

特色专题

基于大模型的密码模块识别工具研究

魏鹏坤, 刘超*, 胡海波

摘要 密码模块识别是金融银行业抗量子密码迁移的起点,其结果构成密码资产清单汇编与迁移优先级排序的基础。针对银行信息系统及操作系统源码规模大、语言多样、封装层次深、依赖链复杂等特点,研究一种基于大模型的密码模块识别工具。该工具以全量源码和依赖信息为输入,通过数据接入与预处理、调用图构建、密码语义识别、可信推理与反思、结果汇编与可视化 5 类智能体协作完成识别,输出可追溯至源码语句的证据单元。该工具以开源模型 Qwen3-8B 为基座实现,不作参数微调。在龙蜥操作系统内核源码构建的 400 个文件级样本上,精确率为 97.50%,召回率为 96.06%。与基于模式匹配的开源密码发现工具的对照实验表明,对照工具未能发现的 84 个真实密码模块中该工具识别出 76 个;消融实验表明,调用图构建与可信推理反思 2 项机制协同将召回率由 20.69% 提升至 96.06%。上述结果表明,语义层面的判据可将密码模块的识别范围由代码文本中显式可见的实现扩展至以注册与封装形式组织的系统级代码。

关键词 密码模块识别;大语言模型;多智能体;抗量子密码迁移;密码资产清单

随着量子计算技术的发展,以大数分解和离散对数困难问题为安全基础的第一代公钥密码算法面临结构性风险。RSA(Rivest-Shamir-Adleman)、Diffie-Hellman、ECC(elliptic curve cryptography)以及与 ECC 同源设计的 SM2 等算法,一旦遭遇密码分析相关量子计算机与 Shor 算法的有效攻击,其支撑的身份认证、密钥协商、数字签名、证书信任链和软件供应链完整性均可能受到影响^[1]。金融银行业信息系统对上述密码能力依赖广泛,是抗量子计算密码(post-quantum cryptography, PQC)迁移的重点领域。

近年来,国内行业实践强化了“先识别、后评估、再迁移”的工程逻辑^[2-3],银行业抗量子敏捷迁移将密

码模块识别置于密码资产清单汇编、量子安全风险评估与迁移实施之前,构成迁移流程的起点^[4]。在国际标准化方面,美国国家标准与技术研究院(National Institute of Standards and Technology, NIST)于 2024 年正式发布 FIPS 203、FIPS 204 与 FIPS 205,分别给出 ML-KEM、ML-DSA 与 SLH-DSA 等 PQC 标准^[5-7],迁移工作由此进入实施阶段。与此同时,对 Signal PQXDH 协议的形式化验证表明,PQC 接入真实系统不仅涉及算法替换,还需处理协议兼容、KEM(key encapsulation mechanism)绑定与混合部署等工程细节^[8]。因此,迁移前需要精确到代码位置与调用路径的密码模块清单,以支撑协议绑定与接口适配等具体位置的改造。

金融银行业信息系统业务连续性要求高、历史系统多、密钥与证书使用周期长、外部接口复杂,尤其

重庆大学信息物理社会可信服务计算教育部重点实验室(CPS-DSC),重庆 400044

收稿日期: 2026-05-13; 修回日期: 2026-08-03

基金项目: 国家重点研发计划项目(2023YFC3305500, 2025YFE0218104)

作者简介: 魏鹏坤, 硕士研究生, 研究方向为智能软件工程, 电子信箱: weipengkun@stu.cqu.edu.cn; 刘超(通信作者), 副教授, 研究方向为智能软件工程, 电子信箱: liu.chao@cqu.edu.cn

引用格式: 魏鹏坤, 刘超, 胡海波. 基于大模型的密码模块识别工具研究[J]. 科技导报, 2026, 44(16): 107-117; doi:10.3981/j.issn.1000-7857.2026.05.00063

需要先把现有密码模块“找准、找全、找清楚”^[9]。银行信息系统的磁盘加密、传输层安全与完整性校验等能力均由操作系统内核提供,内核不依赖应用层密码库,其密码能力由内核内部的密码框架统一注册,再由调用方经接口获取,密码逻辑分散于注册代码、框架接口与硬件适配层之间^[3]。这些接口又经过二次封装、宏定义、适配层重命名或历史版本继承,算法名称与标准接口在代码中已不再显式可见^[10-11]。现有面向源码的密码资产发现工具以密码库应用程序编程接口(application programming interface, API)调用点和算法名称等词法特征为判据^[12-13],对这类模块的覆盖能力有限。

基于上述需求,本文围绕“基于大模型的密码模块识别工具”展开研究,提出如何利用大语言模型的代码语义理解能力、跨语言分析能力和证据生成能力,对复杂源码与依赖网络中的密码模块实施自动化识别与可解释输出。

1 PQC 迁移中的密码模块识别概述

1.1 识别目标:从“算法存在”到“模块、资产与业务关联”

在 PQC 迁移背景下,密码模块识别为迁移决策提供依据,其结果至少包含 5 类信息^[4,14]:一是密码算法或能力类型,涵盖加密、解密、摘要、签名、验签、密钥交换、密钥派生、证书解析、随机数生成和密钥管理等;二是代码级位置,包括文件路径、函数名称、调用语句与依赖组件;三是模块边界,即该代码属于底层算法实现、密码库适配层、协议栈、业务封装层还是测试代码;四是业务关联,即该模块支撑的数据类型、业务接口与调用路径;五是供应链信息,包括模块来源、组件版本与可替换性。这种识别目标具有直接的治理意义^[4],识别结果指明某个模块位于交易签名、证书验证或软件分发链路之中,并被多个业务系统复用,可为后续量子安全风险评估提供优先级依据。

1.2 识别范围:全栈、多语言与供应链

操作系统的密码能力分布于系统全栈^[3-4],涵盖引导与启动阶段的安全启动与内核模块签名,内核层的完整性度量、文件系统加密与传输层安全(transport

layer security, TLS)连接,用户态的登录认证、证书管理与应用包签名,以及经由开源组件与硬件驱动引入的第三方密码实现。这些层次在实现上包含 C/C++、Shell、Python、汇编与配置脚本等多种代码形态,密码能力还会沿依赖关系在组件之间传播,识别范围因而须覆盖多语言代码与供应链依赖。传统静态分析工具往往在单语言、单构建系统或接口命名规范的条件表现较好,但面对跨语言封装、宏展开、条件编译和插件式加载时容易丢失上下文^[15-17]。基于大模型的方法能在统一语义空间中理解函数名称、注释、参数与上下游调用,判定流程不随语言与层次改变,并可沿依赖关系向上下游追溯,与上述识别范围相适配。

1.3 识别产物:面向 C-BOM 和量子风险评估的证据单元

密码模块识别的结果服务于密码资产清单(cryptography bill of materials, C-BOM)建设,并作为量子安全风险评估的输入^[4,14]。C-BOM 逐条记录系统中每一处密码实现所使用的算法、承担的安全功能及其所在位置,是迁移范围界定与优先级排序的事实基础。清单中的每个条目都须给出判定依据,供人工核验其可信程度,识别结果因此不能停留在疑似命中列表中^[11,18]。为此,本文提出将识别结果组织为“证据单元”,每个证据单元以代码位置为基础,包含算法与能力标签、置信度、命中依据、调用上下文、模块角色、依赖关系和建议处置方式。向上,同一组件与同一系统的证据单元汇聚为模块级、组件级与系统级密码资产清单;向下,每一条清单条目可追溯至具体的源码语句与函数调用。

1.4 相关工作

面向源码的密码资产发现工作可分为 3 类。第 1 类是商业密码资产发现平台,如 SandboxAQ AQtive Guard、IBM Quantum Safe Explorer 与 CryptoNext COMPASS 等,此类产品按企业授权分发,其扫描引擎与规则库不公开,许可条款亦限制公开发布性能评测结果,难以作为可复现的学术对照。第 2 类是基于语法树或数据流分析的工具,代表工作为由后量子密码学联盟(Post-Quantum Cryptography Alliance, PQCA)维护的 CBOMkit 及其核心组件 Sonar-cryptography^[12],通过匹配已知密码库的 API 调用点识别密码资产,需

为每种语言分别实现语法解析与密码库 API 规则,目前覆盖 Java、Python 与 Go 这 3 种语言生态。第 3 类是基于模式匹配的工具,代表工作为开源工具 Crypto-Scan^[13],可直接在代码文本上识别算法名称与密钥材料,内置 90 余条检测模式,不以特定密码库的存在为前提,亦不依赖语言特定的解析器。

与上述工作并行的另一方向是密码 API 误用检测,代表工作包括 CryptoGuard^[19]与 CogniCrypt^[20],相应公开基准有 CryptoAPI-Bench^[21]与 MASC^[22]。该方向在文献中常与密码资产发现一并讨论,其评测对象是密码 API 的调用是否符合安全规范。本文的任务是

判定源码文件是否承载密码功能,二者的样本组织与评价指标不可直接比较,本文因此自建评测数据集。

2 基于大模型的密码模块识别总体框架

本文工具采用多智能体协同架构,以全量代码分析为基础,以调用图为骨架,以大模型语义判断为核心,以证据反思和人工复核为质量保障,输出密码模块清单、调用路径、风险标签和可视化结果。工具的总体框架如图 1 所示。

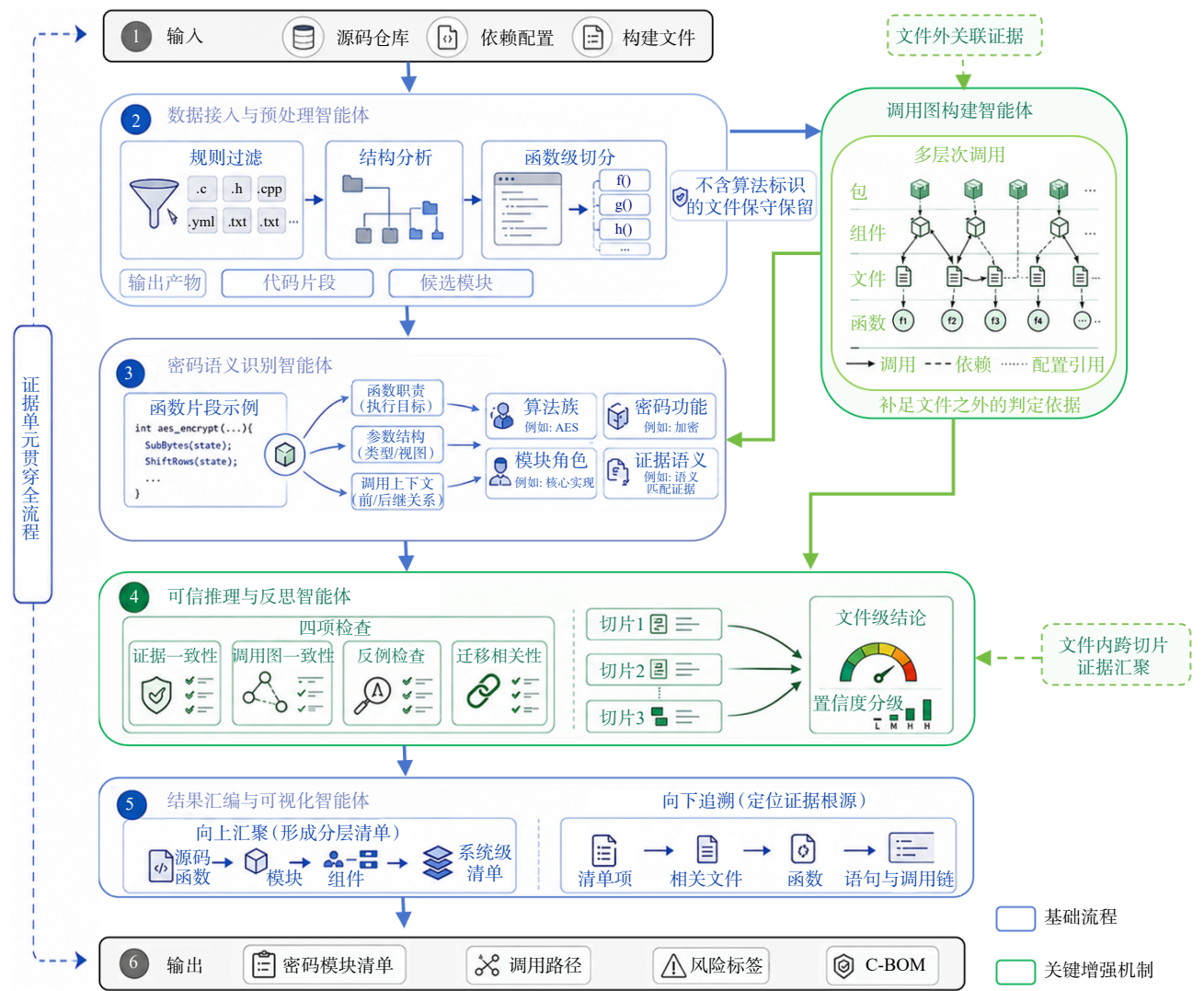


图 1 基于大模型的密码模块识别工具总体框架

2.1 数据接入与预处理智能体

数据接入与预处理智能体负责将操作系统、应用系统或开源组件的源码仓库转化为可分析对象。其

主要任务包括识别项目语言与目录结构,读取构建文件和依赖配置,过滤二进制文件、生成文件和无关资源,保留安全相关配置、证书样例和脚本文件;对超长

源码文件进行函数级或语义块级切分,并保持文件路径、行号、函数名、宏定义和上下文编号;对同一模块的不同语言实现进行统一命名,形成后续识别可引用的代码片段库。对形式上不含算法标识的文件,智能体不作关键词过滤而一律保留,交由后续语义识别环节判定。该环节由此在降低模型推理开销的同时,保留了经二次封装、别名或自定义接口实现的密码相关代码。

2.2 调用图构建智能体

调用图构建智能体以函数、文件、组件和包为节点,以函数调用、模块引用、库依赖与配置引用为边,构建多层次调用图。对于 C/C++ 代码,结合函数定义、头文件引用、宏展开线索与构建脚本;对于 Java、Go、Python 等代码,结合包导入、对象方法调用和依赖声明;对于脚本和配置文件,则关注命令调用、证书路径与算法参数。密码框架的接口声明与硬件适配层代码不含算法的计算过程,其密码相关性须经由跨文件的调用关系确立,调用图由此为语义识别提供单个文件之外的判定依据,并同时给出该模块所处的角色与被复用的程度,作为迁移优先级排序的结构依据。

2.3 密码语义识别智能体

密码语义识别智能体利用大语言模型对候选代码片段进行细粒度语义分析,以代码的功能与用途为判据,综合函数职责、参数含义、返回值、异常处理、调用上下文、注释说明与依赖接口作出判定。该判据不以算法名称在代码中出现为前提,某些函数虽未直接命名为 `rsa_sign` 或 `sm2_verify`,但其内部涉及证书对象、公私钥结构与签名验证流程,仍应判定为签名相关模块;反之,出现 Hash 或 CRC(cyclic redundancy check)字样的文件若仅用于数据校验或压缩,则应与密码摘要模块区分。识别输出采用结构化格式,首先给出是否为密码模块的判定,判定为是的片段进一步输出第 1.3 节所定义的证据单元各字段。

2.4 可信推理与反思智能体

大模型在代码语义理解方面具有优势,但也可能受到上下文不完整、命名相似或训练偏差的影响。为降低误报与漏报,工具引入可信推理与反思智能体,对初步识别结果进行二次检查。检查分为 4 项:一是证据一致性检查,即算法标签是否与实际调用接口、参数结构和数据流相匹配;二是调用图一致性检查,

即该模块是否处在真实的调用或依赖路径中;三是反例检查,即该代码是否属于测试样例、CRC 校验、压缩编码、非密码哈希、伪随机测试或兼容性包装;四是迁移相关性检查,即该模块是否使用量子脆弱公钥密码、是否涉及长期保护数据或身份信任链。这 4 项检查分别取自代码结构、调用关系、反例形态与迁移属性,所依据的证据源互不相同,因而构成对初步结论的交叉核验,而非要求模型对同一片段重复判断^[15,23]。在此基础上,工具对结论实施置信度分级,并将长文件各切片中分散的证据汇聚为统一的文件级结论,从而在保持较高召回率的同时把误报控制在人工可复核的范围内。

2.5 结果汇编与可视化智能体

结果汇编与可视化智能体负责将分散的证据单元汇聚为面向用户的分析结果,输出密码模块列表、算法分布统计、量子脆弱模块清单、关键调用路径、供应链依赖关系与风险标签。对于复杂系统,工具以节点和边展示模块间的调用关系,并支持按算法类型、组件、目录、风险等级与置信度筛选。用户既可从系统级概览下钻至模块、函数与源码证据,也可从某一业务接口反向追踪其所依赖的密码能力,使识别结果不止于一次性扫描的输出,而成为可复核、可持续更新的治理视图。证据单元的字段构成及其向上汇聚与向下追溯关系如图 2 所示。

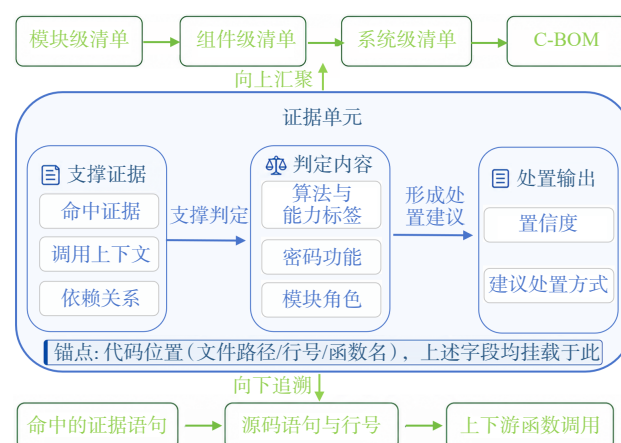


图 2 证据单元的结构与汇聚追溯关系

3 关键技术方法

本文以阿里巴巴通义千问系列开源模型 Qwen3-

8B 作为语义识别的基座模型,该模型为解码器结构的因果语言模型,参数规模约 82 亿,经指令对齐训练,可按结构化格式稳定输出判定结果,支持本地化部署与并发推理。本文直接采用官方公开权重,不作参数微调。

3.1 面向密码语义的提示词设计

密码模块识别的提示词需要同时体现密码学知识、软件工程知识和迁移场景知识,要求模型依据代码的实际用途而非表面形态,区分密码功能与普通数据处理功能。例如,对于哈希相关代码,模型需要说明其是否用于密码摘要、密钥散列消息认证码(keyed-hash message authentication code, HMAC)、签名输入、完整性度量,还是仅用于哈希表、CRC 校验或数据去重;对于随机数接口,模型需要区分密码安全随机数、普通伪随机数和测试随机数;对于证书处理代码,模型需要说明其是否参与身份认证、代码签名、TLS 握手或本地配置解析。提示词规定模型输出为结构化 JSON(JavaScript object notation),每条结论须给出算法族、密码功能与模块角色的判定,并附带函数名、调用语句、算法常量、参数名、相关头文件与上下文说明作为证据,证据不足时输出待复核标记而非强行分类^[15,23]。

3.2 基于调用图的细粒度影响分析

调用图在识别环节之外,还用于迁移影响分析。对于被判定为量子脆弱的密码模块,工具沿调用图向上追踪其调用者,确定系统服务与业务流程对模块的依赖,沿依赖关系向下追踪其底层密码库、硬件接口与第三方组件来源,建立证据单元到业务影响面的映射。在此基础上,工具以入度、出度、路径长度、组件复用度与敏感数据标签计算迁移关注度,仅存在于历史测试目录中的验签函数关注度较低,被协议适配层与多个上层接口共同调用的密钥协商模块则进入高优先级清单。识别结果由此从模块的分布扩展为模块的影响面与处置次序,为量子安全风险评估提供量化输入。

3.3 基于置信度分级的保守识别策略

迁移前的密码资产盘点中,漏报与误报的代价并不对称。遗漏关键业务链路上的公钥密码模块将导致迁移方案不完整,而清单中的误报可通过证据复核逐步剔除^[24]。工具因此采取高召回优先的判定取向,

并以置信度分级控制误报成本,而非通过提高判定阈值压制误报。证据充分且经交叉核验一致的结果为高置信度,直接进入密码资产清单;上下文不足、证据尚不充分的结果为中等置信度,由反思智能体标注不确定性后进入专家复核队列;含密码接口但结合文件路径与构建配置可判定不在生产路径中的测试与兼容代码,降级为低置信度并保留风险标记;形态与密码摘要相近的 CRC 校验、压缩编码与非密码哈希,经反思智能体的反例检查与用途标签区分后进入抑制规则。

4 实验验证与结果分析

4.1 实验对象与数据集构建

本文以国产龙蜥操作系统 23.4 社区版所采用的 ANCK 6.6.102 内核源码快照为实验对象,该快照于 2026 年 1 月 22 日取自龙蜥社区 cloud-kernel 代码仓库的 devel-6.6 分支,共含 98488 个源码文件,逻辑大小约 1.29 GiB。龙蜥操作系统已在金融、政务、电信、能源等领域实现规模化应用^[25],是银行信息系统所依赖的基础软件平台之一。

数据集以分层抽样与人工核验相结合的方式构建。先由完整方法对全部源码文件作出判定,再从预测为密码相关与预测为非密码相关的 2 个文件集合中各随机抽取 200 条,形成 400 条文件级样本。样本标签由 2 名具备操作系统与密码学背景的研究人员在盲法条件下独立核验确定,标注人员仅依据源码内容及理解代码功能所需的上下文作出判断,不接触模型的预测标签、置信度与证据信息,2 人标注的一致性系数 Cohen's Kappa 为 0.87,分歧样本经讨论后形成人工标准标签。等额分层的样本构成与源码树中 2 类文件的真实比例不同,各项指标的外推含义见第 4.8 节。

4.2 评价指标

实验采用准确率(Accuracy)、精确率(Precision, p)、召回率(Recall, r)与 F_1 值评估识别效果。准确率衡量总体预测与人工标签一致的比例,精确率衡量判定为密码模块的样本中实际为密码模块的比例,召回率衡量真实密码模块被发现的比例, F_1 值综合反映精确率与召回率之间的平衡(表 1)。在此基础上,本文

表 1 文件级密码模块识别实验结果

指标	计算方式	结果/%
Accuracy	$(T_p+T_N)/(T_p+F_p+F_N+T_N)$	96.75
Precision	$T_p/(T_p+F_p)$	97.50
Recall	$T_p/(T_p+F_N)$	96.06
F_1	$2T_p/(2T_p+F_p+F_N)$	96.77

将误报率定义为 $1-p$, 即判定为密码模块的样本中实际不属于密码模块的比例, 将漏报率定义为 $1-r$, 即真实密码模块中未被发现的比例, 下文涉及误报率与漏报率的比较均按此口径。

4.3 实验结果

基于数据集中文件级样本的混淆矩阵统计结果为真正例(T_p)195、假正例(F_p)5、假负例(F_N)8、真负例(T_N)192, 按 Wilson 方法计算, 准确率、精确率与召回率的 95% 置信区间分别为[94.5%, 98.1%]、[94.3%, 98.9%]与[92.4%, 98.0%], 各区间下界均不低于 92.4%, 各项指标的点估计如表 1 所示。97.5% 的精确率对完整方法在全量源码树上的识别结果同样成立, 即工具输出的密码模块清单中, 超过 97% 的条目经人工核验确为密码相关文件。从结果看, 工具在复杂源码环境中实现了较高的整体准确性和正类识别能力。

进一步考察误差分布, 发现误判集中于文件级判定的能力边界上。8 例漏报出现在密码框架、接口声明与硬件适配层代码中, 这类文件不含算法的计算过程, 其密码相关性须经由跨文件调用关系或运行时绑定确立。例如 `adf_gen2_config.h` 仅声明 `adf_gen2_dev_config()` 设备配置接口, 其实现文件中调用密码配置函数并设置对称与非对称密码请求环, 多个 Intel QAT 驱动又通过函数指针将该接口绑定为设备配置回调。5 例误报出现在 CRC 校验、普通哈希与压缩编码类代码中, 这类文件将任意长度输入映射为定长输出, 与密码摘要形态相同, 模型据此判定其含有密码功能, 但其校验与压缩逻辑不以认证或机密性为目标, 不构成密码学意义上的安全机制。

误判分析结果表明, 密码模块的判定所需的证据超出了单个文件的边界。这可从 2 个方向作针对性改进。针对漏报, 可引入具备仓库级上下文建模能力的代码智能体, 将判定单元由单个文件扩展至其调用闭包, 使密码语义分散于多文件的模块在完整调用路

径上得到确认。针对误报, 可通过领域语料增强或外部知识库检索引入密码学领域知识, 使模型在代码形态之外获得安全属性层面的判据, 区分具备认证与机密性用途的密码机制与形态相近的普通校验逻辑。

4.4 对比实验

面向源码的密码资产发现工作中, 仅基于模式匹配的工具不依赖特定密码库与语言解析器, 可直接运行于内核源码, 据此选取开源工具 CryptoScan^[13]作为对照。为使 2 种方法可比, 对照实验在同一数据集上进行, 以人工核验标签为唯一评判依据。CryptoScan (v1.3.0) 按默认配置运行, 加载全部 90 条内置检测模式, 保留默认的上下文分析、噪声抑制与结果去重流程, 并启用量子安全算法识别选项, 与本文的密码模块定义保持一致。该工具以代码行 finding 为输出单位, 扫描结果中包含至少一条 finding 的文件判定为密码相关文件, 与本文方法的文件级输出对齐。该数据集按本文方法的预测标签分层抽取, 表 2 中准确率与召回率反映 2 种方法在同一批样本上的相对表现。

表 2 与开源密码发现工具的识别效果对比

方法	Accuracy/%	Precision/%	Recall/%	F_1 /%
CryptoScan	75.00	88.15	58.62	70.41
本文方法	96.75	97.50	96.06	96.77

本文方法较 CryptoScan 在同一批样本上的精确率相对提升 10.6%, 召回率相对提升 63.8%, 误报率由 11.85% 降至 2.50%, 漏报率由 41.38% 降至 3.94%, 相对降幅分别为 78.9% 与 90.5%。本文方法在 4 项指标上均具优势, 其中召回率的提升幅度最大。样本中有 84 个真实密码模块未被 CryptoScan 发现, 本文方法在其中识别出 76 个, 表明模块的密码功能可从代码用途中确立, 词法层面的不可见并不必然意味着判定依据的缺失。这些文件的密码能力经内核密码框架注册与调度, 代码中不出现算法名称与密钥材料, 词法层面缺少可供命中的特征, 而本文方法依据函数职责、参数结构与调用上下文作出判定, 不受算法标识是否显式出现的限制。误报率的下降则进一步表明, 语义判据在扩大识别范围的同时保持了清单条目的可靠性。

4.5 消融实验

为分析各机制的贡献, 本文在同一数据集上设置

4 组实验, 各组的模型权重、推理参数与预处理结果保持一致, 结果见表 3。其中, 数据接入与预处理智能体与密码语义识别智能体构成识别流程的必要环节, 移除后流程无输出, 结果汇编与可视化智能体在文件级标签生成后消费识别结果, 不改变文件级判定。消

融变量为调用图构建与可信推理反思 2 个智能体是否参与, 两者均不启用的配置下文统称为基线配置。在不启用调用图构建智能体的配置中, 可信推理反思智能体相应停用其调用图一致性检查, 仅执行证据一致性检查、反例检查与迁移相关性检查。

表 3 消融实验配置与结果

实验	预处理	语义识别	调用图	反思	Accuracy/%	Precision/%	Recall/%	F_1 /%
密码语义识别智能体	✓	✓	—	—	59.25	95.45	20.69	34.01
+调用图构建智能体	✓	✓	✓	—	86.75	97.47	75.86	85.32
+可信推理反思智能体	✓	✓	—	✓	89.75	97.65	81.77	89.01
完整方法	✓	✓	✓	✓	96.75	97.50	96.06	96.77

实验表明, 4 组配置的 Precision 稳定在 95.45%~97.65%, Recall 由 20.69% 提升至 96.06%, 2 项机制的贡献集中在识别的完整性上。调用图构建智能体补足文件之外的模块关联证据, 可信推理反思智能体则将文件之内分散于各切片的判定汇聚为统一的文件级结论。2 条路径单独启用均可将 Recall 提升至 75% 以上, 同时启用时 Recall 较任一单项配置分别再提高 20.20 与 14.29 个百分点。完整方法的 Precision 为 97.50%, 较仅启用可信推理反思智能体的配置低 0.15 个百分点, 对应 F_p 由 4 个增至 5 个, 即 2 项机制叠加在扩大覆盖范围的同时引入 1 个额外误报, 相对于 Recall 14.29 个百分点的提升, 该代价可以接受。相较基线配置, 完整方法的 F_N 由 161 个降至 8 个, 错误总数由 163 个降至 13 个, 减少 92.02%, 4 组配置之间的差异均通过 McNemar 检验的保守下界, 在 0.001 水平上显著, 检验方法与结果见第 4.8 节。

4.6 开销分析

实验采用 Qwen3-8B 官方 BF16 权重, 推理框架为 vLLM v0.14.0, 部署于 2 张 NVIDIA RTX 3090 显卡, 张量并行规模为 2, 客户端以 16 线程并发提交请求, 服务端最大序列长度设置为 4096 token。模型生成参数设置: 采样温度 temperature=0、核采样阈值 top_p=0.8、最大生成长度 max_tokens=520、重复惩罚系数 repetition_penalty=1.05, 关闭 thinking 模式, 直接输出结构化判定结果。vLLM 显存利用比例设为 0.8, 按每卡 24 GiB 物理显存计算, 推理服务的显存额度约为 19.2 GiB/卡。源码预处理以完整代码行为边界切分, 优先保持函数或语义上下文的完整性, 当单

个函数或语义块超过输入长度限制时, 切片长度上限为 7000 字符, 相邻切片重叠 500 字符, 围绕密码语义信号形成的单个聚焦代码片段最多保留 5000 字符。在该配置下, 工具对龙蜥操作系统内核源码的 98488 个文件完成端到端分析, 总耗时约 2.2 h, 端到端吞吐量约 12.43 文件/s, 批量请求成功率为 100%, 按设备数量与运行时间计算的 GPU 资源占用量约为 4.4 GPU·h。所用 RTX 3090 为消费级显卡, 实验不依赖专用推理加速硬件或算力集群, 10 万文件量级的密码资产清点可在半个工作日内完成, 其时间开销允许该分析作为常规环节纳入迁移前治理流程。

4.7 工具实现与可视化系统

该工具已在龙蜥、鸿蒙和红旗操作系统场景中完成部署运行, 本文的量化评测以龙蜥操作系统内核源码为对象。图 3 为工具在该场景下的可视化系统界面, 其中呈现了识别得到的密码模块在目录结构中的分布、各算法族的数量统计, 以及量子脆弱模块的调用路径。用户可据此定位具体模块并查看其证据单元。

4.8 有效性威胁

按预测标签分层的审计设计使各项结果具有不同的效力。预测正例的精确率 97.50% 与预测负例的漏判比例 4.00% 分别为 2 个集合的无偏估计, 不依赖对全库正负例比例的假设, 前者直接反映密码资产清单条目的可靠性, 后者反映清单之外的遗漏程度, 并对经预处理过滤与经模型判定为负的 2 类文件一并计量。对比实验与消融实验的比较采用同样的基准, 各方法与各配置面对相同的判定对象, 因而其相对差异同样不受样本构成的影响。全库口径的准确率与

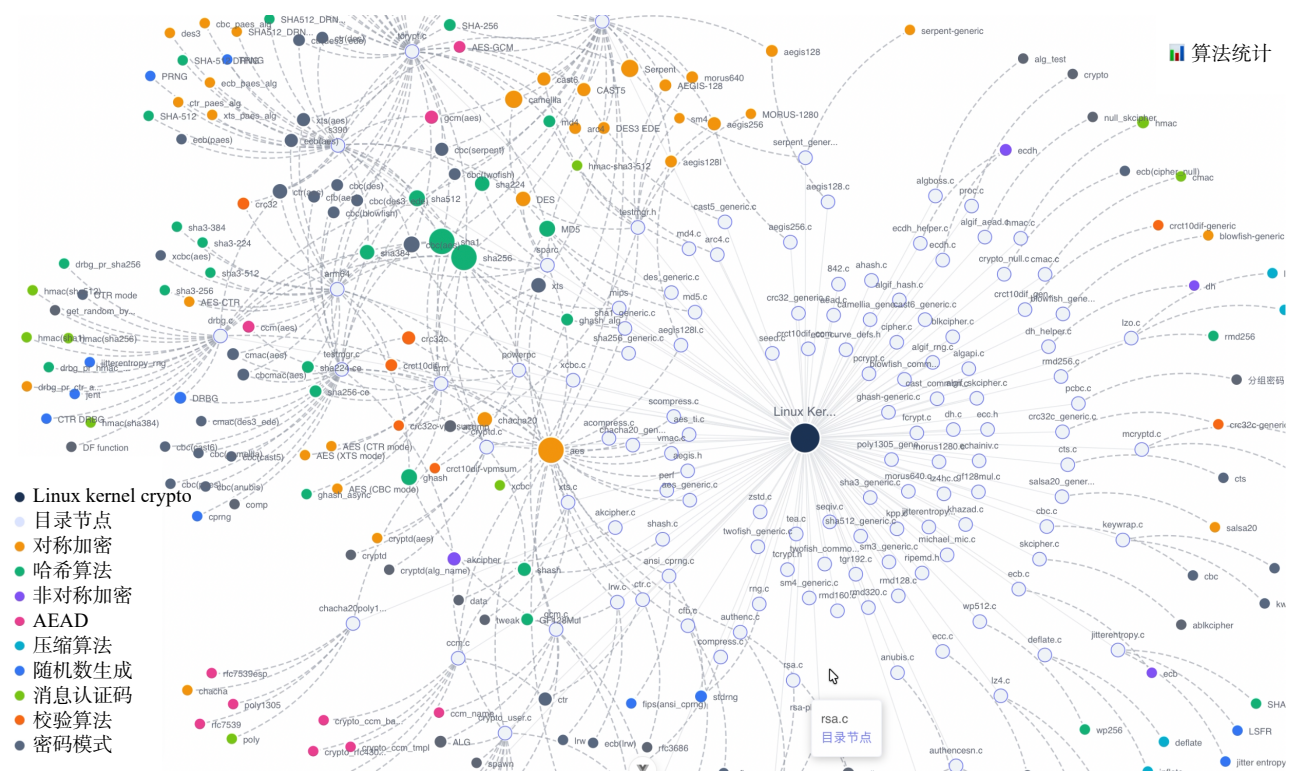


图 3 密码模块识别可视化系统

召回率需结合 2 个预测集合在源码树中的实际规模比例加权得到,各表中的对应取值则由本文的分层样本直接计算。

上述比较的显著性由 McNemar 检验确认。4 组配置与对照工具在同一批样本上给出配对判定, 设 b 与 c 分别为 2 方法中仅一方判定正确的样本数, 则 $b-c$ 恒等于两者正确判定数之差, $b+c$ 不超过两者错误判定数之和, 据此可得 McNemar 统计量的下界。按该下界, 完整方法相对 CryptoScan、基线配置、+调用图构建与+可信推理反思的 χ^2 分别不低于 65.5、126.1、23.1 与 13.5 (自由度为 1, 对应 p 分别小于 6×10^{-16} 、 3×10^{-29} 、 2×10^{-6} 与 3×10^{-4}), 即无论配对判定的具体分布如何, 上述差异均在 0.001 水平上显著, 语义判据与 2 项协同机制的作用得到统计层面的支撑。

本文的任务定位是在中国 PQC 算法标准发布之前,以国际主流密码算法为对象,先行验证识别与评估的技术路径,量化评测据此以操作系统内核源码为对象,内核为银行信息系统提供系统级密码能力,其密码逻辑经框架注册与调度组织。银行核心业务系统源码属关键信息基础设施范畴,涉及商业秘密与系统安全,第三方难以合法获取用于公开发表的评测,

面向该类系统的验证将在条件具备后于金融机构受控环境中开展。

5 在金融银行业抗量子迁移中的应用路径

5.1 从识别结果到迁移决策依据

在迁移框架中,密码模块识别之后紧接着是密码资产清单汇编。识别工具输出的证据单元以算法标签、代码位置、调用上下文与依赖关系构成 C-BOM 的对应字段,金融机构无需另行组织数据采集,即可掌握密码能力在系统中的分布与来源。该清单将业务系统、软件组件、密钥证书与供应链关系连为一体,长期敏感数据的保护链路、关键交易的签名链路

与高复用的密码依赖由此可见,量子安全风险评估得以在事实上展开。评估阶段,工具将 RSA、ECC、SM2 等第 1 代公钥密码模块标注为量子脆弱候选,并依据其所处的业务链路、经依赖关系传播的范围与替换算法所需的接口改动,判断各模块的风险等级与处置成本,据此形成迁移次序。

5.2 从迁移实施到效果验证

POC 迁移并非一次性替换算法,而是涉及协议话

配、接口兼容、证书体系、密钥生命周期与灰度切换的系统工程。对 PQXDH 协议的形式化验证表明,即便采用抗量子 KEM,协议层的绑定关系与并行部署方式仍会影响安全结论^[8]。识别结果因此须保留算法、协议、接口与调用方的完整证据链。依托这一证据链,工具在迁移全程持续发挥作用:迁移前生成候选模块清单并给出改造范围,迁移中跟踪代码变更,迁移后重新扫描,验证旧算法调用是否消除、混合密码路径是否生效、业务调用是否切换至新接口;在代码仓库更新、依赖版本变更与组件替换时同样可重新运行,对比前后 C-BOM 的差异。对分阶段迁移的金融系统,工具可对外协议接口采用传统算法与 PQC 算法并行的混合方案,对历史兼容模块保留风险标签并设置退役计划。

5.3 从工具能力到行业实践

本文方法在实验环境中已得到验证,其在行业范围内的推广还依赖若干配套条件。首先是评测基准。面向金融银行业的行业化基准仍有待建立,样本需覆盖网银、跨行清算、证书认证与软件包签名等典型场景,并包含二次封装、供应链间接依赖与形态相近的干扰项,评价维度除准确率与召回率外,还应考察告警能否为开发人员的修复与迁移决策提供依据^[26]。其次是输出规范。C-BOM 的字段、算法命名、证据格式与导出接口尚未统一,识别结果须以机器可读格式输出,方能进入资产管理平台、代码仓库与密钥管理系统。在此基础上,专家复核的结果应沉淀为提示词样例、误报规则与业务标签,使工具在同类系统中的判定持续改进;识别能力亦应前移至代码审查、依赖引入与构建发布环节,使密码资产的变化在引入时即被掌握,避免迁移完成后再次引入量子脆弱算法。

6 挑战与发展建议

1) 建立面向行业场景的识别基准。密码模块自动化识别工具的效果高度依赖测试基准。建议围绕金融银行业典型场景建设行业化基准数据集,覆盖网银、手机银行、跨行清算、证书认证、UKey、数据库加密、VPN/IPSec、软件包签名、运维登录和日志完整性等场景。基准应同时包含显式密码调用、二次封装、供应链依赖、测试样例、误报干扰项和 PQC 改造

样例,以便客观评估工具的准确性、覆盖率和迁移相关性。在基准设计上,还应吸收近 2 年国际研究对工具评价的经验:一方面评价准确率、精确率、召回率和 F_1 值;另一方面评价告警是否提供可操作的证据、是否能够解释上下文、是否能够被开发人员用于修复或迁移决策^[26]。对于金融行业而言,基准样本还应覆盖供应链间接依赖和跨系统调用场景,因为这些路径往往决定迁移范围和业务影响面。

2) 推动 C-BOM 规范与工具输出格式统一。目前 C-BOM 仍处于探索和完善阶段。为使识别结果真正服务于迁移治理,应尽快推动密码资产清单字段、算法命名、模块角色、证据格式、风险标签和导出接口的统一。工具输出不应只停留在人可读报告,还应支持机器可读格式,便于与资产管理平台、漏洞管理平台、代码仓库、CI/CD(continuous integration/continuous delivery)流水线、密钥管理系统和风险评估引擎对接。

3) 加强人工复核与专家规则协同。大模型识别并不能完全替代密码专家和系统研发人员。对于关键金融业务链路,应建立自动识别、专家复核、规则沉淀、再次扫描的综合机制。专家复核不仅用于纠正单次结果,也应转化并沉淀为提示词样例、误报规则、白名单、黑名单和业务标签,使工具在同类系统中的表现持续提升。

4) 将识别能力嵌入软件生命周期。抗量子迁移是长期工程,不能等到统一替换阶段才开展密码清点。建议将密码模块识别能力前移至需求设计、代码审查、依赖引入、构建发布和上线验收环节。在新系统建设中,应要求关键密码接口具备算法可配置、协议可协商、证书可轮换和依赖可追溯能力;在存量系统改造中,应通过持续扫描掌握密码资产变化,避免迁移后再次引入量子脆弱算法。

7 结论

密码模块识别是金融银行业 PQC 迁移的第 1 环节,也是整个迁移工程能否科学、准确且可验证地推进的基础。面对复杂源码、多语言实现、供应链依赖和业务封装带来的识别难题,基于大模型的密码模块识别工具提供了一种新的工程路径。该工具通过多

智能体协作将全量代码分析、调用图构建、密码语义识别、证据反思和可视化汇编结合,输出代码级的证据单元,支撑 C-BOM 建设与量子安全风险评估。实验结果显示,该工具在文件级样本上的精确率为 97.50%,召回率为 96.06%。与基于模式匹配的开源工具的对照表明,语义判据可以覆盖词法层面不可见的密码模块;消融实验显示,调用图构建与可信推理反思 2 项机制的贡献集中在识别的完整性上,二者协同将召回率由 20.69% 提升至 96.06%。全量分析在 2 张消费级 GPU 上完成,10 万文件量级的密码资产清点可在半个工作日内结束。后续工作将完善金融行业基准数据集、补充随机全库抽样下的总体指标估计,并推动识别能力嵌入软件研发与运维流程。随着 PQC 标准化和金融行业迁移实践不断推进,基于大模型的密码模块识别有望成为 PQC 迁移从“人工排查”走向“数据驱动治理”的关键支撑工具。

数据与代码可获得性声明: 实验所用操作系统内核源码和 Qwen3-8B 模型的公开发布权重均可从相应公开仓库获取;本文实现代码、运行配置及人工标注数据可在合理请求下向通信作者获取。

参考文献 (References)

- [1] Shor P W. Algorithms for quantum computation: Discrete logarithms and factoring[C]//Proceedings of the 35th Annual Symposium on Foundations of Computer Science. Los Alamitos, CA: IEEE Computer Society Press, 1994: 124–134.
- [2] 江苏省金融学会. 银行业抗量子计算密码迁移管理指南: T/JPSFFB 001—2026[S]. 南京: 江苏省金融学会, 2026.
- [3] 中关村网络安全与信息化产业联盟. 国产操作系统商用密码应用专委会. 操作系统密码应用行业发展报告[R]. 北京: 中关村网络安全与信息化产业联盟, 2026.
- [4] Newhouse W, Souppaya M, Barker W, et al. Migration to post-quantum cryptography: Quantum readiness: Cryptographic discovery: NIST SP 1800–38B (preliminary draft) [R/OL]. 2023–12–01. <https://www.nccoe.nist.gov/sites/default/files/2023-12/pqc-migration-nist-sp-1800-38b-preliminary-draft.pdf>.
- [5] National Institute of Standards and Technology. FIPS 203: module–lattice–based key–encapsulation mechanism standard[S]. Gaithersburg, MD: National Institute of Standards and Technology, 2024.
- [6] National Institute of Standards and Technology. FIPS 204: Module–lattice–based digital signature standard[S]. Gaithersburg, MD: National Institute of Standards and Technology, 2024.
- [7] National Institute of Standards and Technology. FIPS 205: Stateless Hash–based digital signature standard[S]. Gaithersburg, MD: National Institute of Standards and Technology, 2024.
- [8] Bhargavan K, Jacomme C, Kiefer F, et al. Formal verification of the PQXDH post-quantum key agreement protocol for end-to-end secure messaging[C]//Proceedings of the 33rd USENIX Security Symposium. Philadelphia, PA: USENIX Association, 2024: 469–486.
- [9] 周诚君, 李一, 王启宇, 等. 时不我待: 量子计算威胁、银行业脆弱性与金融系统性风险[M]. 北京: 中国金融出版社, 2026.
- [10] Ahmed N, Zhang L, Gangopadhyay A. A survey of post-quantum cryptography support in cryptographic libraries [C]//Proceedings of the 2025 IEEE International Conference on Quantum Computing and Engineering. Piscataway, NJ: IEEE, 2025: 906–917.
- [11] Chen Y, Liu Y, Wu K L, et al. Towards precise reporting of cryptographic misuses[C]//Proceedings of the 2024 Network and Distributed System Security Symposium. San Diego, CA: Internet Society, 2024.
- [12] Post-Quantum Cryptography Alliance. CBOMkit: A toolset for dealing with cryptography bills of materials: Version 2.2. 0[EB/OL]. (2026–02–05) [2026–08–06]. <https://github.com/cbomkit/cbomkit/releases/tag/2.2.0>.
- [13] Cybersecurity Nonprofit. CryptoScan: Cryptographic discovery scanner: version 1.3. 0[EB/OL]. (2026–01–30) [2026–08–06]. <https://github.com/csnpp/cryptoscan/releases/tag/v1.3.0>.
- [14] Barker W, Souppaya M. Migration to post-quantum cryptography: Project description[R/OL]. Gaithersburg, MD: National Cybersecurity Center of Excellence, 2021[2026–08–06]. <https://www.nccoe.nist.gov/publications/project-description/migration-post-quantum-cryptography-project-description-final>.
- [15] Li Z, Dutta S, Naik M. IRIS: LLM-assisted static analysis for detecting security vulnerabilities[C/OL]//Proceedings of the 13th International Conference on Learning Representations. 2025[2026–08–06]. <https://openreview.net/forum?id=9LdJDU7E91>.
- [16] Ding Y, Fu Y, Ibrahim O, et al. Vulnerability detection with code language models: How far are we?[C]//Proceedings of the 2025 IEEE/ACM 47th International Conference on Software Engineering. Los Alamitos, CA: IEEE Computer Society, 2025: 1729–1741.
- [17] Sheng Z, Chen Z, Gu S, et al. LLMs in software security: A survey of vulnerability detection techniques and insights[J]. ACM Computing Surveys, 2025, 58(5): 134: 1–134: 35.
- [18] Mousavi Z, Islam C, Babar M A, et al. Detecting misuse of

- security APIs: A systematic review[J]. ACM Computing Surveys, 2025, 57(12): 303: 1–303: 39.
- [19] Rahaman S, Xiao Y, Afrose S, et al. CryptoGuard: High precision detection of cryptographic vulnerabilities in massive-sized Java projects[C]//Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. New York: ACM, 2019: 2455–2472.
- [20] Krüger S, Nadi S, Reif M, et al. CogniCrypt: Supporting developers in using cryptography[C]//Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering. Piscataway, NJ: IEEE, 2017: 931–936.
- [21] Afrose S, Rahaman S, Yao D. CryptoAPI-Bench: A comprehensive benchmark on Java cryptographic API misuses[C]//Proceedings of the 2019 IEEE Cybersecurity Development Conference. Piscataway, NJ: IEEE, 2019: 49–61.
- [22] Ami A S, Cooper N, Kafle K, et al. Why crypto-detectors fail: A systematic evaluation of cryptographic misuse detection techniques[C]//Proceedings of the 2022 IEEE Symposium on Security and Privacy. Piscataway, NJ: IEEE, 2022: 397–414.
- [23] Liu J, Yang Y, Chen K, et al. Generating API parameter security rules with LLM for API misuse detection[C]//Proceedings of the 2025 Network and Distributed System Security Symposium. San Diego, CA: Internet Society, 2025.
- [24] Ami A S, Moran K, Poshyvanyk D, et al. "False negative – that one is going to kill you": Understanding industry perspectives of static analysis based security testing[C]//Proceedings of the 2024 IEEE Symposium on Security and Privacy. Piscataway, NJ: IEEE, 2024: 3979–3997.
- [25] 中国智能计算产业联盟. 国产服务器操作系统发展报告(2025)[R]. 北京: 中国智能计算产业联盟, 2025.
- [26] Steenhoek B, Sivaraman K, Saldivar Gonzalez R, et al. Closing the gap: A user study on the real-world usefulness of AI-powered vulnerability detection and repair in the IDE[C]//Proceedings of the 2025 IEEE/ACM 47th International Conference on Software Engineering. Los Alamitos, CA: IEEE Computer Society, 2025: 1–13.

Research on a large language model-based cryptographic module identification tool

WEI Pengkun, LIU Chao*, HU Haibo

Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, Chongqing 400044, China

Abstract Cryptographic module identification is the starting point of post-quantum cryptographic migration in the banking and financial industry, and its results form the basis for compiling a cryptographic asset inventory and prioritizing migration. In view of the large scale, language diversity, deeply nested wrapper layers and complex dependency chains of banking information systems and operating system source code, an LLM-based cryptographic module identification tool is investigated in this paper. Taking the whole codebase and its dependency information as input, the tool completes identification through the collaboration of five agents responsible for data ingestion and preprocessing, call graph construction, cryptographic semantic identification, trustworthy reasoning and reflection, and result aggregation and visualization, and outputs evidence units traceable to source statements. The tool is implemented on the open-source Qwen3-8B model without parameter fine-tuning. On 400 file-level samples constructed from the source code of the Anolis OS kernel, it achieves a precision of 97.50% and a recall of 96.06%. In a comparative experiment against an open-source pattern-matching cryptographic discovery tool, the proposed tool identifies 76 of the 84 genuine cryptographic modules missed by the reference tool; ablation experiments show that the combination of call graph construction and trustworthy reasoning and reflection raises recall from 20.69% to 96.06%. These results indicate that semantic-level criteria extend the scope of cryptographic module identification from implementations explicitly visible in the code text to system-level code organized through registration and encapsulation.

Keywords cryptographic module identification; large language model; multi-agent; post-quantum cryptographic migration; cryptographic asset inventory ●



(责任编辑 王志敏)