



Research Article

Pre-Authenticated ML-KEM Key Banking with Exposure-Bounded SM4-GCM Rekeying for Real-Time Industrial IoT

Bakhtiyor Abdurakhimov¹ , Javokhir Abdurazzokov^{2,*}

¹Department of Applied Mathematics and Computer Analysis, National University of Uzbekistan named after Mirzo Ulugbek, Tashkent, Uzbekistan

²Department of Exact Sciences and Digital Technologies, Tashkent International University, Tashkent, Uzbekistan

Abstract

Industrial Internet of Things (IIoT) systems require cryptographic migration toward post-quantum key establishment without violating the tight timing constraints of sensing, control, and gateway communication. This paper proposes (Pre-Authenticated Rekey Key-Bank with Real-Time Exposure Control), a hybrid key-management architecture that combines NIST-standardized ML-KEM with an SM4-GCM authenticated data plane. The central idea is to move ML-KEM key generation and public-key delivery out of the latency-critical rekey interval. During low-utilization periods, a device creates a rolling bank of application-enforced single-activation ML-KEM public keys, commits the bank with a Merkle root, and authenticates that root once with ML-DSA. During online rekeying, the peer uses a previously authenticated bank slot, transmits only the ML-KEM ciphertext and compact context, and completes explicit key confirmation before activating the new SM4-GCM epoch. A hard exposure envelope based on key age, message count, protected-byte volume, and security risk prevents indefinite postponement, while a soft pressure function and reserve-aware scheduler place preparation work into available real-time slack. For ML-KEM-768, pre-positioning the 1184-byte encapsulation key reduces the core KEM material on the online path from 2272 bytes to 1088 bytes, an exact analytical reduction of 52.11%. A link-rate model shows that the removed public-key transfer is most valuable on constrained links, while the principal cost is protected storage for unused decapsulation keys. The paper provides protocol state rules, nonce construction, security propositions, analytical communication and memory results, and a reproducible MQTT/OPC UA hardware-validation protocol. The construction does not modify ML-KEM, ML-DSA, or SM4; its contribution is the lifecycle and scheduling architecture that integrates standardized primitives with real-time IIoT constraints.

Keywords

Industrial Internet of Things, Post-Quantum Cryptography, ML-KEM, SM4-GCM, Key Management, Real-Time Rekeying, Smart Manufacturing, Authenticated Encryption

1. Introduction

Industrial production systems increasingly connect sensors, programmable logic controllers, robotic cells, gateways, manufacturing execution systems, and cloud services. These

*Correspondence: Javokhir Abdurazzokov (javohirjon.1992@gmail.com)

Received: 20 September 2026; Accepted: 5 October 2026; Published: 23 October 2026



Copyright: © The Author(s), 2026. Published by Science Publishing Group. This is an **Open Access** article, distributed under the terms of the Creative Commons Attribution 4.0 License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited.

cyber-physical environments differ from ordinary web applications because a cryptographic event can be operationally harmful even when it is secure: a delayed control message may miss a process deadline, trigger retransmission, or force a controller into a safe state. At the same time, many industrial assets remain deployed for years, so confidentiality must be considered over a long horizon in which recorded traffic may be attacked later with more capable computers.

NIST standardized the Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) in FIPS 203 for post-quantum shared-secret establishment [1]. NIST SP 800-227 subsequently provided recommendations for the secure use of KEMs, including key derivation and key confirmation [2]. For ML-KEM-768, the standardized encapsulation key is 1184 bytes, the decapsulation key is 2400 bytes, the ciphertext is 1088 bytes, and the shared secret is 32 bytes [1]. These sizes are practical for many networks, but they become significant on low-rate links, during fleet-wide key rotation, and on endpoints with limited protected memory.

Recent work demonstrates that post-quantum mechanisms can be integrated into IIoT communication. Shahid *et al.* implemented ML-KEM and ML-DSA within TLS 1.3 for industrial authentication and evaluated the resulting overhead on Raspberry Pi hardware [6]. Almutairi and Sheldon showed that network impairment can dominate MQTT-over-TLS 1.3 handshake performance even when ML-KEM-768 is used [7]. Other studies have proposed risk-adaptive post-quantum authentication [8], hierarchical key-management schemes [9], server-assisted key distribution [10], and latency-aware separation of public-key refresh from high-frequency streaming [11]. These results suggest that the placement and timing of PQC operations are at least as important as the primitive choice.

This paper addresses a narrower systems question: *can the public-key preparation part of a post-quantum rekey be completed before the rekey is needed, while still enforcing a hard cryptographic exposure limit and preventing future-key misuse?* The proposed PREKEY-RT architecture answers this question by treating authenticated ML-KEM public keys as a controlled cryptographic inventory. A device prepares future key slots during low-utilization intervals; the peer stores the authenticated public bank; and online activation consumes exactly one slot.

The second design dimension is timing. A purely weighted risk score can postpone rekeying indefinitely when one factor is high but other factors are low. PREKEY-RT therefore separates a *hard exposure envelope* from a *soft scheduling pressure*. The hard envelope uses the maximum of normalized key age, packet count, byte count, and risk state. Reaching any hard limit creates a mandatory rekey. The soft pressure is a weighted function used only to decide whether an earlier refresh is desirable when sufficient slack exists.

The main contributions are: (1) a two-timescale ML-KEM lifecycle that moves key generation and public-key transfer out of the online rekey path; (2) a Merkle-committed rolling

bank authenticated once per generation with ML-DSA; (3) a single-activation slot state machine with transcript binding and explicit key confirmation; (4) a hard exposure deadline that cannot be hidden by an average score; (5) a reserve-aware scheduler for future-key preparation; and (6) exact communication, memory, and amortization formulas that distinguish analytical results from hardware results still requiring experimental validation.

2. Related Work and Research Gap

2.1. Post-Quantum Cryptography in IIoT

FIPS 203 defines ML-KEM-512, ML-KEM-768, and ML-KEM-1024 [1]. The middle parameter set is used in the main profile of this paper because it offers a practical balance between communication size and standardized post-quantum security strength. In industrial deployment studies, certificate and handshake bytes can be more visible than primitive execution time, especially when a gateway terminates the public-key layer [6, 7]. Recent reviews and implementation measurements further emphasize resource-aware deployment. Mishra *et al.* analyze IIoT security from a cryptographic perspective and identify post-quantum integration for constrained IIoT as a continuing systems challenge [13]. Halak *et al.* measured performance, energy, and computation costs of quantum-resistant schemes on embedded platforms [14], while Cruz-Piris *et al.* evaluated execution-time and power costs in IoT/IIoT scenarios [15]. These results motivate architectures that reduce latency-critical computation and communication rather than treating primitive choice as the only optimization variable. NIST's crypto-agility guidance likewise emphasizes adapting cryptography while preserving security and ongoing operation [16].

IHKEM addresses lightweight hierarchical key establishment for sensor networks but does not provide an ML-KEM-specific real-time activation path [9]. SAOKD moves expensive public-key work toward a powerful assisting server and demonstrates the importance of architectural placement for constrained endpoints [10]. PREKEY-RT differs by keeping standard peer-to-device ML-KEM encapsulation while changing when the device key pair is generated and when its public key is delivered.

2.2. Adaptive and Latency-Aware Rekeying

Trust-driven risk adaptation reduces unnecessary public-key operations by invoking PQC only when a trust threshold is crossed [8]. This improves efficiency but does not, by itself, impose independent limits on key age, packet count, or protected-byte volume. PQ-NeuroLink separates frequent symmetric updates from less frequent post-quantum refresh and selects parameters under latency-tail budgets [11]. That work provides a relevant precedent for keeping PQC away from a high-frequency data path. PREKEY-RT instead focuses on the

rekey event itself: the future public key is authenticated before the event, so online activation no longer needs key generation or public-key upload.

2.3. Pre-Authenticated Ephemeral Public Keys

Zajac showed that Merkle trees can authenticate sets of ephemeral public keys and reduce repeated signature overhead in IoT-oriented post-quantum protocols [12]. PREKEY-RT deliberately uses that idea as an enabling component rather than claiming Merkle authentication as new. The system contribution is its integration with standardized ML-KEM, a rolling bank lifecycle, exposure-derived security deadlines, reserve control, and an SM4-GCM industrial data plane.

2.4. Research Gap

The reviewed literature covers post-quantum IIoT handshakes, adaptive authentication, hierarchical key management, server assistance, and authenticated ephemeral key sets. A remaining gap is a reproducible real-time key lifecycle in which (i) future ML-KEM public keys are authenticated and distributed before they are needed, (ii) online activation consumes a single stateful slot, (iii) a hard exposure envelope forces rekeying even during sustained load, and (iv) future-key preparation is scheduled as maintainable cryptographic inventory. The novelty claim in this paper is intentionally limited to this integrated lifecycle and scheduler rather than any individual primitive.

3. Materials and Methods

3.1. System Model and Security Profile

Consider an IIoT deployment with constrained devices S_i , an edge gateway G , and a cloud or industrial control service C . A device may be a sensor, actuator, embedded controller, or a security endpoint attached to a programmable logic controller. The logical path is

$$S_i \leftrightarrow G \leftrightarrow C. \quad (1)$$

The peer P denotes the entity that encapsulates to the device ML-KEM public key; depending on policy, P can be the gateway or a cloud endpoint operating through the gateway.

Each protected flow has an operational deadline D_i . Let $L_i^{\text{base}}(t)$ denote the conservative latency estimate for sensing, application processing, queueing, networking, and ordinary symmetric protection. The instantaneous slack available for non-critical maintenance is

$$S_i(t) = D_i - L_i^{\text{base}}(t) - G_i, \quad (2)$$

where G_i is a guard margin. Offline key preparation is allowed only when $S_i(t)$ exceeds the conservative cost of the preparation step.

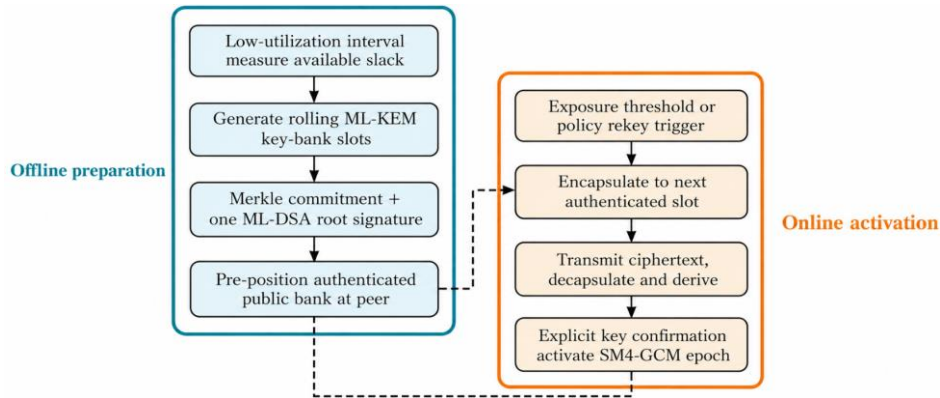


Figure 1. Two-timescale PREKEY-RT lifecycle. Future ML-KEM public keys are prepared and authenticated outside the latency-critical interval; online activation consumes one slot and starts a new SM4-GCM epoch.

Threat model and assumptions. The adversary may eavesdrop, record, inject, modify, replay, delay, and reorder network traffic. Harvest-now-decrypt-later is in scope. Temporary compromise of an active symmetric epoch is considered, but recovery claims assume that the long-term device identity credential and at least one unused protected ML-KEM secret slot remain uncompromised. Physical extraction of an unprotected key bank is outside the protocol proof and must be addressed by a secure element, trusted execution environment,

hardware-bound storage key, or equivalent protection.

The construction aims to provide confidentiality, integrity, replay resistance, context separation between devices and epochs, authenticated selection of future public keys, bounded exposure, and a defined recovery transition. Availability against a jammer that blocks all communication is not claimed.

SM4-GCM cryptographic profile. SM4 is a 128-bit block cipher. RFC 8998 explicitly defines AEAD_SM4_GCM, i.e., Galois/Counter Mode instantiated with SM4, for a TLS 1.3

ShangMi profile [4]. The RFC specifies a 16-octet SM4 key, a 12-octet nonce, and a 16-octet authentication tag, and it requires nonce uniqueness under a fixed key. GCM itself is an authenticated-encryption mode for an underlying block cipher [5].

Two qualifications are important. First, RFC 8998 is an Informational RFC and explicitly states that the ShangMi TLS profile is not endorsed as an IETF Standards Track recommendation [4]. Second, using ML-KEM for key establishment does not increase the 128-bit key width of SM4. Therefore, the paper describes a *post-quantum key-establishment architecture with an SM4-GCM data plane*, not a claim that the full construction inherits the same generic security margin as a 256-bit symmetric key.

3.2. PREKEY-RT Lifecycle and Offline Preparation

PREKEY-RT separates cryptographic maintenance into two timescales. During an offline preparation window, the device generates a bank of future ML-KEM key pairs, commits the public keys with a Merkle tree, signs the root once with its long-term ML-DSA credential, and transfers the authenticated public bank to the peer. During online activation, the peer encapsulates to the next unused slot, sends only the KEM ciphertext and compact activation context, receives key confirmation, and begins the new SM4-GCM epoch.

Offline key-bank preparation. For device S_i , bank generation g , and bank size B , the device creates

$$(pk_{i,g,j}, sk_{i,g,j}) \leftarrow \text{KeyGen}(\cdot), j = 0, \dots, B-1. \quad (3)$$

A slot is an application-enforced *single-activation* object; ML-KEM itself does not require encapsulation keys to be single-use. The lifecycle state is

$$\text{UNUSED} \rightarrow \text{PENDING} \rightarrow \text{CONSUMED}, \quad (4)$$

with rollback to UNUSED forbidden after successful decapsulation and confirmation.

Let T_{exp} be the bank expiry time. Each public slot is committed as

$$L_{i,g,j} = H(\text{PKSLOT} \parallel ID_i \parallel g \parallel j \parallel pk_{i,g,j} \parallel T_{\text{exp}}). \quad (5)$$

The leaves form a binary Merkle tree with root $R_{i,g}$. The signed commitment is

$$\sigma_{i,g} = \text{Sign}_{sk_i^{\text{ID}}} \left(H(\text{BANKROOT} \parallel ID_i \parallel g \parallel B \parallel T_{\text{exp}} \parallel R_{i,g}) \right). \quad (6)$$

The peer verifies $\sigma_{i,g}$ under the provisioned ML-DSA identity public key and stores the verified index-to-public-key mapping. If the complete bank is pre-positioned after verification, no Merkle authentication path is needed during online

activation. If leaves are delivered lazily, a path of $\lceil \log_2 B \rceil$ hash values accompanies the leaf.

3.3. Online Activation and Data Protection

Suppose slot (g, j) is verified and unused. The peer computes

$$(ct_{i,e}, z_{i,e}) \leftarrow \text{Encaps}(pk_{i,g,j}). \quad (7)$$

The activation record is

$$A_{i,e} = (ID_i, ID_P, g, j, e, r_e, h_{e-1}, ct_{i,e}), \quad (8)$$

where e is a monotonically increasing epoch number, r_e is a fresh activation nonce, and h_{e-1} binds the request to the previous accepted channel transcript or epoch state.

The device checks the bank generation, expiry, slot state, epoch monotonicity, and transcript binding before performing

$$\hat{z}_{i,e} = \text{Decaps}(sk_{i,g,j}, ct_{i,e}). \quad (9)$$

After confirmation, $sk_{i,g,j}$ and transient shared-secret material are erased, and the slot is permanently marked CONSUMED.

Key derivation and confirmation. Define the activation context

$$ctx_{i,e} = ID_i \parallel ID_P \parallel g \parallel j \parallel e \parallel r_e \parallel h_{e-1} \parallel H(\text{Policy}_i) \parallel H(FW_i). \quad (10)$$

The epoch root is derived from the KEM secret and activation transcript:

$$RK_{i,e} = \text{HKDF-Extract}(H(A_{i,e}), \hat{z}_{i,e}). \quad (11)$$

Domain-separated subkeys are

$$K_{i,e}^{\text{data}} = \text{Trunc}_{128} \left(\text{HKDF-Expand}(RK_{i,e}, \text{DATA} \parallel ctx_{i,e}, 32) \right). \quad (12)$$

$$K_{i,e}^{\text{conf}} = \text{HKDF-Expand}(RK_{i,e}, \text{CONF} \parallel ctx_{i,e}, 32). \quad (13)$$

$$K_{i,e}^{\text{rat}} = \text{HKDF-Expand}(RK_{i,e}, \text{RATCHET} \parallel ctx_{i,e}, 32). \quad (14)$$

The 128-bit truncation in (12) matches the SM4 key size. Explicit unilateral key confirmation is

$$KC_{i,e} = \text{MAC}_{K_{i,e}^{\text{conf}}}(\text{KC} \parallel H(A_{i,e}) \parallel R_{i,g}). \quad (15)$$

A bilateral profile repeats the confirmation in the reverse direction with a distinct label. This follows the general NIST recommendation that KEM-based key establishment use key

confirmation where appropriate [2].

SM4-GCM record protection. For record sequence number s , define a 96-bit nonce

$$N_{i,e,s} = \text{Encode}_{32}(e) \parallel \text{Encode}_{64}(s). \quad (16)$$

The associated data binds non-secret protocol state:

$$AAD_{i,e,s} = ID_i \parallel ID_P \parallel g \parallel j \parallel e \parallel s \parallel H(\text{Policy}_i). \quad (17)$$

Authenticated encryption is

$$(C_{i,e,s}, T_{i,e,s}) = \text{Enc}_{K_{i,e}^{\text{data}}}^{\text{SM4-GCM}}(N_{i,e,s}, M_{i,s}, AAD_{i,e,s}). \quad (18)$$

The counter s must never repeat within an epoch. An implementation must persist or safely reconstruct the epoch/sequence state across restart so that nonce reuse cannot occur under the same $K_{i,e}^{\text{data}}$.

3.4. Exposure Control and Reserve-Aware Scheduling

Let $a_i(t)$ be the current key age, $n_i(t)$ the number of protected records, $b_i(t)$ the protected-byte volume, and $r_i(t) \in [0, 1]$ a policy-defined risk indicator. Normalize them as

$$A_i(t) = \min\left(\frac{a_i(t)}{T_i^{\max}}, 1\right). \quad (19)$$

$$N_i(t) = \min\left(\frac{n_i(t)}{N_i^{\max}}, 1\right). \quad (20)$$

$$V_i(t) = \min\left(\frac{b_i(t)}{B_i^{\max}}, 1\right). \quad (21)$$

$$R_i(t) = \min\left(\frac{r_i(t)}{R_i^{\max}}, 1\right). \quad (22)$$

The hard exposure state is

$$E_i(t) = \max\{A_i(t), N_i(t), V_i(t), R_i(t)\}. \quad (23)$$

A mandatory activation is created when $E_i(t) \geq 1$. For early opportunistic rekeying, define

$$P_i(t) = w_A A_i + w_N N_i + w_V V_i + w_R R_i, \quad \sum w_* = 1, \quad (24)$$

with $w_* \geq 0$. A soft rekey may be requested when $P_i(t) \geq \theta_s$ and a verified bank slot is available.

Let λ_n and λ_b be recent record and byte rates. The estimated remaining time to a hard bound is

$$\tau_i^{\text{sec}} = \min\left\{T_i^{\max} - a_i, \frac{N_i^{\max} - n_i}{\lambda_n + \epsilon}, \frac{B_i^{\max} - b_i}{\lambda_b + \epsilon}, \tau_i^{\text{risk}}\right\}, \quad (25)$$

where $\epsilon > 0$ avoids division by zero and τ_i^{risk} is a policy

deadline when risk evidence is elevated. The cryptographic security deadline is

$$D_i^{\text{sec}}(t) = t + \max(0, \tau_i^{\text{sec}}). \quad (26)$$

The scheduler attempts opportunistic activation when slack exists but is not allowed to defer beyond D_i^{sec} .

Reserve-aware bank replenishment. Let $Q_i(t)$ be the number of verified unused slots, $Q_i^{\min}$ a safety reserve, and Q_i^{target} a preferred reserve. Background generation is enabled when

$$Q_i(t) < Q_i^{\text{target}} \quad \text{and} \quad S_i(t) \geq C_{\text{kg},i}^{p99} + G_i, \quad (27)$$

where $C_{\text{kg},i}^{p99}$ is a conservative key-generation cost. A simple reserve pressure is

$$\Pi_i(t) = \frac{Q_i^{\text{target}} - Q_i(t)}{Q_i^{\text{target}} - Q_i^{\min+1}}. \quad (28)$$

Activation jobs are ordered by earliest D_i^{sec} ; bank-preparation jobs are ordered by descending Π_i subject to the operational slack condition.

3.5. Protocol Algorithms

Algorithm 1: Offline Prepare (S_i, B, g)

1. Generate B ML-KEM key pairs using (3).
2. Store each $sk_{i,g,j}$ in protected one-time slot storage.
3. Compute leaves by (5) and the Merkle root $R_{i,g}$.
4. Sign the bank commitment by (6).
5. Transfer the authenticated public bank during a maintenance window.
6. Mark all slots UNUSED; set $Q_i = B$.

Algorithm 2: Activate Epoch (P, S_i, g, j, e)

1. Verify that slot (g, j) is authenticated, unexpired, and unused.
2. Run ML-KEM encapsulation by (7).
3. Send activation record (8).
4. Device checks freshness, epoch monotonicity, and transcript binding.
5. Device decapsulates by (9).
6. Both parties derive (11) – (14).
7. Exchange explicit confirmation using (15).
8. Mark slot CONSUMED, erase its secret key, and activate SM4-GCM.
9. Erase the old epoch after the configured overlap/replay window.

Algorithm 3: Exposure Reserve Scheduler ()

1. Update $E_i(t)$ and $P_i(t)$ from (23) and (24).
2. Estimate D_i^{sec} using (26).
3. If $E_i \geq 1$, enqueue mandatory activation.
4. Else if $P_i \geq \theta_s$ and a slot exists, enqueue soft activation.
5. Order activation jobs by earliest security deadline.

6. If $Q_i < Q_i^{\text{target}}$ and (27) holds, generate new slots offline.

7. If $Q_i \leq Q_i^{\text{min}}$, raise maintenance priority and suppress non-essential soft rekeys if policy permits.

4 Security Analysis

4.1. Authenticity of a Bank Slot

Assume the ML-DSA signature on the bank commitment is existentially unforgeable and the Merkle hash is collision resistant. An adversary cannot replace an authenticated public-key slot with a different value while preserving successful verification under the same signed root, except with negligible probability.

Argument. A successful substitution requires either a valid signature on a different root without the signing key or a hash collision that maps a modified leaf/path to the original root. Both events contradict the stated assumptions. When the complete bank is pre-positioned, the peer additionally stores the verified index-to-public-key mapping and rejects mismatches.

4.2. Context Separation

If HKDF behaves as a pseudorandom key derivation function and $ctx_{i,e} \neq ctx_{i,e'}$, knowledge of one derived traffic key does not provide an efficient method for computing the other absent compromise of the relevant epoch root.

The explicit labels DATA, CONF, and RATCHET prevent cross-purpose key reuse.

4.3. Replay and Rollback Resistance

An activation is rejected if its bank slot is already consumed, its epoch does not advance monotonically, the bank is expired, or the transcript binding is inconsistent with the accepted channel state. Data records are rejected when the sequence number has already been accepted or falls outside the replay window. A replayed activation therefore cannot deliberately reactivate an erased secret slot under the defined state machine.

4.4. Nonce Uniqueness

Equation (16) is injective over the pair (e, s) for the configured field widths. Since a new data key is derived for each accepted epoch and s is monotonic within the epoch, nonce reuse under a fixed key is prevented provided that persistent state does not roll back. This condition is critical because RFC 8998 explicitly warns that nonce reuse with the same SM4-GCM key destroys GCM security properties [4].

4.5. Forward Epoch Separation and Recovery

If bank slots are independently generated, a consumed slot

secret is securely erased, and the prior epoch root is erased after transition, compromise of a later epoch does not directly reveal an independent earlier ML-KEM shared secret. This is forward *epoch separation*, not a stronger claim about all forms of endpoint compromise. If an attacker learns the current symmetric state, recovery requires that the attacker lose access and that a fresh authenticated unused ML-KEM slot be activated while the long-term identity credential and that unused slot remain uncompromised.

4.6. Key-Bank Starvation Condition

Let $C_i(t)$ be the cumulative number of consumed slots and $G_i(t)$ the cumulative number of new valid slots added by time t . Then

$$Q_i(t) = Q_i(0) + G_i(t) - C_i(t). \quad (29)$$

For minimum reserve Q_i^{min} , bank starvation does not occur over horizon $[0, H]$ if

$$Q_i(0) + G_i(t) - C_i(t) \geq Q_i^{\text{min}}, \forall t \in [0, H]. \quad (30)$$

This converts bank sizing into an inventory problem: deployment engineering must estimate worst-case rekey demand and the longest period in which background generation may be unavailable.

4.7. Stored Future Secret Keys

Precomputation trades online latency for protected storage. Theft of an unused decapsulation key can expose a future ciphertext addressed to that slot. Consequently, the bank should be small enough to fit protected memory, have a bounded expiry, be erased slot-by-slot, and be renewed frequently. This limitation is fundamental and is treated as a first-class design trade-off rather than hidden overhead.

5. Analytical Results

5.1. Online KEM Communication

A conventional fresh ML-KEM activation that generates a new key pair at rekey time must transfer the encapsulation key and return the ciphertext. The core online bytes are

$$O_{\text{conv}} = |pk| + |ct|. \quad (31)$$

With a pre-positioned authenticated slot, the core KEM material is

$$O_{\text{pre}} = |ct|. \quad (32)$$

Hence the exact core-byte reduction is

$$\rho = \frac{|pk|}{|pk|+|ct|} \times 100\%. \quad (33)$$

For ML-KEM-768, $|pk| = 1184$ and $|ct| = 1088$, so

$\rho = 52.11\%$. Compact activation metadata and key-confirmation bytes are common protocol costs and are intentionally excluded from this core comparison.

Table 1. ML-KEM sizes and online core KEM-byte reduction under pre-positioning.

Set	$ pk $ (B)	$ sk $ (B)	$ ct $ (B)	Reduction (%)
ML-KEM-512	800	1632	768	51.02
ML-KEM-768	1184	2400	1088	52.11
ML-KEM-1024	1568	3168	1568	50.00

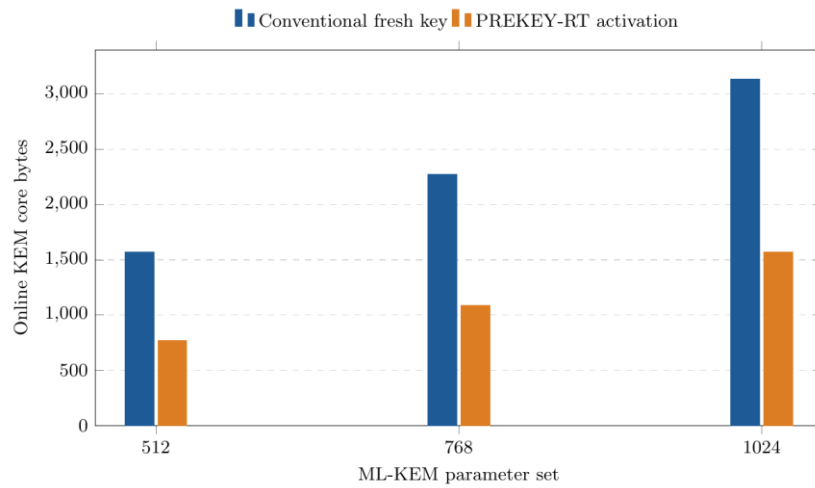


Figure 2. Analytical online KEM core bytes. The public-key upload is moved to the offline preparation phase.

5.2. Link-Rate Sensitivity

Let R be the link rate in bit/s and d the representative one-way propagation/queueing delay. Excluding cryptographic execution and the same confirmation step on both designs,

$$T_{\text{conv}}^{\text{net}} = \frac{8(|pk|+|ct|)}{R} + 2d. \quad (34)$$

$$T_{\text{pre}}^{\text{net}} = \frac{8|ct|}{R} + d. \quad (35)$$

The additional propagation term reflects that a fresh public key must first reach the encapsulating peer. Table 2 uses ML-KEM-768 and $d = 5$ ms as an analytical sensitivity model, not a hardware benchmark.

Table 2. Analytical ML-KEM-768 KEM-transfer time with 5 ms one-way propagation/queueing.

Rate (Mbit/s)	Conventional (ms)	PREKEY-RT (ms)	Saved (ms)
0.25	82.704	39.816	42.888
1	28.176	13.704	14.472

Rate (Mbit/s)	Conventional (ms)	PREKEY-RT (ms)	Saved (ms)
10	11.818	5.870	5.947
100	10.182	5.087	5.095

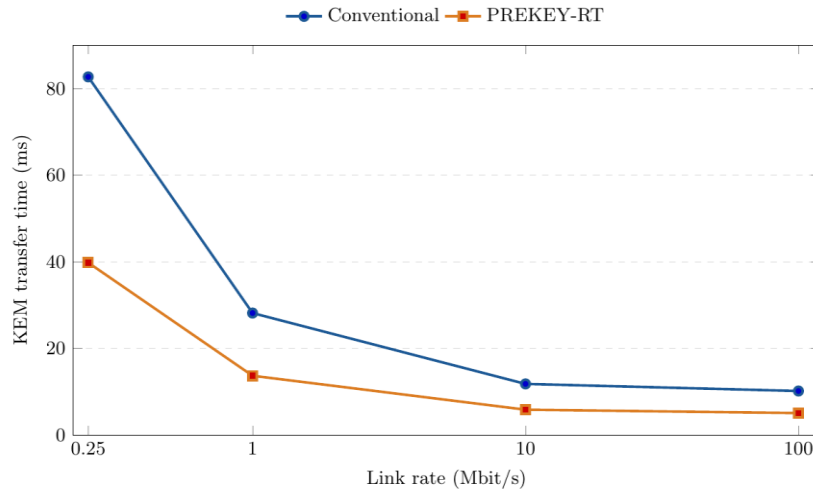


Figure 3. Analytical sensitivity of the ML-KEM-768 transfer interval to link rate. Cryptographic execution and common key-confirmation cost are excluded.

5.3. Protected Memory Cost

If the device stores unused ML-KEM decapsulation keys directly, the raw key-bank footprint is

$$M_{\text{bank}} = B |sk|. \quad (36)$$

For ML-KEM-768 this is 9.38 KiB at $B = 4$, 18.75 KiB at $B = 8$, 37.50 KiB at $B = 16$, and 75.00 KiB at $B = 32$. A seed-regeneration variant can reduce persistent memory but would return key generation to the activation path, so it is a different point in the latency-memory trade-off.

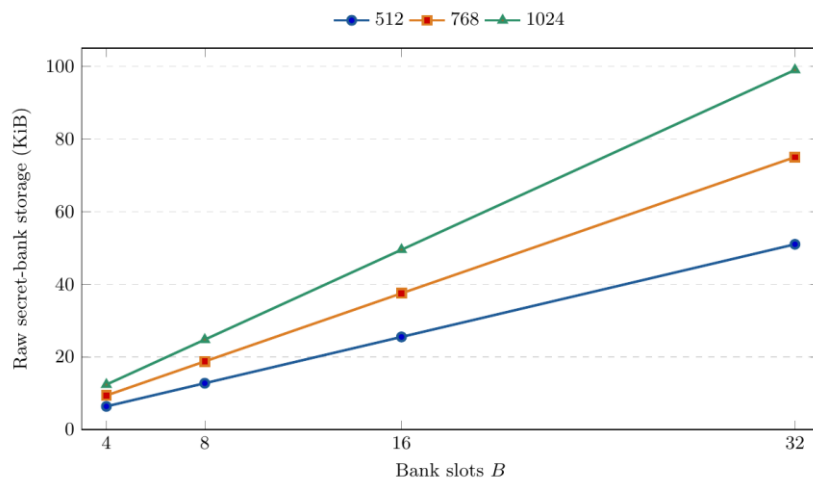


Figure 4. Raw protected storage required for unused ML-KEM secret slots.

5.4. Amortized ML-DSA Authentication

FIPS 204 gives an ML-DSA-65 signature size of 3309 bytes [3]. Signing every future ML-KEM public key separately would repeat this cost. Signing one Merkle root per bank yields an amortized signature cost

$$O_{\text{sig/key}}(B) = \frac{3309}{B} \text{ bytes per committed slot} \quad (37)$$

ignoring the small root and bank metadata. For $B = 16$, this is 206.8 bytes/slot; for $B = 32$, 103.4 bytes/slot. If the entire public bank is pre-positioned after root verification, no Merkle path appears in the online activation record.

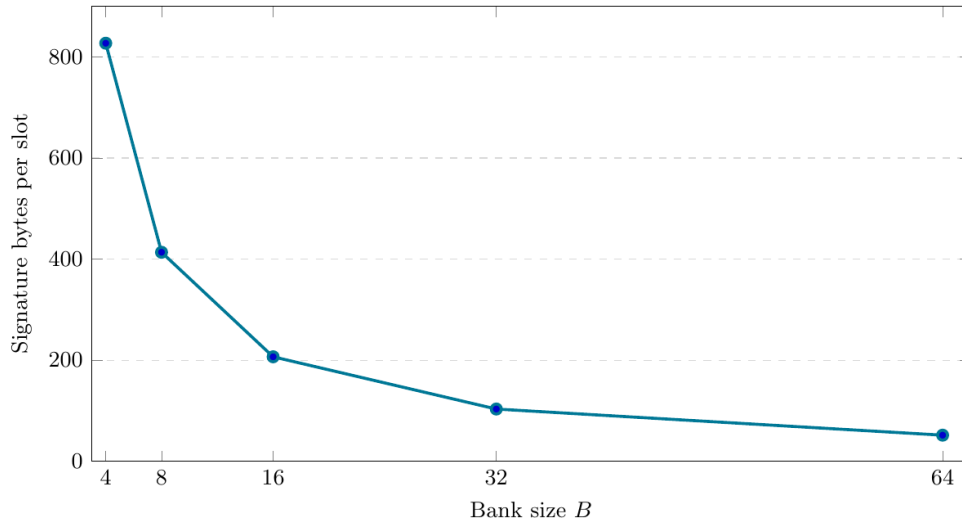


Figure 5. Amortized ML-DSA-65 signature bytes per public bank slot when one root signature authenticates the whole bank.

5.5. SM4-GCM Record Overhead

RFC 8998 specifies a 16-byte authentication tag and a 12-byte nonce for AEAD_SM4_GCM [4]. In PREKEY-RT, the nonce can be reconstructed from the authenticated epoch and sequence number, so the full nonce need not be transmitted if both peers share the same state representation. A compact record therefore adds primarily the 16-byte tag plus protocol-specific sequence/epoch fields. The design must never infer that removing an explicit nonce field permits nonce reuse; uniqueness remains a hard invariant.

6. Experimental Validation Protocol

6.1. Hardware Tiers

A publication-grade validation should evaluate at least three tiers: (i) an MCU-class node such as Cortex-M4/M33 or ESP32-S3, (ii) an ARM64 Linux gateway such as Raspberry Pi 4/5, and (iii) an x86-64 server. Each report should include the exact ML-KEM/ML-DSA library version, compiler, optimization flags, assembly acceleration, SM4-GCM implementation, CPU frequency policy, and protected-storage mechanism.

6.2. Communication Stacks and Baselines

MQTT and OPC UA should be evaluated as separate application stacks. Four baselines are recommended: B1, conventional online ML-KEM key generation and public-key upload; B2, fixed-period two-tier rekeying without a pre-authenticated bank; B3, risk-triggered post-quantum reauthentication without hard age/packet/byte bounds; and B4, the complete PREKEY-RT lifecycle. Where source code is available, recent systems should be reproduced rather than compared only with published numbers [8, 11].

6.3. Metrics

Primary timing metrics are ML-KEM key-generation, encapsulation and decapsulation time, total activation latency, p50/p95/p99 application latency, and deadline-miss ratio. Resource metrics are peak RAM, flash/ROM, protected bank footprint, CPU cycles, processor utilization, and measured energy per rekey. Communication metrics are bytes per activation, fragmentation count, retransmissions, and amortized bytes per protected record. Security-operation metrics include maximum key age, maximum records and bytes per epoch, mandatory versus opportunistic rekey count, reserve distribution, and starvation events.

6.4. Workloads and Statistical Procedure

Experiments should cover periodic sensing, burst telemetry, alarm bursts, synchronized fleet rekey, temporary gateway overload, elevated anomaly/risk state, and forced suppression of background replenishment. Each configuration should be repeated enough times to estimate tail latency with confidence intervals. Because network latency is often non-Gaussian, medians and quantiles should be reported alongside means. Paired traces should use paired nonparametric tests or bootstrap confidence intervals, and effect sizes should accompany p -values.

7. Discussion

7.1. What Is and Is Not New

Combining ML-KEM with a symmetric cipher is not a sufficient novelty claim. Adaptive rekeying is also not individually new, and Merkle authentication of ephemeral public keys predates this work [8, 11, 12]. The contribution claimed here is narrower: future standardized ML-KEM public keys are managed as authenticated inventory; online activation consumes exactly one bank slot; hard exposure bounds determine the latest permissible activation time; and a separate reserve scheduler determines when future slots are created. This separation turns key maintenance into two coupled scheduling problems rather than one large real-time handshake.

7.2. Interpretation of the Analytical Gain

The 52.11% ML-KEM-768 result is an exact reduction in *core online KEM bytes*, not a claim of 52.11% end-to-end latency or energy reduction. Actual improvement depends on link rate, propagation, queueing, key-generation speed, re-transmissions, and the size of common confirmation/protocol fields. The link model in Table 2 isolates only the communication effect of pre-positioning the public key.

7.3. SM4-GCM Scope and Post-Quantum Terminology

SM4-GCM is technically well-defined: RFC 8998 defines AEAD_SM4_GCM and provides test vectors [4]. However, the RFC is Informational and not an IETF endorsement of the ShangMi TLS profile. Moreover, SM4 uses a 128-bit key. Consequently, the manuscript deliberately avoids describing the entire data plane as having the same generic post-quantum security margin as ML-KEM-768. Deployments that require a larger symmetric key margin can preserve the PREKEY-RT bank/scheduler design while replacing the data-plane AEAD with an approved 256-bit-key alternative.

7.4. Availability, Depletion, and Trigger Abuse

An attacker may try to force rekey requests to consume bank slots. The device and peer must therefore reject unauthenticated rekey triggers, separate soft risk evidence from hard cryptographic limits, rate-limit extraordinary policy events, and never mark a slot consumed until the activation has reached the confirmation point. If the reserve is exhausted, industrial policy may enter a safe state, defer non-critical traffic, or perform emergency online key generation.

7.5. Side-Channel and Storage Considerations

Precomputation does not provide side-channel resistance. ML-KEM and SM4-GCM implementations on embedded devices must use hardened libraries and secure random generation. The stored future secret bank enlarges the amount of sensitive material resident on the endpoint; a practical deployment should therefore optimize bank size jointly over memory, replenishment opportunities, expected rekey frequency, ex-traction risk, and bank expiry.

8. Conclusions

This paper proposed PREKEY-RT, a pre-authenticated ML-KEM key-bank and exposure-bounded real-time rekeying architecture for Industrial IoT. The method moves ML-KEM key generation and public-key distribution to low-utilization intervals, authenticates a rolling public bank with one ML-DSA-signed Merkle commitment, and reduces the online activation to ML-KEM ciphertext delivery, decapsulation, context-bound key derivation, and key confirmation. For ML-KEM-768, this moves the 1184-byte public key off the critical path and reduces the online core KEM material from 2272 bytes to 1088 bytes, an exact 52.11% analytical reduction.

The second contribution is a scheduling model that separates hard exposure from soft pressure. Key age, record count, protected-byte volume, and risk each retain an independent hard bound, while a weighted pressure value is used only for opportunistic early refresh. A reserve controller schedules future-key generation in available slack and exposes the protected-memory cost explicitly.

The data-plane profile uses SM4-GCM. Its existence and parameters were verified against RFC 8998: SM4-GCM is defined as an AEAD construction with a 16-octet key, 12-octet nonce, and 16-octet tag. The paper also states the RFC's Informational status and avoids conflating ML-KEM post-quantum key establishment with a 256-bit symmetric security margin. The next required step is hardware validation on MCU, ARM64 gateway, and server platforms under MQTT and OPC UA workloads, with direct comparison against fresh online ML-KEM, fixed-period two-tier rekeying, and risk-triggered reauthentication.

Abbreviations

IIoT	Industrial Internet of Things
PQC	Post-Quantum Cryptography
KEM	Key-Encapsulation Mechanism
ML-KEM	Module-Lattice-Based Key-Encapsulation Mechanism
ML-DSA	Module-Lattice-Based Digital Signature Algorithm
AEAD	Authenticated Encryption with Associated Data
HKDF	HMAC-based Key Derivation Function
MQTT	Message Queuing Telemetry Transport
OPC UA	Open Platform Communications Unified Architecture

Author Contributions

Bakhtiyor Abdurakhimov: Resources, Supervision, Validation, Writing – review & editing

Javokhir Abdurazzokov: Conceptualization, Formal Analysis, Methodology, Software, Visualization, Writing – original draft

Funding

This work was partially supported by the Ministry of Higher Education, Science and Innovation of the Republic of Uzbekistan under grant number AL-9624115223 and partially supported by the Ministry of Digital Technologies of the Republic of Uzbekistan under scientific project number 36 entitled "Development of key generation algorithms for establishing secure communication at the transport layer".

Data Availability Statement

The analytical data reported in the manuscript are generated directly from the cited standardized ML-KEM and ML-DSA object sizes and the equations stated in the paper. Reproduction tables and scripts are included in the accompanying project package. Hardware measurements are not represented as collected results.

Conflicts of Interest

The authors declare no conflicts of interest.

References

- [1] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. FIPS 203. 2024. <https://doi.org/10.6028/NIST.FIPS.203>
- [2] Alagic, G., Barker, E., Chen, L., Moody, D., Robinson, A., Silberg, H., Waller, N. *Recommendations for Key-Encapsulation Mechanisms*. NIST Special Publication 800-227. 2025. <https://doi.org/10.6028/NIST.SP.800-227>
- [3] National Institute of Standards and Technology. *Module-Lattice-Based Digital Signature Standard*. FIPS 204. 2024. <https://doi.org/10.6028/NIST.FIPS.204>
- [4] Yang, P. ShangMi (SM) Cipher Suites for TLS 1.3. RFC 8998. 2021. <https://doi.org/10.17487/RFC8998>
- [5] Dworkin, M. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. NIST Special Publication 800-38D. 2007. <https://doi.org/10.6028/NIST.SP.800-38D>
- [6] Shahid, A. B., Mansoor, K., Bangash, Y. A., Iqbal, W., et al. Post-quantum cryptographic authentication protocol for industrial IoT using lattice-based cryptography. *Scientific Reports*. 2026, 16, 9582. <https://doi.org/10.1038/s41598-025-28413-8>
- [7] Almutairi, M. G., Sheldon, F. T. Network conditions dominate MQTT-over-TLS 1.3 performance: A comparative evaluation of ML-KEM-768, RSA-2048, and ECC-P256. *Electronics*. 2026, 15(17), 3914. <https://doi.org/10.3390/electronics15173914>
- [8] Shreyas, A., Arun Kumar, B. R., Manasa, U. Trust-Driven Risk-Adaptive Post-Quantum Authentication for Resource-Constrained IoT Systems. In: *2026 IEEE 18th International Conference on Computational Intelligence and Communication Networks (CICN)*. 2026. <https://doi.org/10.1109/CICN70047.2026.11594135>
- [9] Jain, K., Singh, A. IHKEM: A post-quantum ready hierarchical key establishment and management scheme for wireless sensor networks. *Microprocessors and Microsystems*. 2025, 118, 105205. <https://doi.org/10.1016/j.micpro.2025.105205>
- [10] Alqattan, A., Mahdi, M., Dimitriou, T. SAOKD: Server-Assisted Oblivious Key Distribution for Resource-Constrained, Post-Quantum IoT Devices. In: *2026 22nd IEEE International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. 2026, pp. 246–254. <https://doi.org/10.1109/DCOSS-IoT69657.2026.00038>
- [11] Qayyum, T., Tariq, A., Belkacem, A. N. PQ-NeuroLink: latency-aware post-quantum-ready secure communication for distributed brain-computer interfaces. *Frontiers in Systems Neuroscience*. 2026, 20, 1891041. <https://doi.org/10.3389/fnsys.2026.1891041>
- [12] Zajac, P. Ephemeral Keys Authenticated with Merkle Trees and Their Use in IoT Applications. *Sensors*. 2021, 21(6), 2036. <https://doi.org/10.3390/s21062036>
- [13] Mishra, N., Islam, S. K. H., Zeadally, S. A survey on security and cryptographic perspective of Industrial-Internet-of-Things. *Internet of Things*. 2024, 25, 101037. <https://doi.org/10.1016/j.iot.2023.101037>

- [14] Halak, B., Gibson, T., Henley, M., Botea, C.-B., Heath, B., Khan, S. Evaluation of Performance, Energy, and Computation Costs of Quantum-Attack Resilient Encryption Algorithms for Embedded Devices. *IEEE Access*. 2024, 12, 8791–8805. <https://doi.org/10.1109/ACCESS.2024.3350775>
- [15] Cruz-Piris, L., Marín-López, A., Alvarez-Campana, M., Sanz, M., Moreno, J. I., Arroyo, D. Measuring the impact of post quantum cryptography in Industrial IoT scenarios. *Internet of Things*. 2025, 34, 101793. <https://doi.org/10.1016/j.iot.2025.101793>
- [16] Barker, E., Chen, L., Cooper, D., Moody, D., Regenscheid, A., Souppaya, M., Newhouse, W., Housley, R., Turner, S., Barker, W., Kent, K. *Considerations for Achieving Crypto Agility: Strategies and Practices*. NIST Cybersecurity White Paper 39. 2025. <https://doi.org/10.6028/NIST.CSWP.39>