Proceedings of CONF-MLA 2025 Symposium: Intelligent Systems and Automation: AI Models, IoT, and Robotic Algorithms
DOI: 10.54254/2755-2721/2025.LD29484

Theoretical Challenges and Future Development of Cryptographic Key Management Mechanisms in Software Engineering Development

Fengshuo Liu

School of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang, China
483137467@qq.com

Abstract. In the context of IoT software engineering, existing research improves IoT security using software-based adjustments, like dynamic key updates in Libcoap and blockchain-hash key storage. Yet, it has clear flaws: lightweight post-quantum cryptography (PQC) is incompatible with low-power devices (e.g., 8-bit MCUs) due to high overhead and no standardized interfaces; meanwhile, multi-layer IoT architectures lack coordination between key lifecycle management and software development. This study focuses on two questions: challenges of cryptographic key management mechanisms and their future directions, with threefold significance in filling IoT key management gaps, supporting post-quantum theories, and expanding software engineering-cryptography boundaries. It analyzes challenges via IoT software traits: device resource adaptation conflicts (low-power hardware vs. PQC demands), cross-layer process disconnection ("software-first, key-later" model issues), and multi-protocol compatibility confusion (diverse protocol logics). For future trends, it proposes lightweight PQC (optimized algorithms, hardware-software acceleration) and standardized interfaces. The study concludes that IoT key management faces three core challenges tied to IoT software traits, and future solutions lie in lightweight PQC and integrated interfaces to enhance security and efficiency.

Keywords: IoT Cryptographic Key Management, Post-Quantum Cryptography (PQC), Low-Power IoT Devices, Cross-Layer Process Coordination, Multi-Protocol Compatibility

1. Introduction

From a software engineering view, current research optimizes IoT security via software-level adaptation: integrating dynamic key updates into Libcoap, designing modular AES/ECC-based authentication for resource-constrained devices, or exploring blockchain-hash key storage to reduce central dependence [1]. To address quantum threats, modifying protocols like MQTT for quantum key metadata transmission faces issues such as protocol field length limits.

Key research gaps exist at the software level: Lightweight post-quantum cryptography (PQC) algorithms disconnect from low-power device (e.g., 8-bit MCU) software stacks—their computational/memory overhead exceeds device capacities, and lacking standardized interfaces with

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.LD29484

traditional encryption modules hinders application [2]. Also, multi-layer architecture lacks coordination between key lifecycle management and software development, causing inefficient cross-layer key synchronization.

For IoT key management, prioritize "software lightweightness": develop lightweight PQC algorithms (e.g., reducing rounds, optimizing memory) and standardized interfaces for seamless integration with traditional encryption, solving 8-bit MCU compatibility.

Optimizing software-level adaptation between quantum keys and IoT protocols (e.g., extending MQTT field length, designing quantum key metadata compression [3]) is critical for post-quantum key security migration.

This paper addresses two questions: 1) What challenges face cryptographic key management? 2) What is its future direction?

This study has threefold significance: filling IoT software key management theoretical gaps, offering post-quantum key management theoretical basis, and expanding software engineering-cryptography interdisciplinary boundaries.

2. Analysis of challenges faced by cryptographic key management mechanisms

Within the context of IoT software engineering, cryptographic key management mechanisms confront multi-dimensional challenges that impede both their security and efficiency, challenges that are intrinsically linked to the technical traits of IoT software. These challenges align with three core theoretical themes, as detailed below.

2.1. Device software resource adaptation challenge: contradiction between low-power hardware and algorithm requirements

A large number of low-power IoT devices (e.g., 8-bit MCUs, smart sensors) possess limited software stack carrying capacity [4], which presents a significant conflict with the demands of cryptographic key management algorithms—this is the core challenge under the "device software resource adaptation" theme.

From the hardware side, 8-bit MCUs typically have only a few KB to tens of KB of memory and slow operation speeds. However, traditional Post-Quantum Cryptography (PQC) algorithms (e.g., lattice-based, hash-based cryptography) require complex computations and substantial memory to ensure security. For instance, a typical lattice-based PQC algorithm requires over 10 KB of memory just for initial parameter loading, and a single key generation operation consumes tens of thousands of clock cycles—far exceeding 8-bit MCUs' software carrying limits, making direct PQC deployment impossible.

From the software adaptation side, existing PQC algorithms lack unified integration interfaces with traditional encryption modules (e.g., AES, ECC). In development, engineers integrating both traditional and PQC encryption into low-power devices must write custom interface codes to coordinate the two. This increases the development workload and often leads to key synchronization delays or conflicts in encryption logic stemming from poor adaptation. Industry surveys indicate that the rework rate from key algorithm-software stack adaptation in low-power IoT software development is as high as 35%, seriously disrupting projects and endangering device security.

The root cause of this challenge lies in the mismatch between the performance requirements of the algorithms and the carrying capacity of the underlying hardware and software. Unresolved, it leaves countless low-power IoT devices vulnerable.

DOI: 10.54254/2755-2721/2025.LD29484

2.2. Cross-layer software process coordination challenge: disconnection between key lifecycle and development process

IoT software employs a multi-level architecture [5] (perception, network, protocol adaptation, business layers), but the key management lifecycle (generation, distribution, update, destruction) remains severely disconnected from the software development process—this is the core challenge under "cross-layer process coordination".

The conventional 'software-first, key-later' development model, wherein business modules (e.g., data collection, command transmission) are developed first and key management is retrofitted, introduces significant problems.

In process coordination, not presetting key synchronization fields during protocol adaptation layer development forces large-scale modifications to existing protocol parsing modules when adding key distribution later. This compromises software architecture stability and introduces potential vulnerabilities. For example, an IoT platform ignored key synchronization in early development; later modifying MQTT message formats for perception-network layer key updates caused compatibility issues with 100,000+ devices, costing over 5 million yuan to fix.

In permission management, the lack of associated key permission control during business layer development results in mismatches between keys and business data access rights, leading to security risks such as "low-permission services using high-permission keys". Security tests reveal a 42% key permission error rate in "key later" IoT software—far higher than the 8% of "key-synced" development.

This challenge proves deep coupling between the key lifecycle and development process is essential; without it, IoT software security and maintainability suffer drastically.

2.3. Multi-protocol software ecosystem compatibility challenge: management confusion from protocol differences

The IoT encompasses diverse protocols (LoRa, Wi-Fi, MQTT, CoAP) with vastly different key management logics and data formats, rendering multi-protocol compatibility for key management difficult—this is the main challenge under "multi-protocol ecosystem compatibility".

In terms of key parsing, different protocols impose unique requirements for key encapsulation and transmission: LoRa embeds keys in physical layer frames; Wi-Fi classifies keys (PSK, GTK) via 802.11i for MAC layer storage; MQTT transmits keys as headers or payloads. Without a unified parsing interface, separate key processing logics are needed for each protocol, increasing costs and causing management confusion.

In ecosystem coordination, multi-protocol device interconnection complicates key negotiation [6]. A smart park project featuring LoRa sensors, Wi-Fi cameras, and MQTT gateways required three independent key management systems, raising hardware/operation costs and causing synchronization delays. Transitions between protocols resulted in encryption interruptions, elevating the risk of security incidents by 28%.

With evolving protocols (e.g., Matter), new protocols force repeated key module development, slowing iteration. Industry data shows that key management development costs account for 30-40% of total costs in multi-protocol projects, a figure that rises with the increasing number of protocols. This highlights the urgent need for a unified key management standard framework.

DOI: 10.54254/2755-2721/2025.LD29484

3. Future development trends of cryptographic key management mechanisms

3.1. Lightweight and integrated trend: solving device software resource adaptation

To address the conflicts between low-power device software stacks and cryptographic algorithms, future mechanisms will concentrate on lightweight Post-Quantum Cryptography (PQC) algorithms and standardized integration interfaces, which are central to the concept of "device software resource adaptation".

In the search and development of lightweight PQC R&D, researchers optimize algorithms (reduce rounds, shrink parameters) and adapt them for specific hardware implementations. For lattice-based algorithms, reducing dimensions and simplifying matrix operations cuts memory usage to under 5 KB and clock cycles by 60%, fitting 8-bit MCUs. Researchers are also developing hardware-friendly PQC algorithms specifically designed for low-power devices characterized by limited registers and simple instruction sets. This approach is often paired with a “hardware acceleration + software optimization” strategy: for instance, smart sensors utilize hardware for PQC core operations (hash calculation, matrix multiplication) and software for parameter config/result checks, boosting speed 3-5x.

In the realm of standardized interfaces, the industry will set unified specs for seamless PQC-traditional encryption integration. These interfaces encompass three primary modules: parameter transmission (unifying key length, hash bitcount), key synchronization (aligning PQC and AES key generation/update), and algorithm switching (dynamic selection based on device resources—PQC for sufficient resources, traditional for constraints).

Such an interface obviates the need for custom adaptation code, thereby reducing rework rates to below 10% while simultaneously enhancing both efficiency and security.

4. Conclusion

Cryptographic key management mechanisms currently confront multi-dimensional challenges in the context of IoT software engineering, which are tightly tied to IoT software’s technical traits and hinder key management security and efficiency. These challenges align with three core theoretical themes, as summarized below.

For Device Software Resource Adaptation, low-power devices (e.g., 8-bit MCUs and smart sensors) possess limited software stack capacity. Traditional Post-Quantum Cryptography (PQC) algorithms require complex operations and large memory—for instance, a typical lattice-based PQC algorithm needs over 10 KB of memory for initial parameter loading, far exceeding 8-bit MCUs’ limits. Additionally, PQC algorithms lack standardized integration interfaces with traditional encryption modules (e.g., AES, ECC), compelling engineers to write custom interface code, which results in a 35% rework rate in low-power IoT software development and leaving devices vulnerable.

Regarding Cross-Layer Process Coordination, IoT software’s multi-level architecture (perception, network, protocol adaptation, business layers) is characterized by a disconnect between the key lifecycle (generation, distribution, update, destruction) and the software development process. The "software first, key later" model leads to issues: unpreset key synchronization fields in the protocol adaptation layer require later large-scale modifications (e.g., an IoT platform’s MQTT protocol adjustment costing over 5 million yuan to fix compatibility with 100,000+ devices), and unassociated key permission control in the business layer results in a 42% key permission error rate

DOI: 10.54254/2755-2721/2025.LD29484

—far higher than the 8% of synchronized development—thereby undermining software stability and security.

References

[1] Shi, X. B. (2024). A Secure Encryption Method for Computer Network Communication Data Based on AES and ECC Hybrid Algorithm. Journal of Changzhou Institute of Technology, 37(03), 6-10. https: //doi.org/CNKI: SUN: CZGB.0.2024-03-002.

[2] Lai, J. S., Zhao, W. Y., & Zhang, H. Y. (2024). Analysis of Information Security Threats and Countermeasures in Quantum Computing. Information and Communications Technology and Policy, 50(07), 24-29. https: //doi.org/CNKI: SUN: DXWJ.0.2024-07-005.

[3] Zhang, H. (2019). Research and Application of MQTT Protocol Extension and Protocol Optimization (Master's Thesis, Beijing University of Posts and Telecommunications). https: //kns.cnki.net/kcms2/article/abstract?v=45FX-MmhZ-E8pOHWHnQkr5kUatz1kroDaSMtWgnVBWIN2-h1aJAzotrR_VzRoXGsoXOombegBi8ZyRH3rvunS_TLeFFjr2P9_hzUhLwQ1L4BXyObXYP8_RSC-BaBZuGTmv3r7-iFOFFDsu6l1JORHJj7pE6GkYzceUTzLrOs9-1tIIFP5afFGw==& uniplatform=NZKPT& language=CHS

[4] Lugantsev, L. D., & Chernenko, M. O. (2013). Software of Computer Monitoring Carrying Capacity and Resource of Tubular Elements of Constructions. Izvestiya MGTU MAMI, 7(1-4), 18-25. https: //doi.org/10.17816/2074-0530-67749.

[5] Tan, R. Y., Wang, D., & Chen, C. (2018). Design of Multi-Level System Architecture Based on Software Bus Technology. Journal of Natural Science of Hunan Normal University, 41(04), 91-94.

[6] Wu, S. Q., Li, X. M., Wang, X., & Cai, H. X. (2021). Research on Key Negotiation in Adversarial Networks. Application Research of Computers, 38(05), 1539-1543. https: //doi.org/10.19734/j.issn.1001-3695.2020.08.0197.