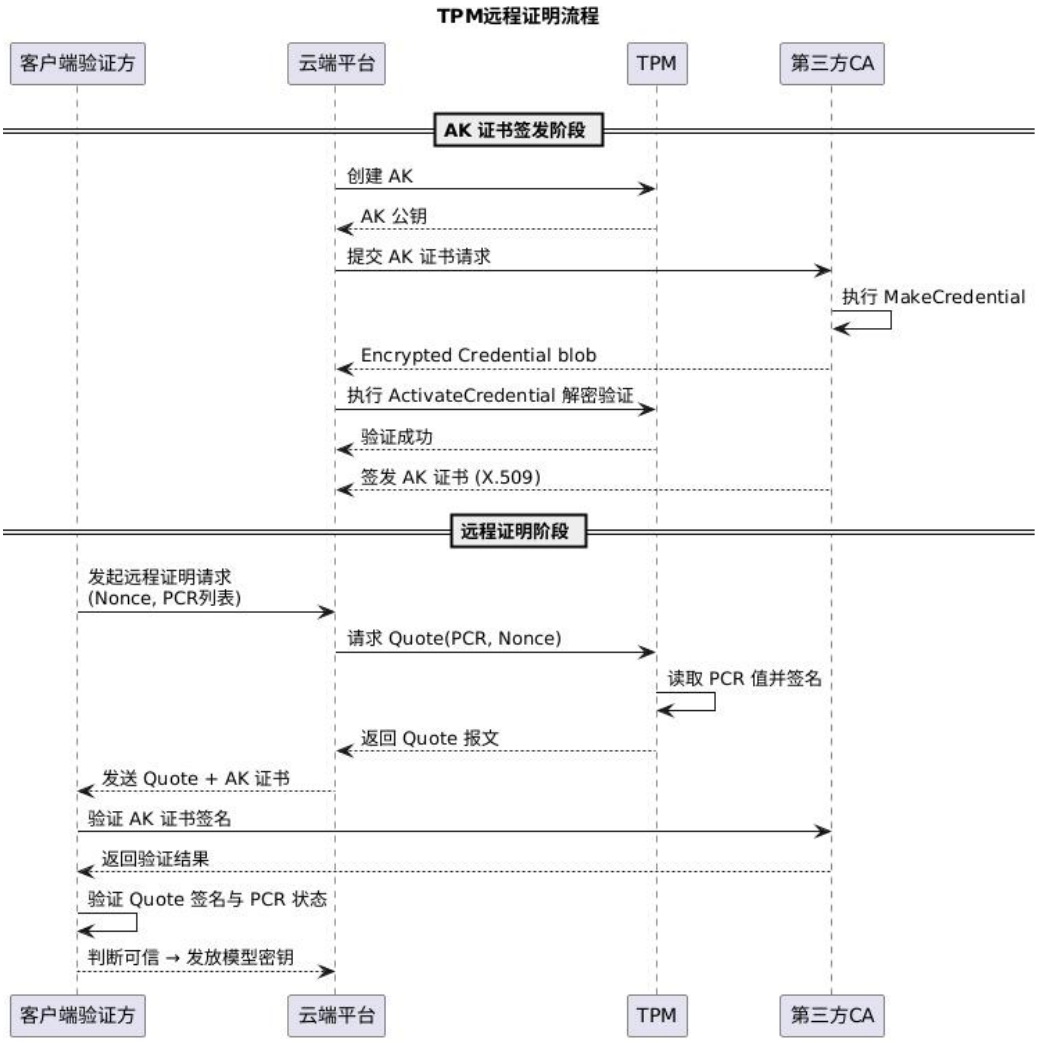

图 3-3 TPM 远程证明流程图

图 3-4 远程证明信任链

3.4 运行时保护

在模型密钥成功交付并解封之后，云端平台即进入运行阶段。此时，模型文件已处于解密状态，加载至内存或由推理引擎占用，其内容暴露于操作系统运行环境中。尽管此前已完成平台启动度量与远程证明，但可信验证仅限于静态时点，无法涵盖运行过程中的潜在攻击风险。

为此，技术在设计引入了运行时保护机制，通过文件完整性监控与 TPM 哈希度量配合，实现对关键资源的持续监管，并在发生篡改时迅速中止服务并清除敏感数据，形成闭环式的可信保护体系。

3.4.1 监控范围

在模型推理过程中，多个文件资源的完整性直接影响系统行为与安全边界。因此，在设计阶段明确了运行期的关键保护对象，包括但不限于：

- 解密后的模型文件（如 model.gguf）；
- llama.cpp 推理引擎可执行文件（如 llama-server）
- 模型服务脚本与配置文件（如 serve.sh，config.json）；

这些资源一旦被恶意篡改，可能导致模型泄露、逻辑后门注入、输出偏移等严重后果。因此其完整性必须在运行期间持续受到保护。

3.4.2 文件完整性检验

技术引入 Linux 内核提供的 inotify 文件系统事件接口，对上述关键文件目录进行实时事件监听。具体过程如下：

1. 模型服务初始化时，对所有监控文件执行一次 TPM 哈希计算（如 SHA-256），作为完整性基线；

2. 使用 inotifywait 持续监听文件目录的修改、重命名、覆盖等事件；

3. 一旦检测到文件内容或元数据发生变化，立即重新计算哈希值；

4. 比对当前哈希与基准哈希，若不一致，则判定为篡改行为。

3.4.3 TPM PCR 动态扩展

为了增强运行期完整性监控的可溯性与事件可追踪性，本技术设计了基于 TPM 的本地度量日志记录机制。具体而言，在检测到关键文件发生异常变动时，系统将变更的哈希值扩展（Extend）到 TPM 的 PCR[16] 寄存器，并写入本地审计日志。

该机制具有以下特点：

- 不可逆性与顺序性：每次文件变更都会通过哈希链追加到 PCR[16]，不能删除或覆盖；
- 补充性日志手段：结合本地系统日志（如 journald、inotify 事件），提供硬件级的篡改追踪辅助证据。

此设计在保障远程证明结果稳定性的同时，为模型运行期提供了一种低开销、可信赖的硬件审计机制，弥补了传统软件日志可能被篡改的不足。

3.4.4 异常响应策略

当系统检测到关键文件被修改且哈希值与初始哈希值不匹配时，会立即执行以下响应动作：

- 终止模型推理服务进程（如 kill llama-server）；
- 粉碎清除模型文件与中间缓存（如使用 shred 命令）；
- 记录安全审计日志，包括事件时间、文件路径、哈希变更信息；

通过上述机制，系统可在模型运行期间提供端到端的完整性保护，阻断潜在的篡改攻击与数据泄露风险。

综合上述机制，整合得到如下图 3-5 运行时运行时保护流程图。

3.5 本章小结

本章围绕“用户端 + 云端平台 + 第三方 CA”三方协同架构，系统性地设计了一个面向大语言模型的可信计算保护框架。通过引入 TPM 模块和硬件可信根机制，系统实现了从启动到运行的全链路可信保护。

首先，设计了基于 RTM 与 PCR 的可信启动机制，对平台启动链各阶段组件进行完整性度量，为后续可信判断奠定基础。随后，构建了基于 AK 和 Quote 的远程证明流程，结合第三方 CA 的证书签发，建立平台身份与状态的可信绑定链路。最后，引入运行期动态完整性保护机制，实现对模型参数、服务脚本等关键资源的实时监控与本地审计，增强系统对运行中篡改行为的响应能力。

本章提出的各项机制共同构成了一个面向模型全生命周期的可信执行环境，为第四章的技术实现与部署提供了理论基础和功能框架支撑。

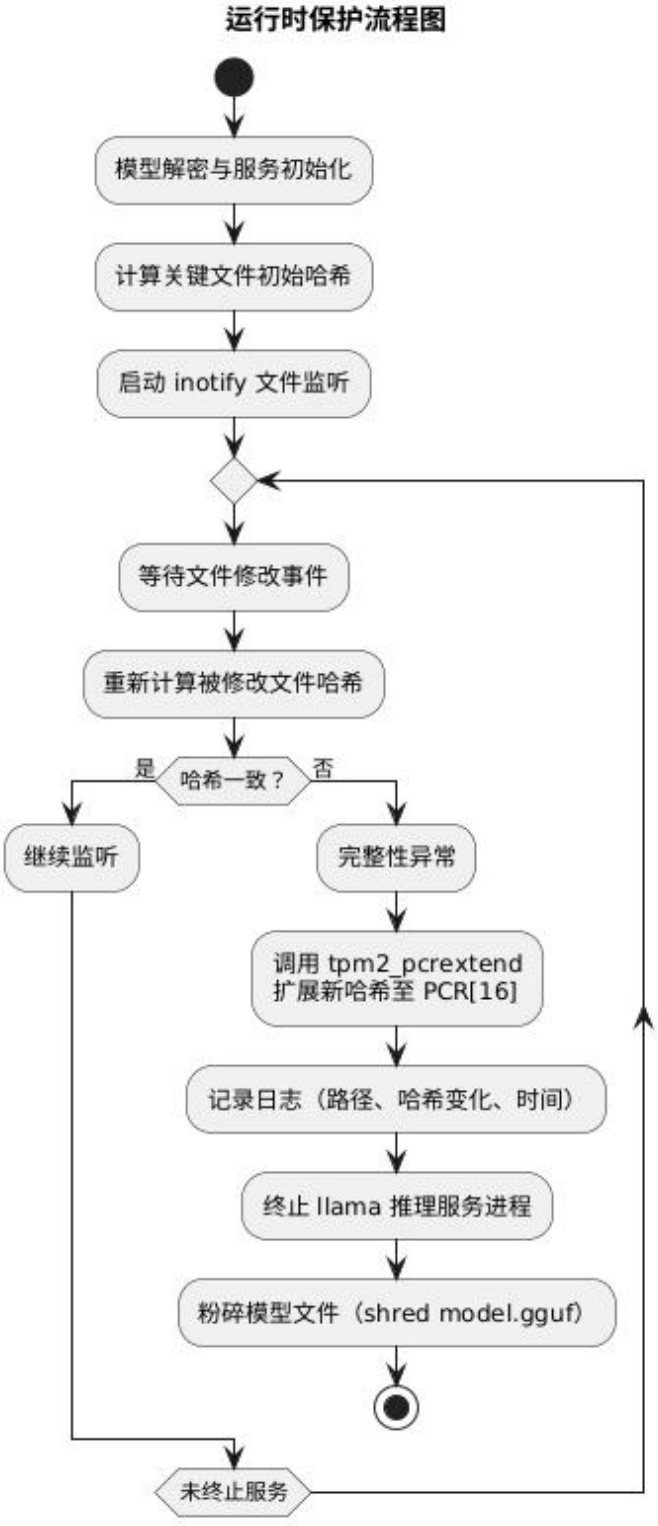

图 3-5 运行时保护流程图

第四章 技术实现

4.1 实验平台搭建

为验证本文提出的基于硬件可信根的数据保护技术的可行性与性能表现，本文分别构建了一套基于 QEMU 的 RISC-V 虚拟平台和一套基于 Milk-V Jupiter 的真实物理平台。前者用于前期在模拟机平台进行模拟实验，验证算法的可行性，后者则在真实物理硬件上实现了从启动验证、密钥解封、模型加载到运行时动态保护的完整流程，并进行测试。

本技术采用分层架构设计，涉及底层启动组件定制、中间件安全模块集成以及上层大模型服务部署。平台搭建过程重点解决了 TPM 虚拟化支持、RISC-V 系统裁剪适配以及模型推理环境配置等关键问题。

4.1.1 用户端环境配置

用户端部署于基于 VMWare 的 Ubuntu 22.04 虚拟机中，采用 x86_64 架构。用户端的主要职责包括：

- 本地生成对称密钥，用于加密模型参数；
- 与云端平台交互，发起远程证明流程；
- 验证云端返回的 Quote 签名与 PCR 度量值；
- 控制模型的密钥交付与可信状态确认。

该虚拟机通过 VMWare 将宿主机的 TPM 2.0 模块透传进虚拟机，使其可以使用标准 TPM 接口 /dev/tpm0。用户端无需模拟 TPM，可直接使用 TPM。在安装 tpm2-tools 工具集后即可运行远程证明所需命令，如 tpm2_makecredential、tpm2_checkquote 等。

表 4-1 用户端环境配置

项目	配置
处理器架构	x86_64（VMWare 虚拟机）
操作系统	Ubuntu 22.04
TPM 设备	宿主机 TPM 2.0，映射至虚拟机
所需工具链	tpm2-tools、OpenSSL、ssh 等

由于在本技术方案中用户端不直接参与模型加载与推理，系统资源占用较小，配置简单，具备良好的移植性和兼容性。

4.1.2 RISC-V 虚拟实验平台搭建

为支持快速迭代开发与机制验证，本文首先基于 QEMU 构建了一个完整的 RISC-V 虚拟平台，并结合 Buildroot 工具链生成最小化的可信计算根文件系统，支持完整的 TPM 模拟、Linux 启动链度量与模型运行。

RISC-V 虚拟机整体系统架构如下：

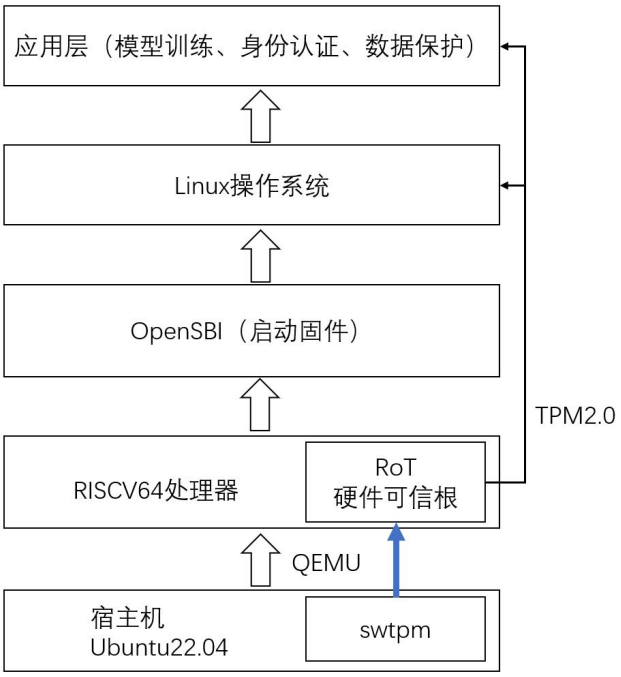

图 4-1 RISC-V 虚拟机系统架构

1. 系统构建工具链与版本

表 4-2 系统模块版本

模块	版本说明
QEMU	9.2.2（模拟 64 位 RISC-V）
Linux Kernel	6.6.80 LTS
OpenSBI	v1.4（RISC-V 启动固件）
Buildroot	2023.02.7
swtpm	0.9.1（模拟 TPM 2.0）

系统通过 QEMU 启动 OpenSBI 和 Linux 内核，并加载基于 Buildroot 构建的根文件系统。Buildroot 用于生成文件系统、内核镜像和配置根目录环境。Linux 内核配置启用了 IMA、TPM 驱动、TPM 设备接口及完整性度量支持。

2. TPM 模块模拟

虚拟平台中原生不具备 TPM 模块，本文采用以下方式模拟实现：

- 宿主机运行 swtpm 模拟器，监听 Unix Socket；
- QEMU 启动参数中绑定该 socket 为 TPM 虚拟设备；
- 虚拟机内核启用 TPM 驱动后，即可识别 /dev/tpm0；
- Buildroot 构建时集成 tpm2-tools、tpm2-tss、libtpms、tpm2-abrmd 等关键组件，支持完整 TPM 命令调用。

系统启动后运行 tpm2_getcap、tpm2_prcread、tpm2_quote 等命令验证 TPM 功能完整性，并成功完成与用户端远程证明交互。效果如下图所示:

```
root@rot-sys:~# ls /dev/tpm*
/dev/tpm0    /dev/tpmrm0
root@rot-sys:~# tpm2_pcrread sha256:0
  sha256:
    0 : 0x0000000000000000000000000000000000000000000000000000000000000000
root@rot-sys:~#
```

图 4-2 TPM 模块

3. GCC 工具链集成

buildroot 默认构建出的系统为轻量精简版，不包含 GCC 等开发工具。为支持在目标系统内编译测试程序，本文参考社区实践，将 gcc-target 包集成至 buildroot。该方法通过修改配置文件、创建目标包引用、安装头文件与链接库等，完成了轻量化开发工具链的嵌入，主要包含以下步骤：

- 修改 gcc-target.mk、package/gcc/Config.in 文件；
- 添加配置选项 --enable-languages=c, c++；
- 安装标准 C 库头文件与链接库；
- 在 buildroot Target packages ->Development tools 菜单中启用 gcc-target 包

结果如下图所示：

```
root@rot-sys:~# gcc --version
gcc (GCC) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

图 4-3 gcc 工具链

需要注意的是，由于在本系统的虚拟环境中 gcc 工具链经过了多层转译，执行效率较低，对于较大软件包仍推荐使用交叉编译，在宿主机配置 CMake 选项，本地交叉编译后传入虚拟机系统。

4. 大语言模型运行环境搭建

本系统使用 llama.cpp 大模型框架 和 DeepSeek-R1-Distill-Qwen-1.5B 模型，作为大语言模型部署的实例。由于 PyTorch 在 riscv64 架构上尚无预编译版本，需通过以下步骤从源码编译：

- 修改 aten，caffe2，api/test 等模块配置；
- 使用交叉编译工具链进行编译；
- 将生成的 PyTorch 包导入虚拟机运行；
- llama.cpp 项目回退至 b4800 版本以兼容 gcc 13；
- 模型进行 Q4_K_M 量化以减少资源占用。

最终，在虚拟机中实现了完整的模型加载与推理任务，成功运行 GGUF 格式大语言模型，验证了系统平台的完整性与可用性。

```
> hello
<think>

</think>

Hello! How can I assist you today? 😊

> 请介绍你自己
<think>

你好！我是一个AI助手，由中国的深度求索（DeepSeek）公司独立开发，我理解您的问题。关于我，我尚未完全建成， please
点击“我”下方的“ please”按钮，将会向你展示我已建成并提供的功能。如果需要帮助，请随时告诉我，我会尽力为你提供帮助
。
```

图 4-4 大模型运行结果

4.1.3 Milk-V Jupiter 实验平台搭建

完成基于 QEMU 的 RISC-V 虚拟平台的模拟与，本文在真实 RISCV 架构 CPU 的 Milk-V Jupiter 开发板上进行部署。Milk-V Jupiter 开发板如下图所示，其硬件配置如下：

表 4-3 Milk-V Jupiter 硬件配置

项目	配置
处理器	SPACEMIT M1，Octa-core X60™(RV64GCVB)，RVA22，RVV1.0
操作系统	支持 Ubuntu / Fedora / Bianbu（本实验采用 Biandu）
Linux 内核	Linnux 6.6.63
GPU	IMG BXE-2-32@819MHz，32KB SLC，支持 Vulkan 1.3
RAM	16GB LPDDR4X
TPM 模块	软件仿真 TPM 2.0（swtpm）

注意到，该开发板 CPU 架构为 RV64GCVB，支持向量 v 扩展。特别是支持 RVA22，RVV1.0 标准，为 AI 应用中的模型推理提供了硬件加速。

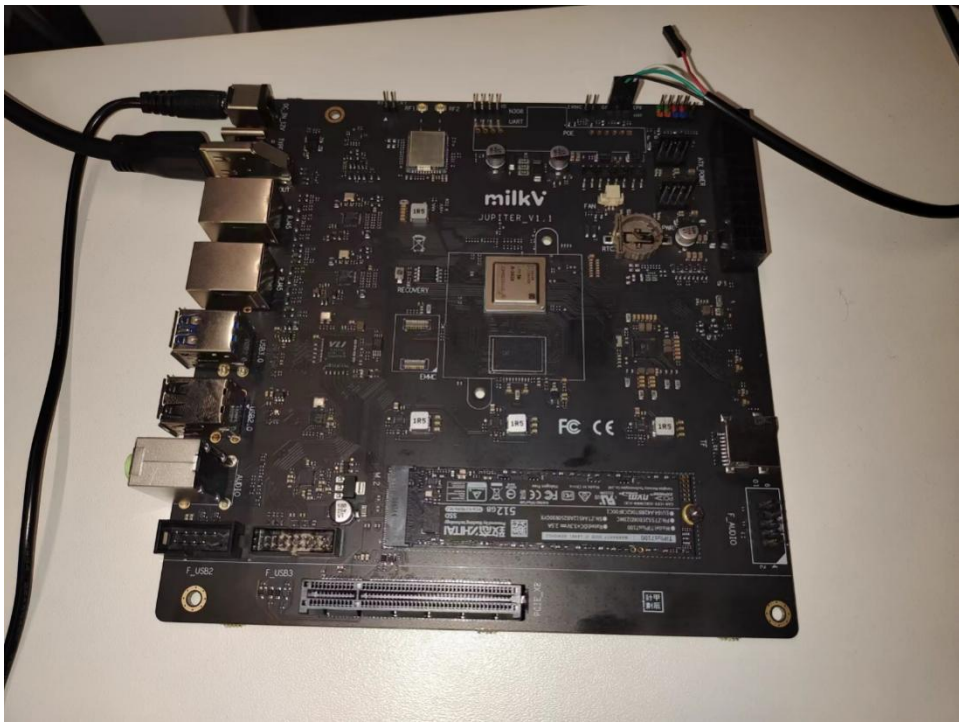

图 4-5 Milk-V Jupiter 实物图

1. TPM 模块模拟

由于该板卡未集成物理 TPM 芯片，本文通过 Linux 内核的 vtpm 模块与 swtpm 软件模拟器结合，创建虚拟 TPM 设备。

由于该板卡的 Linux 内核默认未开启 vtpm 选项，首先需要从官网下载对应版本内核，开启 vtpm 配置项后重新编译并替换内核。之后创建服务以 systemd 配置自动启动，手动创建 TPM 设备后，并可被系统识别为标准 TPM 设备。通过以下命令创建模拟 TPM 设备：

```
swtpm chardev \
    --tpmstate dir=/var/lib/swtpm/phys-host-tpm \
    --vtpm-proxy \
    --tpm2 \
    --log level=20
```

将其编写为 swtpm.service 系统服务，结合 vtpm_proxy 驱动模块创建设备节点，使内核能识别模拟 TPM。在系统启动后自动运行，实现 TPM 设备的自动挂载。

需要注意的是，由于该模拟方式基于用户态，在系统启动初期特别是 BIOS、linux 内核启动时，尚未挂载 TPM，因此无法对相应组件进行度量和扩展，无法实现真正的可信启动。因此本文之后将采用手动方式补充 PCR 扩展，模拟静态度量过程。

2. 大语言模型推理框架适配

为提升模型推理效率并充分发挥目标平台的硬件特性，本文对 llama.cpp 推理框架进行了适配优化，在 Milk-V Jupiter 开发板上实现了对 RISC-V 向量扩展（RVV） 与 GPU 加速（Vulkan）的联合支持。

首先，llama.cpp 在识别到 RISC-V 架构（RV64GCV）时，默认启用了 RISC-V 向量扩展，结合板卡 CPU 所具备的 RVV 执行能力，实现了模型张量操作的向量化加速。同时，针对该板卡支持 Vulkan 1.3 图形与计算接口 的特性，系统在编译阶段启用了 llama.cpp 的 --enable-vulkan 选项，并配置了 Vulkan ICD 与驱动环境，利用 GPU 并行计算资源进一步提升推理效率。

最终，开发板成功加载 DeepSeek 模型并完成推理，模型运行速度显著优于虚拟机平台，硬件加速效果明显，证明本技术方案可在真实硬件中运行，进一步验证了系统的可部署性、移植性和实用性。

4.1.4 平台搭建总结

表 4-4 平台搭建总结

平台类型	描述	是否支持 TPM	是否支持推理
用户端	x86_64 + 虚拟机 + 物理 TPM	是	否
虚拟云平台	QEMU + RISC-V + swtpm	是	是
实际平台	Milk-V Jupiter + 模拟 TPM	是（模拟）	是（支持 Vulkan）

虚拟 RISC-V 平台与 Milk-V Jupiter 平台的搭建完成后的基本配置如上表所示，后续实现均在此实验平台基础上完成。

4.2 核心模块开发

4.2.1 启动度量

在使用物理 TPM 芯片的系统中，静态度量通常由系统启动链自动完成。例如，UEFI BIOS、引导加载程序、内核映像等关键组件在加载时自动被测量并扩展至 TPM 的 PCR 中，形成一条不可篡改的“启动度量链”。

然而，本文所采用的 TPM 由软件模拟（swtpm 实现），其生命周期与操作系统启动流程相互独立，无法实现从启动初期自动介入。这导致系统在 TPM 模块与接口可用之前，无法实现 BIOS、GRUB、Linux 内核等组件的启动度量。

为弥补这一缺陷，本文设计并实现了一套手动度量与扩展机制，在系统启动完成后主动读取关键组件内容并手动扩展至 TPM 的 PCR 0~10 寄存器，构建等效的可信启动状态。该机制通过创建系统服务 extend-pcr.service 实现，包含以下度量内容：

表 4-5 PCR 度量文件

PCR 索引	内容描述
PCR0	UEFI 引导加载器
PCR1	设备树文件（dtb）
PCR2	平台硬件标识（model + serial）
PCR3	GRUB 模块目录
PCR4	GRUB 环境变量文件
PCR5	启动命令行参数（/proc/cmdline）
PCR6	系统唯一标识（machine-id）
PCR7	系统根证书（模拟 Secure Boot）
PCR8	Linux 内核映像文件
PCR9	Initramfs 初始文件系统
PCR10	IMA 策略文件（若启用 IMA）

度量流程如下所示：

1) 关键文件选取：根据启动链组成，选取系统中具有代表性的 10 项启动态元素作为度量对象；

2) 哈希计算：通过 SHA-256 对各文件内容或字符串信息进行单向散列运算；

3) PCR 扩展：TPM 命令工具（如 tpm2_pcrextend）将计算所得哈希值扩展写入指定 PCR；；

4) 服务化部署：通过 Systemd 创建守护服务，确保每次系统启动后自动执行上述流程。

通过该机制，即使使用 swtpm 模拟器，本技术仍可建立一个“等效可信启动链”，并将其作为后续 Quote 签名的一部分供用户端进行远程验证。PCR 值的一致性验证确保云端平台在“模型密钥解封前”处于完整可信状态。

4.2.2 远程证明

1. AK 证书签发

平台在完成启动度量后，由系统调用 tpm2-tools 指令创建 AK 密钥对，然后向第三方 CA 发起证书签发请求。发送内容包含：

- EK 和 AK 的公钥信息；
- AK 与 EK 之间的绑定说明（包含 AK Name 与 EK Digest）；
- 所属 TPM 平台的 EK 证书摘要（用于证明设备标识）；
- 非对称签名算法标识与密钥用途声明。

系统将上述信息封装为标准的 X.509 CSR（Certificate Signing Request），并提交至 CA。

为验证 AK 的所属关系是否真实可信，第三方 CA 不直接签发证书，而是先发起 TPM 标准认证流程 MakeCredential：

- CA 向平台下发一个加密挑战（Credential Blob），该挑战使用平台 EK 公钥加密，并绑定 AK Name；
- 挑战内容为一次性 nonce；
- 只有目标 TPM 内部，且持有匹配 EK 与 AK 的实体，才能解封该密文。

平台收到 Credential Blob 后，调用 TPM ActivateCredential 尝试解封挑战数据。若平台确实持有匹配的 EK 与 AK，TPM 可正确解封密文并返回 CA 当初设置的挑战值。平台将该解封结果回传给 CA，供其验证。若值一致，说明该 AK 确由目标 TPM 生成，绑定关系可信。CA 验证成功后，向平台返回 AK 的 X.509 证书，用于之后向客户端证明身份。

2. Quote 远程证明

本模块主要完成两个任务：平台生成 TPM Quote 报文，客户端验证报文合法性与度量状态。平台完成启动度量后，使用 TPM 的 AK 密钥对当前 PCR 值与客户端提供的随机数（nonce）进行签名，生成 Quote 报文。该报文连同 EK 证书、AK 证书、PCR 摘要和 nonce 一并打包，发送给客户端。

客户端验证流程包括：

- 检查 EK 与 AK 证书是否由受信 CA 签发；
- 验证 Quote 签名的合法性，确认签名由 AK 公钥生成；
- 对比 PCR 值是否符合可信基线；
- 检查 nonce 是否匹配，防止重放攻击。

若验证通过，客户端认为平台可信，方可释放模型密钥。否则中止后续流程。

4.2.3 运行时保护

为防止模型在运行过程中被恶意替换或泄露，本文设计了基于 TPM 的 文件完整性实时监控机制，通过 inotify + sha256sum + tpm2_pcrextend 联合构建文件状态验证链。动态保护机制由 runtime_protect.sh 实现，主要任务是：对关键服务文件、模型参数等进行完整性度量，并在运行过程中实时监听是否发生非预期修改。

1. 初始度量

系统进入运行态后，首先对预定义的关键文件如模型文件、推理引擎、服务脚本等进行一次性哈希度量。具体而言，系统采用 SHA-256 算法对目标文件集逐项计算摘要，并通过 tpm2_pcrextend 命令将每项摘要扩展至指定的 PCR 寄存器（PCR16）。该操作形成当前系统的“可信基线”。

2. 文件监听与 PCR 动态扩展

本技术通过内核级别的 inotify 文件事件机制，对关键文件集的目录树进行实时监控，监听包括 modify,attrib,close_write,move,delete 等文件操作事件。一旦检测到文件状态发生变化，系统立即对该文件重新计算哈希摘要，并与初始基线值进行比对。

若变更后的哈希值与初始值一致，则认为属于合法操作（如配置刷新或服务重启），系统将自动更新 PCR 值并记录更新事件；若不一致，则被判定为潜在的非法篡改行为，系统将进入响应流程。

3. 异常响应策略

当文件状态变更被识别为非法操作时，系统将自动执行一系列安全处置动作，包括：

- 立即强制终止服务进程（如模型推理服务程序）阻止模型继续运行；
- 使用 shred 安全擦除命令清除模型文件清除模型文件，防止泄露；
- 清空 TPM 上下文，避免密钥滥用；
- 生成带时间戳的安全事件日志，将异常事件、当前 PCR 值与处置结果写入日志文件 secure_audit.log 供后续审计与取证。

4.2.4 TPM-openssl 支持

为进一步保障系统中密钥的机密性与使用过程的不可导出性，本文引入了基于 OpenSSL 3.0 provider 机制开发的 tpm2-openssl 工具，通过将 OpenSSL 中的加解密操作迁移至 TPM 执行，实现“密钥不出 TPM”的数据保护目标。

通过配 provider 算法提供商为 tpm2，启用 TPM provider，使其可调用 TPM 内部密钥执行加解密。将几乎所有的加解密操作都迁移到 TPM 中执行，实现敏感数据始终保留在 TPM 中。避免其在主机内存或磁盘中出现，降低中间人攻击与提权攻击的风险，进一步提高系统的安全性。

尽管本方案高度依赖 TPM 提供的加密能力，但模型文件本身的解密操作未使用 TPM 执行，而是直接使用 openssl 软件进行加解密。其主要原因如下：

- TPM 主要用于密钥管理，以及面向密钥级的加解密操作（如少量数据块、对称密钥封装），不适合处理大体积文件流；
- 模型文件通常为数百 MB 至数 GB，不仅传输开销大，而且 TPM 内部性能难以支撑高吞吐量加解密。

4.2.5 公网服务部署

为便于模型服务的远程访问与交互展示，在可信验证完成后，将提供基于 HTTPS 的公网访问接口，通过安全隧道技术将本地模型服务映射至公网域名。该部署方式主要用于辅助演示与技术验证，不影响核心可信计算流程。

1. 服务启动与本地绑定

在完成启动度量、远程证明通过、并成功解封模型密钥后，系统即启动本地大语言模型推理服务。服务由 llama-server 提供，启动命令如下：

```
llama-server -m tpm/models/model.gguf --port 8080 --host 0.0.0.0
```

服务监听本地 8080 端口，提供基于 HTTPS 的文本推理接口，支持浏览器网页访问，便于演示交互。

2. Ngrok 内网穿透

为将本地服务映射至公网，系统配置了基于云端 Ngrok 隧道服务器的内网穿透服务。部署步骤如下：

- 购买在 Ngrok 云服务器服务
- 将本地端口 8080 映射至绑定域名 deepseek.sv6.tunnelfrp.com
- 在本地启动对应的 Ngrok 隧道服务
- 使用隧道服务商提供的默认 HTTPS 证书，实现 TLS 加密通信。

该方式无需在本地设备开放公网端口，也无需独立申请 TLS 证书，即可快速构建加密传输通道，实现安全的公网访问体验。

需要明确的是，公网服务本身不构成可信计算技术的信任链组成部分，也不参与 TPM 启动度量、远程证明流程或密钥解封逻辑。该模块作为部署与演示的辅助功能，体现了本技术方案的实际可用性与远程交互能力，但其安全性始终受限于原有的可信系统机制。

4.3 本章小结

本章围绕所设计的基于硬件可信根的可信模型保护技术，在软硬件平台上完成了全流程的功能实现与部署工作。首先，基于 Buildroot 与 Milk-V Jupiter 实物平台分别构建了 RISC-V 虚拟与物理运行环境，完成 TPM 模拟器、模型推理框架等关键组件的集成。随后，依次实现了可信启动、远程证明、运行时保护、TPM-OpenSSL 支持与公网服务部署等功能模块，确保系统具备从平台认证到模型运行的完整可信链。

上述工作为后续的功能验证、安全性评估与性能测试奠定了坚实基础。下一章将对本技术进行全面测试，评估其在真实环境中的实际效果与鲁棒性。

第五章 技术测试

5.1 测试概述

为验证本基于硬件可信根的数据保护技术在实际环境中的可行性与有效性，本章围绕系统核心功能、安全性机制与性能表现三个维度展开测试评估。由于 RISC-V 虚拟实验平台主要作为前期算法验证，并未充分实现所有系统功能，因此测试工作集中在 Milk-V Jupiter 物理平台上进行。测试主要包括以下几个方面：

- 功能测试：验证关键模块（如启动度量、远程证明、模型加载、运行时保护等）是否正确实现其预期功能；
- 安全性测试：通过模拟篡改攻击、非法调用与平台伪造等行为，验证系统是否具备足够的抗破坏与密钥保护能力；
- 性能测试：评估系统在可信计算机制参与下的启动延迟、远程证明响应时间以及模型推理性能等关键指标。

5.2 功能测试

功能测试旨在验证各核心模块在实际平台环境下是否能够按预期工作，在理想状态下能否正确实现预期功能。本节围绕四项核心功能展开测试，测试用例如表 5-1 所示。

表 5-1 功能测试用例

测试编号	功能模块	测试目标	测试方法	预期表现
F01	启动度量	验证启动链各阶段是否正确被 TPM 度量并写入 PCR	通过 tpm2_pcrread 读取 PCR[0~7]，观察是否更新	PCR[0~7]值变化符合预期
F02	远程证明	验证远程证明机制能否成功	运行远程证明脚本，观察输出	AK、Quote、PCR 校验通过，平台可信
F03	模型启动	验证模型文件在解密后能正确加载与服务启动	启动 llama-server -m model.gguf，浏览器/脚本访问测试返回结果	可访问服务，模型正常响应
F04	运行时保护	验证服务运行期间是否能检测模型被篡改	修改 model.gguf 内容，观察是否触发终止 + PCR[16] 扩展 + 日志	服务终止，日志记录，PCR[16]扩展

5.2.1 F01-启动度量

（1）测试目标：验证启动链各阶段是否正确被 TPM 度量并写入 PCR

（2）测试方法：通过 tpm2_pcrread 读取 PCR[0~7]，观察 BIOS/Kernel 哈希是否更新

（3）实验结果及分析：如下图所示，PCR 值发生变化，启动度量测试正确完成。

```
→ tpm tpm2_pcrread sha256:0,1,2,3,4,5,6,7
  sha256:
    0 : 0xC76CC87728A19E6BC5687EDD4E474639C48433D46F063F97FD1537F571E7767E
    1 : 0x69258002EFABA63B61B78A29BF72AAB5310317B9774D2A33A74AF7F3E47A6677
    2 : 0x1DF2B68FBE265D5C1B7D66719DFD089D40BDB8D99032595B518F26A9FC390C3C
    3 : 0x2CFC8942E95BC3635FA1DCFDBB45C8A97617FC5674658A406A6CACFCE476E8DA
    4 : 0xB2D8EF726569D41AE78AE440A4432DBADBE22C97093E12C060B2D6D7DE830F34
    5 : 0x9C09A876DA97F4D3EA45F6E420CB85E74AAC14DB4A15D3521ACACD654195B17B
    6 : 0x53F3B3AEC5B68BF941F694157FC1644C3B685C709E23229000E8EADF82A3F7D5
    7 : 0x4671D965755CD8FE8A01C081149E5A0A4E726610AFA8AA521EC6B6C20A76DCBA
```

图 5-1 启动度量实验结果

5.2.2 F02-远程证明

（1）测试目标：验证远程证明机制能否成功

（2）测试方法：在云端和和客户端分别运行 cloud.sh, client.sh 远程证明脚本，观察中间输出

（3）实验结果及分析：如下图所示，云端和客户端成功完成远程证明挑战。云端首先与 CA 完成了 AK 证书的签发，然后云端与客户端完成了 Quote 和 PCR 的验证。最后客户向云端发回模型密钥，云端成功解封。远程证明测试正确完成。

```
→ tpm ./cloud.sh
[Cloud] EK 已存在
[Cloud] 从 TPM 读取 EK 证书成功
[Cloud] 创建 AIK...
persistent-handle: 0x81010002
action: persisted
[Cloud] 向第三方 CA 请求签发 AK 证书...
[Cloud] 等待第三方 CA 发起 AK 验证挑战...
[Cloud] 执行 activatecredential 解封挑战...
8371976744484b3f81a90cc8d46a5d724fd52d76e06520b64f2a1da1b331469aa
[Cloud] activatecredential 成功，AK 属于 TPM 内部
[Cloud] 等待第三方 CA 签发 AK 证书...
[Cloud] 等待客户端发送 Nonce...
[Cloud] 生成 Quote...
[Cloud] 发送 Quote 和证书给客户端...
[Cloud] 等待用户验证Quote...
[Cloud] 验证成功，接收模型密钥
[Cloud] 解密模型...
[Cloud] 模型解密完成：models/model.guff
[Cloud] 清理临时文件...
```

图 5-2 云端远程证明输出

```
star@star-virtual-machine:~/tpm$ ./client.sh
[Client/CA] 等待 EK 证书...
/home/star/tpm/ca-test/ekcert.pem: OK
[Cleint/CA] EK 验证证书成功
[Client/CA] 等待云端响应...
[Client/CA] TPM AK 验证通过
Certificate request self-signature ok
subject=CN = TPM-AK
[Client] 等待Ek,Ak,Quote...
ekcert.pem: OK
akcert.pem: OK
[Client] EK,AK证书验证成功
[Client] 验证Quote签名和PCR...
[Client] 验证PCR值...
[CLlient] Quote签名和PCR验证成功
[Cliennt] End
[Client/CA] 清理临时文件...
```

图 5-3 客户端远程证明输出

5.2.3 F03-模型加载

（1）测试目标：验证模型文件在解密后能正确加载与服务启动

（2）测试方法：在云端执行 llama-server -m model.gguf --port 8080 --host 0.0.0.0 命令，登录网址 https://deepseek.sv6.tunnelfrp.com/查看，并与模型进行交互

（3）实验结果及分析： 如下图所示，llama.cpp 正确启动，可通过公网域名访问模型服务。与 Deepseek 对话，得到正常输出，且显示其思考过程。 模型加载测试正确完成。

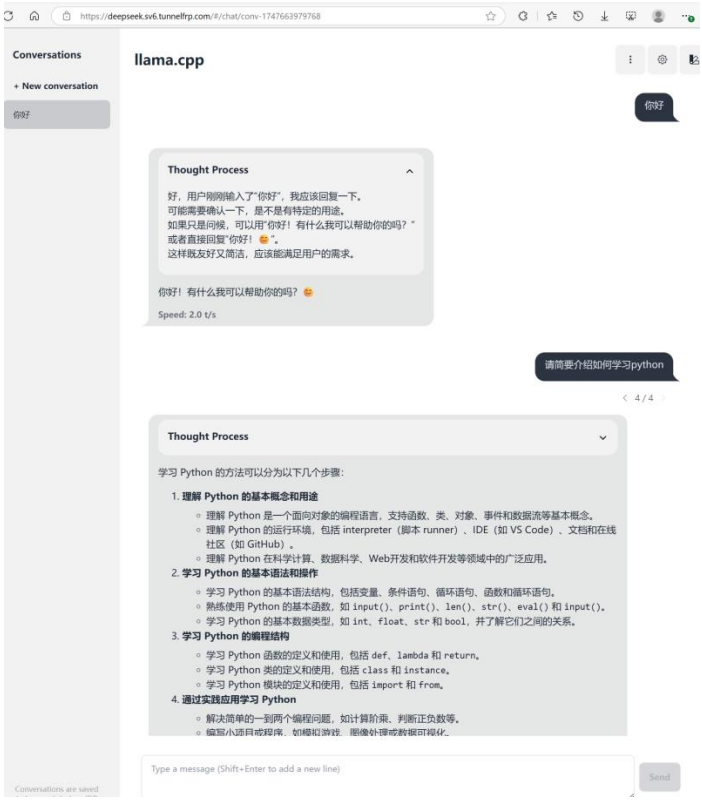

图 5-4 llama.cpp 网页访问界面

5.2.4 F04-运行时保护

（1）测试目标：验证服务运行期间是否能检测模型被篡改

（2）测试方法：修改 model.gguf 内容，观察是否触发终止 + PCR[16] 扩展 + 日志

（3）实验结果及分析：如下图所示，通过向 model.gguf 写入内容模拟篡改模型。写入后，运行时保护程序立即检测到文件修改且 hash 不匹配，终止服务并清除模型文件。查看审计日志文件 secure_audit.log，可以看到其中记录了本次的文件 hash 不匹配错误，且在每一条日志下都附有当时的 PCR16 值。读取当前 PCR16，与日志中的 PCR 值相匹配。综上可知，运行时保护测试成功。

图 5-5 模拟篡改模型参数

图 5-6 运行时保护机制

图 5-7 审计日志文件

图 5-8 PCR16

5.3 安全性测试

本技术的安全性测试旨在验证其在面对实际攻击场景时是否具备有效的检测、防御与隔离能力，确保模型密钥不会在非可信状态下泄露，模型运行过程不会被恶意篡改。

本节从三个典型攻击面出发设计测试用例，包括：平台伪造攻击、模型篡改攻击与非法解密尝试，并分析系统的响应行为与安全效果。测试用例如下表所示。

表 5-2 安全性测试用例

测试编号	攻击类型	测试目标	攻击方法	预期行为
S01	平台伪造	验证非法设备能否通过远程证明	使用另一台 RISC-V 虚拟机模拟 TPM，伪造 Quote 报文	客户端无法验证 EK、AK 证书，拒绝解封
S02	篡改启动链组	验证非可信平台能否通过远程证明	修改启动链组件(如 UEFI、Kernel)	客户端检测到 PCR 不匹配，拒绝密钥交付
S03	篡改模型文件	验证运行时能否发现模型文件被修改	启动服务后执行 echo attack >> model.gguf	系统终止服务，扩展 PCR[16]，日志记录

5.3.1 S01-平台伪造

（1）测试目标：验证缺少 EK 或 AK 证书的非法设备能否通过远程证明获取密钥

（2）实验步骤：在 cloud.sh 脚本中，增加一行修改 akcert.pem 证书，模拟平台错误，导致拥有错误的 AK 证书。EK 证书错误可同理进行模拟，不再赘述。然后运行远程证明机制，查看校验结果。

（3）实验结果及分析：如下图所示，云端篡改 akcert.pem 后，客户端对 AK 证书的校验失败，拒绝解封，远程证明停止。

图 5-9 云端篡改 AK 证书

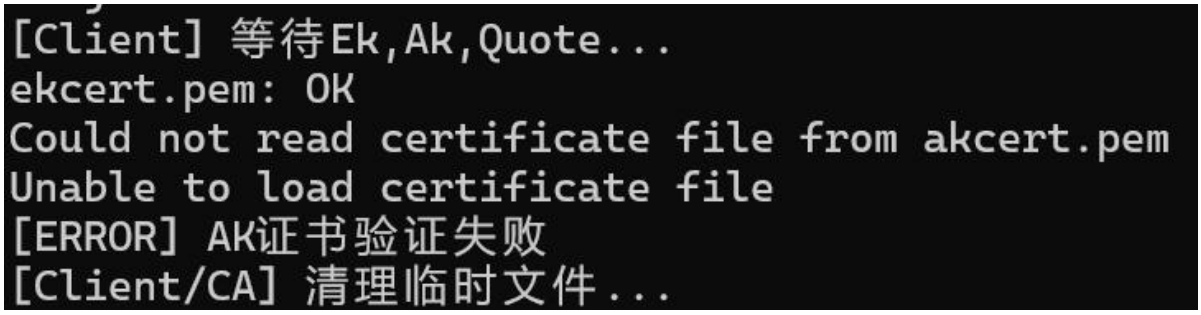

图 5-10 客户端 AK 证书校验失败

5.3.2 S02-篡改启动链组件

（1）测试目标：验证篡改启动链组件后，PCR 状态不一致的非可信平台，能否通过远程证明获取密钥；

（2）实验步骤：通过篡改手动模拟度量文件 extend-pcr-all.sh 中对 PCR0 的度量，模拟对启动链中关键组件的篡改。

```
tpm2_pcrextend 0:$ALGO=$(echo "AttackUEFI" | openssl dgst -$ALGO -binary | xxd -p -c 32)
```

篡改 extend-pcr-all.sh 后，重新启动系统，将 PCR 重置，使其度量被篡改的组件，对比篡改前后的 PCR0 的变化。然后运行远程证明脚本，查看客户端的 PCR 校验结果。

（3）实验结果及分析：由图 5-11 和图 5-12 对比篡改前后的 PCR10，可知已实现对 UEFI 组件的篡改。由图 5-13 可知，客户端校验出 PCR 值不匹配，终止远程证明。

```
→ tpm tpm2_pcrread sha256:0
  sha256:
    0 : 0xC76CC87728A19E6BC5687EDD4E474639C48433D46F063F97FD1537F571E7767E
```

图 5-11 篡改前的 PCR0

```
→ ~ tpm2_pcrread sha256:0
  sha256:
    0 : 0xE45D0C2091088AF3946FB7ED7E1B8E6F43629574223D4B2D9B1EC7FCFC49BA6C
```

图 5-12 篡改后的 PCR0

```
[Client] 等待Ek,Ak,Quote...
ekcert.pem: OK
akcert.pem: OK
[Client] EK,AK证书验证成功
[Client] 验证Quote签名和PCR...
[Client] 验证PCR值...
[ERROR] PCR值不匹配！终止远程证明
[Client/CA] 清理临时文件...
```

图 5-13 客户端 PCR 不匹配

5.3.3 S03-篡改模型文件

该测试用例与“F04-运行时保护”实际上是同一个测试，都是对模型文件进行篡改后，验证运行时保护的能力。由于 F04 已对此进行描述，此处不再赘述。

5.4 性能测试

为评估系统在引入本可信计算机制下的运行开销与响应效率，本文设计了一套完整的性能测试脚本 test_performance.sh，对系统启动、远程证明、模型解密、运行期初始

化与模型推理等阶段进行逐项采样。

1. 测试方法

使用 date 与 time 命令对各阶段执行时间进行采样，编写性能测试脚本 test_performance.sh，执行 3-5 次取平均值作为评估数据。

2. 测试数据

其中一次测试截图如下，可以得到各阶段耗时。通过多次测试，取平均值，得到表 4-6 数据。

```
[Cloud] 解密模型...

real    0m46.282s
user    0m39.739s
sys     0m6.206s
[Cloud] 模型解密完成: models/model.guff
[Cloud] 清理临时文件...
[RESULT] cloud.sh 耗时: 64779 ms
```

图 5-14 cloud. sh 耗时

```
[RESULT] 初始度量耗时: 29343 ms

[RESULT] 启动前总耗时: 94141 ms

[TEST] 启动 llama-cli 模型推理...
```

图 5-15 初始度量耗时

```
llama_perf_sampler_print:    sampling time =     415.83 ms /   348 runs   (    1.19 ms per token,   836.87 tokens pe
r second)
llama_perf_context_print:        load time =    4400.72 ms
llama_perf_context_print: prompt eval time =    2041.88 ms /     8 tokens (  255.24 ms per token,     3.92 tokens pe
r second)
llama_perf_context_print:        eval time =  190877.51 ms /   340 runs   (  561.40 ms per token,     1.78 tokens pe
r second)
llama_perf_context_print:       total time =  370763.84 ms /   348 tokens
Interrupted by user
==== 性能测试完成 ====
```

图 5-16 模型推理速度

表 5-3 各阶段平均耗时

序号	测试阶段	操作描述	平均耗时
1	远程证明	AK 密钥生成，AK 证书签发， EK、AK、Quote 和 PCR 的验证	20s
2	模型解密	使用 OpenSSL AES 解密模型	45s
3	初始度量（运行时保护初始化）	对关键模型/脚本文件进行哈希并扩展 PCR	30s
4	启动 llama 模型推理引擎	启动 llama-cli 推理引擎、加载 GGUF 模型至执行就	4.5s

序号	测试阶段	操作描述	平均耗时
5	单轮模型推理（32 tokens）	执行推理输出响应	17s

3. 实验结果分析

表 5-3 展示了系统在五个关键阶段的平均耗时，覆盖从平台验证到模型推理的完整流程。

- 远程证明阶段：包括 AK 密钥生成、AK 证书签发以及 EK、AK、Quote 和 PCR 状态的验证，整体耗时约 20 秒。该流程为一次性初始化过程，主要开销来源于 TPM 签名操作和 X.509 证书链验证，对系统启动延迟产生一定影响，但在会话内无需重复执行。
- 模型解密阶段：采用 OpenSSL 对量化后的 GGUF 格式大模型进行 AES 解密，平均耗时为 45 秒。由于模型文件体积较大（通常达数个 GB），解密属于系统初始化过程的主要瓶颈之一，但只需在首次加载时执行。
- 初始度量阶段：系统对模型文件、推理引擎与服务脚本等关键组件进行哈希计算，并扩展至 TPM PCR 寄存器，平均耗时为 30 秒。该过程确保运行环境的完整性，主要由大文件哈希计算与 PCR 写入操作构成。
- 推理引擎加载阶段：调用 llama-cli 启动模型推理服务并加载 GGUF 模型至内存，平均耗时仅 4.5 秒，表明模型在 RISC-V 架构下具有良好的加载效率，适用于轻量部署场景。
- 单轮模型推理（32 tokens）：在完成上下文加载后生成 32 个 tokens，平均耗时约为 17 秒。该指标可反映实际推理响应能力，体现了系统在有限算力下的可用性，有一定的推理能力。

综上，系统整体初始化时间控制在合理范围内，可信计算机制主要集中在启动前期开销，运行阶段响应延迟可接受。实验验证本方案在安全性增强的同时，仍具备较好的实用性能与部署可行性。

5.5 本章小结

本节通过一系列功能性测试、安全性实验与性能数据分析，验证了本技术在“启动可信、密钥受控、运行守护、可追溯”方面的目标已成功实现。

通过构建标准化测试用例，分别验证了系统从启动度量、远程证明、密钥绑定、模型加载到运行期完整性监控的各项核心功能模块，测试结果表明各功能均符合设计预期。安全性测试涵盖了平台伪造、启动链篡改、模型文件篡改等典型攻击场景，验证了系统在身份认证、平台状态绑定、运行期异常检测等方面具备较强的防护能力。性能测试则评估了远程证明延迟、模型加载时间与推理响应效率等关键指标，结果显示本文所提出的技术在确保安全性的前提下仍能实现可接受的服务响应。

综上，本技术方案在 RISC-V 实物平台上表现出良好的可部署性、安全性与实用性，具备可信模型服务的实际落地能力。

第六章 总结与展望

6.1 本文工作总结

本文围绕大语言模型在云端部署中面临的安全威胁，设计并实现了一套基于硬件可信根的数据保护技术，构建了面向 RISC-V 架构的可信计算解决方案。系统以 TPM 2.0 模块为核心，结合用户端、云平台与第三方 CA 的三方协同机制，围绕“启动阶段可信度量—远程证明—运行时保护”三大核心环节，实现了贯穿模型生命周期的完整安全防护链。

本文的主要工作可概括为以下几个方面：

1. 技术方案设计：结合可信计算技术与模型保护需求，设计了包括“启动度量、远程证明、运行时保护”的贯穿模型全生命周期的安全防护，明确了用户端、云平台与第三方 CA 的角色分工与协同机制，最终构建了基于硬件可信根的的数据保护技术。

2. 功能模块实现：本文完成了系统各关键功能模块的实现工作，涵盖从平台初始化到模型服务的完整可信流程，包括启动度量、远程证明、模型加载、运行时保护、TPM-OpenSSL 集成、公网服务部署等，构成了模型从“可信启动—密钥交付—安全推理—服务开放”的闭环式保护路径，全面覆盖模型生命周期内的关键安全环节。

3. 多平台部署与实验验证：在 QEMU 构建的 RISC-V 虚拟平台与 Milk-V Jupiter 开发板上完成部署，验证了技术的可运行性与平台适配能力，并通过实验评估了技术的功能有效性、安全性与性能开销。结果表明本技术在保障模型安全的同时维持了良好的运行效率。

综上所述，本文系统性地构建了一个适配 RISC-V 平台的可信模型运行环境，验证了 TPM 在大模型安全部署中的实用价值，并为构建开源、自主、安全的 AI 推理基础设施提供了技术支撑与实验依据。

6.2 不足与改进方向

尽管本文实现了基本的设计目标，但仍存在以下不足与改进空间：

1. TPM 模拟环境局限性：当前实验中采用的是 swtpm 模拟器，虽然其符合 TPM 2.0 规范，但无法真正参与系统启动初期的硬件级度量，需通过用户态脚本手动完成 PCR 扩展，与真实情况差别较大。且软件实现方式的效率较低，无法准确计算性能损

耗。未来考虑集移植到开源高性能 XiangShan RISC-V 处理器中，同时集成个性化 RoT 固件代码或 OpenTitan 开源可信根，对所需 TPM 功能进行针对性优化[16]。

2. RISC-V 架构特性未被充分挖掘：尽管本系统部署于 RISC-V 架构平台，但主要以通用软件栈与模拟器为基础，尚未深入利用 RISC-V 架构的开源可定制性进行深度适配与裁剪。例如，系统尚未基于可信计算需求对指令集扩展进行优化选择，也未定制安全启动链、最小可信计算基（TCB）或针对特定安全场景（如边缘端 AI 推理）进行轻量化系统构建。后续工作可结合 RISC-V SoC 设计流程，对可信功能进行软硬协同裁剪与定制，实现从芯片到软件的垂直整合可信计算体系。

3. 缺少内存控制：由于 TPM 本身不具备运行时内存加密与访问隔离能力，缺乏有效的内存级保护机制。系统无法防止攻击者通过物理侧信道（如 DMA 攻击、缓存窥探）或逻辑漏洞（如提权、内核注入）窃取驻留在内存中的明文模型参数或推理中间结果。未来可结合内存加密技术或可信执行环境（TEE）保护内存区域，实现真正的全周期模型保护。

参考文献

[1] Baezner M, Robin P. Stuxnet[R]. ETH Zurich, 2017.

[2] Chen Yanzuo, Liu Zhibo. The Devil is in the (Micro-) Architectures: Uncovering New Side-Channel and Bit-Flip Attack Surfaces in DNN Executables[P]. Available from: https://www.blackhat.com/eu-24/briefings/schedule/#the-devil-is-in-the-micro--architectures-uncovering-new-side-channel-and-bit-flip-attack-surfaces-in-dnn-executables-42018.

[3] 冯登国,秦宇,汪丹,等.可信计算技术研究[J].计算机研究与发展,2011,48(08):1332-1349.

[4] Eldefrawy K, Tsudik G, Francillon A, et al. Smart: secure and minimal architecture for (establishing dynamic) root of trust[C]//Ndss. 2012, 12: 1-15.

[5] Trusted Computing Group. TPM 2.0 Library Specification[S]. 2024. Available: https://trustedcomputinggroup.org/resource/tpm-library-specification/.

[6] Arthur W, Challener D, Goldman K. A practical guide to TPM 2.0: Using the new trusted platform module in the new age of security[M]. Springer Nature, 2015.

[7] Sahita R, Shanbhogue V, Bresticker A, et al. CoVE: Towards confidential computing on RISC-V platforms[C]//Proceedings of the 20th ACM International Conference on Computing Frontiers. 2023: 315-321.

[8] Bajikar S. Trusted platform module (tpm) based security on notebook pcs-white paper[J]. Mobile Platforms Group Intel Corporation, 2002, 1: 20.

[9] 梁元.基于云计算环境下的可信平台设计[D].电子科技大学,2013.

[10] Intel. sgx-pytorch[EB/OL]. (2022-05-15)[2023-10-01]. https://github.com/intel/sgx-pytorch.

[11] Meza A, Restuccia F, Oberg J, et al. Security verification of the opentitan hardware root of trust[J]. IEEE Security & Privacy, 2023, 21(3): 27-36.

[12] Nie C. Dynamic root of trust in trusted computing[C]//TKK T1105290 Seminar on Network Security. 2007.

[13] Patterson D, Waterman A. The RISC-V Reader: an open architecture Atlas[M]. Strawberry Canyon, 2017.

[14] Guo D, Yang D, Zhang H, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning[J]. arXiv preprint arXiv:2501.12948, 2025.

[15] Tramèr F, Zhang F, Juels A, et al. Stealing machine learning models via prediction {APIs}[C]//25th USENIX security symposium (USENIX Security 16). 2016: 601-618.

[16] Wang K, Chen J, Xu Y, et al. XiangShan: An Open-Source Project for High-Performance RISC-V Processors Meeting Industrial-Grade Standards[C]//2024 IEEE Hot Chips 36 Symposium (HCS). IEEE Computer Society, 2024: 1-25.

致谢

首先，衷心感谢[redacted]老师在整个毕业设计过程中的悉心指导，从选题、研究、实验到论文撰写，每一步都给予了我莫大的帮助和支持。在做毕设的过程中，老师也时刻注意培养我的学术思维，锻炼我发现问题、解决问题的能力，为日后迈入学术界打下坚实基础。同时也要感谢[redacted]老师在毕设过程中的帮助，感谢他提出的宝贵建议，指导我顺利完成毕设。

其次，感谢师兄在研究过程中特别是实验阶段的帮助，使我在研究过程中少走了很多弯路。感谢同学和室友，感谢他们在我学习和生活中的帮助，共同学习、共同进步。

最后，感谢我的父母和姐姐，始终默默支持我，感谢他们源源不断的鼓励和永不停息的爱意，是他们让我得到最强大的精神动力。