

An Empirical Study of Post-Quantum Cryptography Adoption in Open-Source Software

Anonymous Author(s)
Anonymous Institution(s)

Abstract—Post-quantum cryptography (PQC) standards establish algorithms for quantum-safe systems, but empirical evidence about how software projects implement and maintain PQC support remains limited. We investigate this implementation and maintenance work by mining 114,361 commits and 5,713 issues across 179 open-source repositories. Human annotation and validated LLM classification label more than 47,000 commits and 5,600 issues, and time-series models relate PQC activity to NIST standardization milestones. Maintenance of existing implementations accounts for about 90% of scheme-level changes, whereas new implementations account for less than 4%. A taxonomy of 11 recurring technical challenges shows that build, packaging, and dependency failures and functional defects make up half of all categorized issues and recur in more projects than any other category. After NIST finalized the first standards in 2024, standardized scheme names appeared in more than 80% of PQC changes, although predecessor names such as Kyber persisted. Development activity per repository also increased, but first adoptions did not measurably accelerate. These results characterize PQC adoption as a continuing maintenance and integration effort. Migration plans should therefore budget for sustained maintenance, invest in build reliability and integration testing, and track both standardized and predecessor scheme names. We make the dataset, taxonomy, annotations, and analysis pipeline publicly available.

Index Terms—post-quantum cryptography, mining software repositories, empirical software engineering, cryptographic migration

I. INTRODUCTION

A sufficiently large quantum computer would break the public-key cryptography used in TLS, code signing, software updates, and VPNs [1]. The U.S. National Institute of Standards and Technology (NIST) launched a multi-round standardization process [2], finalizing ML-KEM, ML-DSA, and SLH-DSA in August 2024 [3] and selecting HQC for standardization in March 2025 [4]. Adoption cannot wait for a cryptographically relevant quantum computer because adversaries can retain encrypted traffic for later decryption, while replacing cryptography across software ecosystems takes years [5], [6].

Prior studies measure whether shipped applications contain PQC [7] or observe negotiated network protocols [8], [9]. Industry reports from Meta, Apple, AWS, Chromium, and Microsoft document performance, interoperability, dependency, and rollout constraints in their own systems [10]–[15]. These studies and reports document deployment but do not trace how developers introduce schemes, combine them with classical cryptography, resolve implementation problems, and respond to changing standards within repositories.

Repository histories and developer discussions capture this process. Prior empirical work examines classical cryptographic API misuse [16], [17] and quantum computing software [18]–[20]. Static analysis of Python, Java, and Go repositories measures where PQC APIs appear [21], but, to the best of our knowledge, no prior study characterizes how PQC support is implemented, maintained, and discussed across linked commits and developer threads.

We study 179 GitHub repositories spanning PQC-focused projects and general-purpose systems with PQC activity through three research questions.

- **RQ1.** Which PQC schemes are implemented and maintained in open-source software, and are they deployed as pure-PQC or hybrid constructions?
- **RQ2.** What technical issues do developers encounter while implementing, integrating, and migrating to PQC?
- **RQ3.** How does open-source PQC adoption evolve around NIST standardization milestones?

RQ1 and RQ3 analyze commits reachable from a default branch or included in a merged pull request. RQ2 analyzes issues with their comments and resolving pull requests, when available. Human annotation and validated classifiers distinguish PQC adoption and technical challenges from keyword matches.

Contributions. We provide three empirical contributions.

- **A characterization of PQC adoption.** We quantify standardized and alternative schemes, distinguish hybrid from pure-PQC deployment, and compare adoption across project and language ecosystems.
- **A taxonomy of recurring technical challenges.** We identify technical challenges from developer threads and measure their frequency and time to recorded closure.
- **A temporal analysis of PQC development.** We measure changes in PQC adoption around NIST standardization milestones.

Section II presents the study design. Section III reports the methods and results for each RQ. Section IV discusses the findings, Section V reviews related work, Section VI addresses threats to validity, and Section VII concludes.

II. STUDY DESIGN

We combined repository mining, human annotation, and validated LLM classification to study PQC adoption and technical challenges (Figure 1). Section III explains the approach for each RQ.

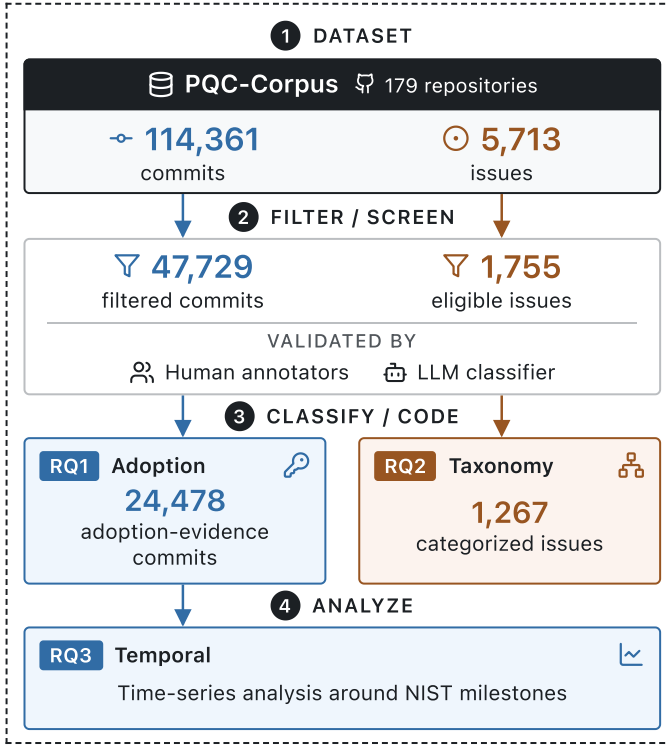


Fig. 1. Overview of our study.

A. Data Collection

We collected commits, pull requests (PRs), and issues from 179 public GitHub repositories.

Repository selection. We manually selected 50 repositories from *Awesome Quantum Software*, *Awesome PQC Resources*, *Awesome Post-Quantum*, and PQClean’s recommended projects [22]. The curated set included general-purpose projects such as OpenSSL [23] and Bouncy Castle [24], extending coverage beyond repositories that advertised PQC in their descriptions.

We expanded this set through three GitHub channels in September 2026. We searched 25 PQC topics with `topic:` and repository names, descriptions, and topics with `in:name,description,topics` using a 95-term PQC vocabulary. We also retrieved repositories owned by PQC organizations, including `open-quantum-safe`, `pq-code-package`, and `mupq`. We split topic searches by creation year over 2016–2026 to stay below GitHub’s 1,000-result limit. For metadata searches, we joined up to six unambiguous terms, such as ML-KEM, FrodoKEM, and FIPS 203, with OR and paired ambiguous names such as Falcon with a context phrase such as `post-quantum`.

We retained discovered repositories with at least 30 stars and PQC-related metadata, excluding forks and archived or disabled repositories. The curated set was exempt from these filters. Deduplication and screening yielded 129 additional repositories after 18 manual exclusions for reasons such as misleading names, absent PQC content, or non-software material.

Collection scope. We measured the proportion of each repository’s PRs and issues matching 90 PQC terms in their titles, bodies, or comments, excluding five ambiguous discovery terms and counting each record once. We set the collection threshold at 0.148, the lowest matching share among 24 PQC-focused reference repositories. All 17 general-purpose reference repositories fell below this threshold.

We collected all PRs, issues, and commits from the 81 repositories at or above this threshold (Corpus A). For the 87 below it (Corpus B), we collected PRs and issues only when their title or body matched a keyword, and commits with a message match or a link to a selected PR. The remaining 11 repositories had no PRs or issues and contributed only Git commit histories (Corpus C).

We preserved Git histories in frozen mirrors for RQ1 and RQ3. We used GitHub’s REST API to retrieve PR and issue records and its GraphQL API to link PRs to commits and the issues they closed. We retrieved all comment and review pages for selected threads without further keyword filtering. We collected records over September 21–24, 2026, retaining author types and original timestamps, including those of reused PR and issue inventories.

We retrieved 114,361 repository–commit pairs, 11,919 PRs, and 5,713 issues as candidates for classification (Section III). The artifact provides full queries, selection decisions, and collection history.

B. Commit Selection and Filtering

We retained 73,076 repository–commit pairs reachable from the default branch or belonging to a merged PR. We removed 2,175 rebased copies by matching patches and commit metadata. Using patch identifiers following German et al. [25], we removed 2,443 merge commits that repeated another patch and 1,587 original PR commits whose changes were preserved by a squash merge on the default branch. From the remaining 66,871 commits, we removed 3,067 merges whose branch tips were already represented.

We excluded 11,375 inherited commits from the OQS OpenSSH fork with no PQC term in the commit message, linked PR or issue text, file paths, or changed lines. We retained changes mentioning terms such as Kyber, ML-KEM, liboqs, or sntrup, leaving 2,256 commits from the fork. Applying the same keyword check to repositories whose names, descriptions, and topics did not indicate PQC removed another 1,934 commits.

Of the 50,495 remaining commits, we labelled 2,766 as non-PQC by rule. These comprised empty commits and commits with no PQC term in the diff that were bot-authored or changed only CI configuration, lock files, or repository metadata such as licence files. This left 47,729 commits from 163 repositories for classification.

C. Analysis Overview

Human annotation established reference labels and the technical-challenge taxonomy. We classified the remaining records with validated models, then analyzed commit labels

for scheme adoption and temporal trends and issue labels for recurring challenges and time to recorded closure.

D. LLM-Based Validation

Prior PQC evaluation addressed code-fragment migration [26], not commit and issue classification. Without a standardized benchmark for these tasks, we compared eight proprietary and open-weight models released between March 17 and October 7, 2026. These were Gemini 3.8 Flash [27], Claude Haiku 5.5 [28], GPT-5.4 mini [29], Muse Spark 1.3 [30], GLM-5.3 [31], Kimi K2.7 Code [32], Qwen3.8-Flash [33], and Mistral Medium 3.5 [34].

We ran the comparisons through OpenRouter¹ between October 5 and 11, 2026. We used medium reasoning effort where supported, except for GLM-5.3, which used high effort. Output limits were 8,192 tokens, increased to 32,768 for GLM-5.3 and Kimi K2.7 Code. The artifact provides API identifiers and inference settings following reporting guidance [35].

Within each comparison, models received identical codebooks, prompts, evidence, and output schemas. We used development data for model selection and prompt refinement, then froze the model, prompt, and evaluation criteria before a single held-out evaluation to limit selection bias [36]. We assessed precision, recall, macro-F1, and per-class F1, reporting class counts and bootstrap confidence intervals. This protocol followed recommendations for LLM-supported software-engineering research [35], [37], [38]. We classified the corpus after the model met the acceptance criteria in Sections III-A and III-B.

III. EMPIRICAL STUDY

We examined scheme adoption, recurring technical challenges, and temporal trends using classifiers evaluated against human annotations. Each RQ reports its approach, results, and observations.

A. RQ1 Adoption Landscape

RQ1 examined which PQC schemes projects implemented or maintained and whether they used pure-PQC or hybrid constructions. We classified commits and aggregated their labels into repository–scheme adoption evidence.

1) *Approach: Unit and evidence.* We analyzed repository–commit pairs from the filtered corpus in Section II-B. The classifier received patches and changed paths, commit messages, linked merged PRs, and issues closed by those PRs. The prompt prioritized patch evidence over prose and prohibited reliance on repository reputation or external knowledge.

We assigned each commit a PQC-relevance label and each changed scheme an adoption status, deployment mode, classical hybrid partner, and confidence level. Adoption statuses were *implemented*, *maintenance*, *planned*, *discussed*, *deprecated* or *removed*, *rejected* or *reverted*, and *unclear*. Only the first two counted as adoption evidence. We coded new working support as *implemented*; benchmarks, test vectors, regenerated files, and partial changes as *maintenance*; and

scanners, catalogues, and comparison tables as *discussed*. Deployment modes were *hybrid*, *pure*, *both*, *not applicable*, and *unclear*. Hybrid labels required classical and PQC components in the same mechanism. We kept predecessor and standardized names distinct, including Kyber from ML-KEM and Dilithium from ML-DSA.

Human annotation. We drew a repository-aware, disproportionate stratified sample of 1,000 commits following software-repository sampling and GitHub-mining guidance [39], [40]. Sampling balanced scheme strata, merged-PR and default-branch evidence, and creation periods, with a cap of 20 records per repository. Rare schemes were oversampled; sample proportions were not used to estimate prevalence. We allocated 100 records to the pilot, 200 to development, and 700 to held-out evaluation. Three authors independently labelled the pilot with the same codebook. All had at least seven years of software-engineering experience; one had three years of quantum-computing experience and two had one year each. We measured relevance agreement with Fleiss’ κ , pairwise Cohen’s κ , and Krippendorff’s α . For scheme sets, we measured exact agreement and mean Jaccard overlap. Status and mode agreement used Fleiss’ κ over commit–scheme pairs named by all three annotators.

Model selection. The eight models in Section II-D received the same initial prompt, codebook, and evidence, including repository descriptions. We withheld record identifiers, corpus assignments, sampling strata, and keyword matches. Outputs followed a schema with one status and mode per scheme. We scored 196 development records after excluding three oversized patches and one encoding failure. Invalid responses and requests exceeding context limits counted as errors.

We measured relevance with Cohen’s κ and positive-class F1, and scheme identification with micro-F1 over commit–scheme pairs. Status and mode κ used schemes named by both the model and human annotator, with each commit weighted equally. We estimated 95% confidence intervals from 1,000 record-level bootstrap resamples with a fixed seed. Model comparisons used paired metric differences over the same resamples, with Holm correction. For sensitivity to scheme-stratum composition, we also calculated record weights

$$w_i = \frac{N_{h(i)}}{n_{h(i)}}, \quad \sum_h N_h = 73,076, \quad (1)$$

where $h(i)$ is record i ’s primary scheme stratum, N_h counts commits in that stratum of the original 73,076-commit sampling frame before subsequent filtering, and n_h counts scored development records. These stratum weights do not account for repository caps or coverage constraints and are not exact inverse-probability weights. We applied them to relevance, status, and mode κ , adoption-evidence Jaccard overlap, and adoption-evidence micro-F1. Weighted comparisons used within-stratum bootstrap resampling. Together with scheme micro-F1, these formed the six ranking metrics M . We ranked models within each metric, averaging tied ranks, and calcu-

¹<https://openrouter.ai>

lated

$$\bar{r}_m = \frac{1}{|M|} \sum_{k \in M} \text{rank}_k(m). \quad (2)$$

The three models with the lowest $\bar{r}_m$ advanced.

Prompt revision and alternative annotator test. We revised the prompt using development-set disagreements, emphasizing patch-based decisions and the distinction between implementation, maintenance, and discussion. We specified pure deployment as the default for standalone implementations, required the output to include every changed scheme, and classified packaging and build changes to scheme code as *maintenance*, consistent with the codebook.

We evaluated the revised prompt on the pilot using the alternative annotator test [41]. For each commit i , we compared model f and annotator j against the remaining annotators K_j ,

$$S_{ij}^f = \frac{1}{|K_j|} \sum_{k \in K_j} \text{sim}(f(x_i), h_k(x_i)), \quad (3)$$

$$S_{ij}^h = \frac{1}{|K_j|} \sum_{k \in K_j} \text{sim}(h_j(x_i), h_k(x_i)), \quad (4)$$

where sim was exact match for relevance, Jaccard overlap for scheme and adoption-evidence sets, and the agreement share for status and mode over jointly named schemes. We defined model wins as $W_{ij}^f = \mathbb{1}[S_{ij}^f \geq S_{ij}^h]$ and annotator wins as $W_{ij}^h = \mathbb{1}[S_{ij}^h \geq S_{ij}^f]$, counting ties for both. Their means over commits were ρ_j^f and ρ_j^h . We tested $H_{0j}: \rho_j^h - \rho_j^f \geq \epsilon$ with one-sided paired t -tests on $W_{ij}^h - W_{ij}^f$. We applied the Benjamini–Yekutieli correction [42] to the three annotator comparisons, using $q = 0.05$ divided across the three candidate models by Bonferroni correction [43]. Using $q = 0.05$ per model gave the same outcomes. A field passed when the share ω of rejected annotator-specific null hypotheses reached 0.5. We also calculated mean advantage probability $\bar{\rho} = \frac{1}{3} \sum_j \rho_j^f$. We fixed the skilled-annotator margin at $\epsilon = 0.15$ before evaluation and tested 0.10 and 0.20 for sensitivity.

Corpus classification. Filtering (Section II-B) left 47,729 commits in 163 repositories. Excluding human-annotated records left 46,964 commits in 147 repositories for automated classification. We used the passing model with the strongest agreement, submitting 95 OpenRouter Batch API batches of at most 1,000 commits through Google Vertex. The original 700-record held-out set remained separate from model and prompt selection.

2) *Result: Human agreement.* Relevance agreement reached Fleiss’ $\kappa = 0.819$, with unanimity on 88 of 100 pilot commits and pairwise Cohen’s κ of 0.756–0.857 (Table I). All three annotators assigned identical scheme sets to 73.0% of commits; mean pairwise Jaccard overlap was 0.864. Among eleven schemes named on at least five commits, per-scheme κ ranged from 0.670 for SPHINCS+ to 0.961 for XMSS. For the 103 commit–scheme pairs named by all annotators, status κ was 0.900 and mode κ was 0.725. Pairwise mode κ values were 0.607, 0.839, and 0.711.

TABLE I
HUMAN AGREEMENT ON 100 PILOT COMMITS. SCHEME-SET α USES JACCARD DISTANCE. STATUS AND MODE COVER SCHEMES NAMED BY ALL THREE ANNOTATORS.

Field	Fleiss κ	α	Pairwise agr.	Unanimous
PQC relevance	0.819	0.820	92.0%	88.0%
Scheme set	n/a	0.841	80.3%	73.0%
Adoption status	0.900	0.900	94.8%	92.2%
Deployment mode	0.725	0.726	90.0%	85.4%

Model selection. Relevance F1 ranged from 0.841 to 0.940, with the top three models within 0.016 (Table II). Gemini 3.8 Flash reached status $\kappa = 0.832$ (95% bootstrap interval 0.717–0.919) and mode $\kappa = 0.762$ (0.648–0.861), compared with 0.498–0.766 and 0.488–0.700, respectively, for the other models. After Holm correction, Gemini exceeded Mistral, Kimi, Qwen, and GLM on status κ , and Claude, Muse, Mistral, Kimi, and Qwen on mode κ . Its relevance advantage was significant against Mistral and Qwen, which failed on 24 and 23 of the 196 records, mainly after exhausting output limits. Across the six ranking metrics (Eq. (2)), Gemini and Muse tied at 2.17, followed by GPT-5.4 mini at 2.33; the remaining models scored at least 4.33. Gemini, Muse, and GPT-5.4 mini advanced.

TABLE II
MODEL SELECTION ON 196 DEVELOPMENT RECORDS USING THE INITIAL PROMPT. COHEN’S κ IS RECORD-WEIGHTED; FAILURES COUNT AS ERRORS. COLUMN BESTS ARE BOLD.

Model	PQC F1	Scheme F1	Status κ	Mode κ
Gemini 3.8 Flash	0.936	0.860	0.832	0.762
Muse Spark 1.3	0.940	0.870	0.766	0.547
GPT-5.4 mini	0.924	0.875	0.713	0.667
Claude Haiku 5.5	0.915	0.734	0.703	0.591
GLM-5.3	0.935	0.862	0.666	0.700
Kimi K2.7 Code	0.908	0.829	0.624	0.488
Qwen3.8-Flash	0.856	0.734	0.557	0.511
Mistral Medium 3.5	0.841	0.676	0.498	0.506

Prompt revision and alternative annotator test. On the pilot, Gemini’s mean κ was 0.830 for relevance, 0.828 for status, and 0.787 for mode, with scheme micro-F1 of 0.827 and adoption-evidence Jaccard overlap of 0.892. Muse’s corresponding κ values were 0.830, 0.810, and 0.738; GPT-5.4 mini’s were 0.789, 0.630, and 0.620. After Holm correction, GPT-5.4 mini trailed Gemini by 0.198 on status κ , 0.167 on mode κ , 0.058 on adoption-evidence Jaccard overlap, and 0.031 on scheme micro-F1. Gemini and Muse did not differ significantly on any unweighted metric.

At $\epsilon = 0.15$, Gemini and Muse passed all five fields with $\omega = 1.00$ (Table III), using the alignment scores in Eqs. (3) and (4). GPT-5.4 mini passed relevance, schemes, and adoption evidence but failed status and mode ($\omega = 0$). Results were unchanged at $\epsilon = 0.20$. At $\epsilon = 0.10$, Gemini and Muse passed relevance, schemes, and adoption evidence; GPT-5.4 mini passed only relevance. We selected Gemini 3.8 Flash with the revised prompt. It had the highest $\bar{\rho}$ for status, mode,

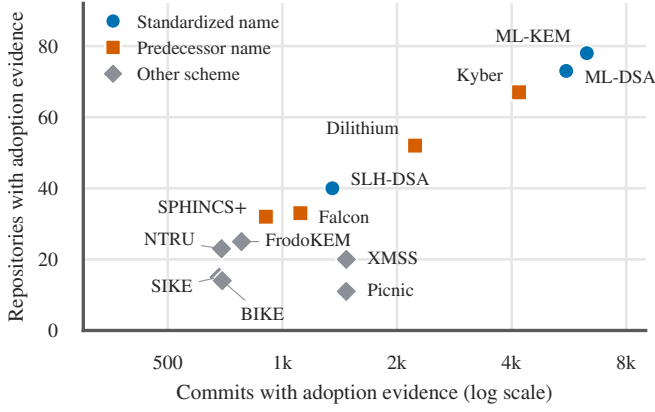


Fig. 2. Adoption breadth and commit activity for the 13 schemes with the most adoption-evidence commits (*implemented* or *maintenance*).

TABLE III
ALTERNATIVE ANNOTATOR TEST AT $\epsilon = 0.15$. CELLS REPORT MEAN ADVANTAGE PROBABILITY $\bar{p}$ AND WINNING RATE ω (PASS AT $\omega \geq 0.5$). ROW-BEST $\bar{p}$ VALUES ARE BOLD.

Field	Gemini 3.8 Flash	Muse Spark 1.3	GPT-5.4 mini
PQC relevance	0.980 / 1.00	0.980 / 1.00	0.960 / 1.00
Scheme set	0.970 / 1.00	0.953 / 1.00	0.930 / 1.00
Adoption status	0.950 / 1.00	0.933 / 1.00	0.833 / 0.00
Deployment mode	0.950 / 1.00	0.939 / 1.00	0.889 / 0.00
Adoption evidence	0.970 / 1.00	0.940 / 1.00	0.910 / 1.00

and adoption evidence, and higher stratum-weighted mode κ than Muse (Eq. (1)), at 0.870 versus 0.766 (Holm-adjusted $p = 0.016$).

Held-out evaluation. We evaluated the frozen classifier once on 690 held-out records. Humans labelled all 700, but 10 oversized or non-UTF-8 commits were never submitted to any model. Failed responses counted as errors. Gemini reached F1 0.93 for PQC-related commits (precision 0.97, recall 0.89) and 0.85 for non-PQC commits, giving macro-F1 0.89. Scheme micro-F1 was 0.84, with precision 0.93 and recall 0.77. Status and mode macro-F1 were 0.70 and 0.68. At the commit-scheme level, *rejected or reverted* had only 2 examples; *unclear* deployment had 46 and recall 0.04. Recall for *deprecated or removed* was 0.58, while frequent classes reached F1 of 0.80–0.89.

Corpus classification. Gemini labelled 46,051 of 46,964 commits (98.1%) for \$422.96. The 913 unlabelled records comprised 363 over-context requests, 245 invalid answers (200 at the output limit), 218 oversized patches, 82 non-UTF-8 patches, and 5 reviewed failures. Of the labelled commits, 25,213 (54.8%) were PQC-related and 20,838 were not. PQC-related commits occurred in 144 of 147 repositories; the model assigned high confidence to 96.0% of them.

3) *Observation:* Among PQC-related commits, 24,478 (97.1%) implemented or maintained a scheme, covering 135 repositories, 107 schemes, and 834 repository-scheme adoption pairs. Lattice-based standards had the broadest adoption

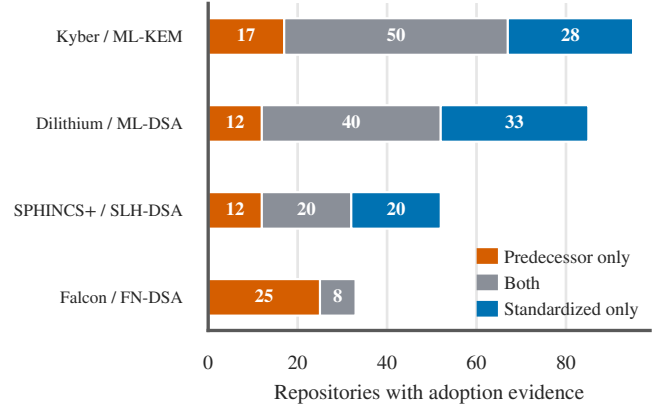


Fig. 3. Predecessor-standard coexistence. Repository counts by adoption of predecessor names, standardized names, or both.

(Figure 2). ML-KEM appeared in 78 repositories, ML-DSA in 73, Kyber in 67, and Dilithium in 52.

Predecessor and standardized names coexisted across all four renamed lineages (Figure 3). Of 95 repositories with Kyber or ML-KEM evidence, 50 contained both and 17 only Kyber. Dilithium and ML-DSA appeared in 85 repositories, including 40 with both; SPHINCS+ and SLH-DSA appeared in 52, including 20 with both. All 8 repositories with FN-DSA also contained Falcon, which appeared in 33. In each lineage, standardized names coexisted with predecessors at least as often as they appeared alone.

Commit volume did not imply broad adoption (Figure 2). Picnic had 1,471 adoption-evidence commits in 11 repositories, whereas 1,353 SLH-DSA commits spanned 40. Maintenance dominated the 34,937 scheme labels at 90.6%, followed by *implemented* (3.8%), *discussed* (3.5%), *deprecated or removed* (1.6%), and *rejected or reverted* and *planned* (0.2% each). RQ3 used the 1,331 *implemented* labels to identify first-adoption events.

Of 32,994 adoption labels, 90.1% were pure, 4.6% hybrid, 4.3% both, and 1.0% unclear. The 2,932 hybrid or both labels spanned 51 repositories. X25519 was the leading classical partner in 908 commits, followed by P-256 in 416, P-384 in 352, Ed25519 in 348, and P-521 in 190. Hybrid key exchange centered on Curve25519 and NIST curves; hybrid signatures centered on Ed25519.

RQ1 Summary. ML-KEM (78 repositories) and ML-DSA (73) had the broadest adoption, while predecessor schemes persisted in all four renamed lineages. Pure-PQC deployment dominated (90.1% of adoption labels); hybrid or mixed modes accounted for 8.9% across 51 repositories. Maintenance comprised 90.6% of scheme labels, indicating sustained work on existing implementations.

B. RQ2 Recurring Technical Challenges

RQ2 examined recurring technical challenges in PQC implementation, integration, and migration. We constructed an

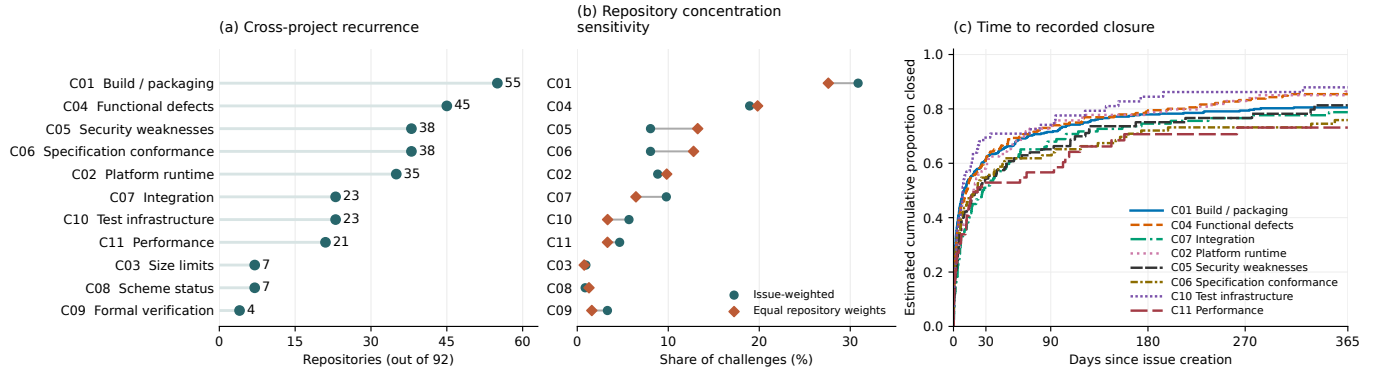


Fig. 4. Recurrence and closure of PQC technical challenges. (a) Number of repositories reporting each category. (b) Category shares when weighting issues or repositories equally. (c) Kaplan–Meier estimates of the proportion closed by each day for categories found in at least 10 repositories, with open issues censored at collection.

inductive taxonomy and validated a classifier against adjudicated human labels before estimating category frequencies.

1) *Approach: Unit and eligibility.* We analyzed issues with their comments. For issues closed by a PR, we included the PR text, comments, and commit messages as supporting evidence, without counting the PR separately. Coders and the classifier received the same fixed rendering of issue metadata, opening text, chronological comments, and linked PR evidence.

We retained issues that were both PQC-related and reported a concrete technical problem or obstacle. We excluded feature requests, usage questions without a reported problem, announcements, routine maintenance, and incidental keyword matches. Three researchers independently screened the same 100 pilot issues; each of the 200 development and 300 screening held-out issues received one human label. We measured agreement on the pilot and evaluated the frozen screening classifier on all 300 held-out issues. Taxonomy coding rechecked eligibility, and only confirmed challenges entered the final category counts.

Taxonomy construction. Two researchers independently open-coded four rounds of initially screen-positive issues (50, 100, 100, and 100), following qualitative-analysis and coding guidance [44], [45]. Sampling used a fixed seed and previously uncoded issues, balancing repositories, creation years, schemes, ecosystems, and project roles. Coders assessed eligibility, described each eligible problem, and assigned an open code. From round 2, they also selected a primary category and optional sub-category or proposed a new category. We grouped related codes through thematic synthesis [46].

A third researcher chaired adjudication meetings to reconcile disagreements and revise category definitions. We computed Cohen’s κ before adjudication. Each issue received one primary category based on the problem that motivated its creation; sub-categories captured variation within that category. We stopped after two consecutive rounds added no primary category [47]. Taxonomy construction took approximately 18 hours.

Classifier validation. We supplied Gemini 3.8 Flash with the final codebook and eligibility rules, including category def-

initions, open codes, and tie-breakers. Its schema-constrained output recorded eligibility, a problem description, an open code, primary category and sub-category, an evidence quote, and confidence. We did not refine the prompt on labelled data.

We selected 316 initially screen-positive issues outside the discovery rounds, stratifying by corpus and creation year with at most 15 issues per repository. The target size followed Cochran’s formula for a proportion at 95% confidence and a $\pm 5\%$ margin, with finite-population correction for the 1,755 screen-positive issues. Both coders independently applied the frozen codebook. We required pre-adjudication primary-category $\kappa \geq 0.80$ on mutually confirmed issues, then evaluated the frozen classifier once against adjudicated labels.

Acceptance required primary-category macro-F1 ≥ 0.80 and F1 ≥ 0.70 for each category with at least 30 reference cases. We estimated 95% bootstrap intervals from 2,000 issue-level resamples. The category evaluation included human-confirmed challenges and counted model rejections as errors. A separate evaluation included all held-out issues, treating rejection as an additional class.

Counting and analysis. We combined adjudicated discovery and held-out labels with classifier labels for the remaining issues. Category shares included only confirmed challenges. We measured recurrence as the number of repositories per category and compared issue-weighted shares with shares giving each represented repository equal weight.

We estimated time from issue creation to recorded closure with Kaplan–Meier curves [48], censoring open issues at their collection timestamps. We plotted the cumulative proportion closed, $1 - \hat{S}(t)$, assuming non-informative censoring within categories. Closure did not establish successful resolution, and we did not reconstruct reopening histories. Figure 4c covers the eight categories present in at least 10 repositories, representing 1,202 issues (94.9%). The artifact provides all-category curves with pointwise 95% intervals and the validation and analysis scripts. These comparisons were descriptive and unadjusted, without between-category significance tests.

2) *Result:* Removing 26 bot-authored issues from the 5,713 collected issues left 5,687 candidates. We used 600 for screen-

TABLE IV
TAXONOMY OF RECURRING PQC TECHNICAL CHALLENGES ACROSS 1,267 CATEGORIZED ISSUES.

#	Primary category (code)	Operational definition	Issues, n (%)
1	Build, packaging, and dependency failures (C01) <i>Subcategories.</i> Platform builds; configuration builds; dependency and packaging; source defects	PQC code cannot be compiled, linked, installed, or resolved as a dependency, or the installed package is incomplete	391 (30.9)
2	Incorrect results and functional defects (C04)	The implementation gives wrong results, hangs, or misses a case it should support, with no dispute about the standard	240 (18.9)
3	Protocol and ecosystem integration (C07)	A scheme cannot be used in an existing protocol, product, or deployment because identifiers, library versions, or integration pieces are missing or incompatible	124 (9.8)
4	Platform-specific runtime faults (C02) <i>Subcategories.</i> Memory and stack limits; compiler and toolchain faults; ABI and alignment faults	Code builds but crashes or gives wrong results on one platform, compiler, runtime, or memory configuration	112 (8.8)
5	Security weaknesses (C05) <i>Subcategories.</i> Side channels; key handling; input validation; misuse risk	The implementation or interface leaks secrets, accepts invalid input, mishandles keys, or makes unsafe use easy	102 (8.1)
6	Specification conformance and interoperability (C06) <i>Subcategories.</i> Draft versions; specification deviation; reference tracking and test vectors; representation differences; specification text errors	Behaviour, results, or encodings differ from the standard, another scheme version, or the reference implementation	102 (8.1)
7	Test infrastructure and test vectors (C10)	Tests cannot run or cannot be trusted because data, fixtures, or infrastructure are missing or broken	72 (5.7)
8	Performance and efficiency (C11)	A scheme, implementation, or test is too slow, or a benchmark misreports speed or memory	59 (4.7)
9	Formal verification and proof obstacles (C09)	A formal proof is blocked, or a formal statement or model is wrong	42 (3.3)
10	Resource and format size limits (C03)	Keys, signatures, or ciphertexts exceed a protocol, format, or interface size limit	12 (0.9)
11	Scheme-level breaks and algorithm status (C08)	A scheme loses security or upstream support, affecting code that depends on it	11 (0.9)

ing annotation and screened the remaining 5,087 with Gemini. The model selected 1,573 issues; combined with 182 human-confirmed issues, these formed the 1,755 screen-positive pool. Taxonomy discovery used 350 issues and held-out evaluation used 316, leaving 1,089 for automated classification.

Screening validation. On the shared pilot, Krippendorff’s nominal α was 0.925 for PQC relevance, 0.918 for a technical challenge, and 0.903 for eligibility. Across all 300 screening held-out issues, eligibility precision was 0.818, recall 0.889, F1 0.852, and Cohen’s κ 0.794. Recall fell below the planned 0.90 target. The artifact reports the full screening evaluation and evidence-quote checks.

Taxonomy. Round 1 produced 10 categories; round 2 added performance and efficiency. Rounds 3 and 4 added none, meeting the stopping rule, and held-out coding identified no additional category. The final taxonomy contained 11 primary categories and 16 sub-categories (Table IV).

On mutually confirmed issues, pre-adjudication primary-category κ was 0.920 in round 2, 0.904 in round 3, 0.891 in round 4, and 0.857 on the held-out set (95% interval 0.805–0.904), exceeding the 0.80 requirement. Eligibility agreement was lower ($\kappa = 0.457$ –0.797), reflecting disagreements over whether issues reported technical problems. Because round-1 coders used different category names, we compared their issue groupings with the adjusted Rand index (0.54).

Classifier validation. On the 235 human-confirmed held-out challenges, Gemini achieved macro-F1 0.874 (95% inter-

val 0.796–0.914), accuracy 0.800, and $\kappa = 0.771$. It met the macro-F1 threshold and the per-category requirement for C01 (F1 0.847) and C04 (0.833), the only categories with at least 30 cases.

Eligibility errors predominated. The model rejected 30 of the 235 confirmed challenges and accepted 17 of the 81 human-rejected issues, giving eligibility $\kappa = 0.629$. Among the 205 challenges confirmed by both humans and the model, primary-category accuracy was 91.7%; sub-category accuracy was 90.4% where the primary category was correct. Across all 316 held-out issues, including rejection as a class, macro-F1 was 0.841 and κ was 0.761.

Category counts. Of the 1,089 remaining issues, Gemini categorized 755, rejected 333, and returned no label for one. Taxonomy classification cost \$9.55, comprising \$2.07 for held-out evaluation and \$7.48 for the remaining issues. The combined human- and model-labelled corpus contained 1,267 challenges, of which 1,072 were closed and 195 remained open at collection.

3) *Observation:* Build, packaging, and dependency failures were the largest category, with 391 issues (30.9%), followed by incorrect results and functional defects with 240 (18.9%). These categories appeared in 55 and 45 of the 92 represented repositories (Figure 4a) and remained the largest under equal repository weights, at 27.6% and 19.8% (Figure 4b). Security weaknesses (C05) and specification conformance and interoperability (C06) each appeared in 38 repositories, compared

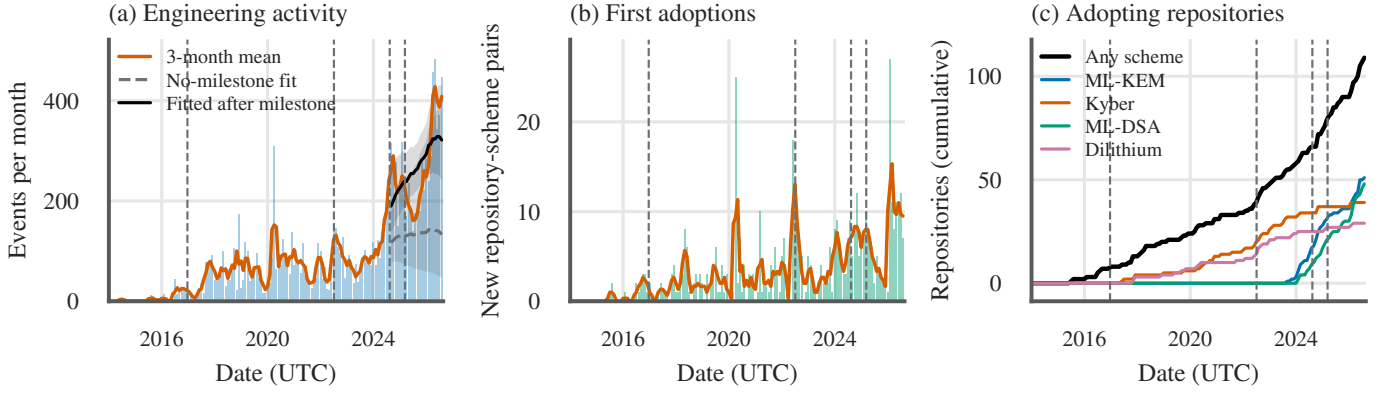


Fig. 5. Temporal evolution. (a) Monthly adoption-evidence events, with the fit after the 2024 finalization and its no-milestone counterfactual (95% bands). (b) New repository-scheme pairs per month, dated at the first *implemented* event. (c) Cumulative repositories with a first adoption, overall and for four schemes. Dashed vertical lines mark the four NIST milestones.

TABLE V
ISSUES STILL OPEN AND UNDER OBSERVATION BY DAY SINCE CREATION
FOR FIGURE 4C.

Category	Days since issue creation				
	0	90	180	270	365
C01 Build / packaging	391	106	81	72	62
C04 Functional defects	240	55	40	31	25
C07 Integration	124	34	26	22	18
C02 Platform runtime	112	26	22	16	10
C05 Security weaknesses	102	30	17	15	12
C06 Specification conformance	102	33	24	22	18
C10 Test infrastructure	72	15	9	8	7
C11 Performance	59	23	13	11	9

with 23 for integration despite its higher issue count. Formal-verification obstacles were concentrated in four repositories.

Estimated median time to recorded closure was 14.8 days overall, 9.9 days for build failures, and 26.9 days for integration challenges (Figure 4c; observation counts in Table V). Performance had the largest open share, at 27.1%. These unadjusted differences also reflected project composition and issue ages.

Classifier rejections were concentrated in build and integration challenges, accounting for 12 and 7 of the 30 rejected human-confirmed cases. This pattern indicates possible undercounting in the 1,089 automatically classified issues. Nine categories had fewer than 30 held-out cases and were not subject to the per-category F1 requirement; their scores were reported with the corresponding support.

RQ2 Summary. Across 11 categories, build, packaging, and dependency failures (30.9% of issues) and functional defects (18.9%) were the most common challenges. They recurred in 55 and 45 repositories and remained dominant under equal repository weights. Integration challenges were less frequent but had a longer estimated median time to recorded closure than build failures (26.9 versus 9.9 days).

C. RQ3 Temporal Evolution

RQ3 examined changes in PQC engineering activity and first adoption of individual schemes around NIST milestones.

1) *Approach:* We analyzed two outcomes from RQ1’s PQC-related commits. *Engineering activity* comprised commits with at least one *implemented* or *maintenance* label. We grouped commits linked to the same merged PR into one event and counted unlinked commits individually. *First adoption* was the earliest *implemented* event for each repository-scheme pair. We retained predecessor and standardized names as separate schemes and repeated the analysis after merging each lineage within a repository under its earliest adoption date.

We dated events by PR merge time or, without a merged PR, commit time. Commit and author dates provided sensitivity checks. Monthly series ended in August 2026, the last complete month before collection. They used the same 24,478 commits as RQ1; excluding events after August 2026 left 13,429 engineering events from 23,513 commits in 134 repositories, plus 488 first-adoption pairs in 109 repositories. Of RQ1’s 834 pairs with adoption evidence, 341 had only *maintenance* labels and 5 first adoptions occurred after August 2026.

We assessed four NIST milestones (Table VI), covering the 2016 call, July 2022 selection, August 2024 finalization of FIPS 203, 204, and 205, and March 2025 HQC selection. For milestones meeting the window requirements below, we fitted segmented negative-binomial models [49], [50] to monthly event counts,

$$y_t \sim \text{NB}(\mu_t, \alpha),$$

$$\log \mu_t = \log e_t + \beta_0 + \beta_1 t$$

$$+ \beta_2 \text{post}_t + \beta_3(t - t_0) \text{post}_t, \quad (5)$$

where y_t is the count in month t , e_t is exposure, t_0 is the milestone month, and post_t indicates subsequent months. We reported $\exp(\beta_2)$ as the immediate level ratio and $\exp(\beta_3)$ as the change in monthly growth. Baseline exposure counted repositories created by month-end and last pushed at or after month-start. To account for temporal dependence in repository

data [51], [52], we used Newey–West standard errors [53] with a three-month bandwidth. We compared negative-binomial and Poisson fits using dispersion [50] and AIC [54], and tested Pearson residuals with Ljung–Box tests [55].

Analysis windows extended up to 36 months on each side, omitted the milestone month, and stopped at another primary milestone at least 12 months away. Estimation required at least 12 months and 100 events per side, or 30 events for first adoption. We excluded milestones within 12 months of a preceding milestone.

Following guidance on pre-specified model choices [56], we evaluated intervention starts at 0, 3, and 6 months. Shifted starts assigned the intervening months to the pre-period; excluded starts omitted those months while retaining the original pre-trend. We assessed autocorrelation sensitivity with Newey–West bandwidths of 6 and 12 months, generalized estimating equations with AR(1) working correlation, and a negative-binomial model with a lagged outcome.

Further checks used alternative dates, capped repository contributions at 20 events per month, and aggregated events quarterly. Alternative exposure measures were repositories with a selected commit that month, all selected commits, and, for first adoption, repositories without an earlier *implemented* commit. These checks addressed GitHub activity biases [57]. We interpreted estimates as temporal associations, not causal effects [58].

2) *Result:* Engineering events increased from 172 in 2016 to 1,005 in 2022 and 2,915 in the first eight months of 2026, while mean repository exposure rose from 15 to 118. Monthly activity per repository fell to 0.73 events in 2023, rose to 1.58 in 2024, and reached 3.08 in 2026 (Figure 5a). First adoptions numbered 63 in 2022, 31 in 2023, 79 in 2024, 60 in 2025, and 87 in the first eight months of 2026. Cumulatively, 109 repositories recorded at least one first adoption (Figure 5b,c).

TABLE VI
NEGATIVE-BINOMIAL MILESTONE ESTIMATES USING INTEGRATION DATES. INCIDENCE RATE RATIOS (95% INTERVALS) DESCRIBE IMMEDIATE LEVEL CHANGES AND CHANGES IN MONTHLY GROWTH.

Milestone	Outcome	Level ratio	Growth ratio
2016 call (Dec 20)	Engineering	1.17 (0.41–3.31)	0.92 (0.86–0.98)
	First adoption	not estimable	
2022 selection (Jul 5)	Engineering	1.17 (0.63–2.18)	1.03 (0.99–1.07)
	First adoption	0.69 (0.39–1.22)	1.01 (0.95–1.08)
2024 FIPS (Aug 13)	Engineering	1.59 (1.09–2.31)	1.02 (0.98–1.06)
	First adoption	1.13 (0.63–2.03)	1.01 (0.96–1.06)
2025 HQC (Mar 11)	not estimated (within 12 months of 2024 FIPS)		

Negative-binomial models fitted the counts better than Poisson models, with dispersion statistics of 1.1–1.6 versus 15–27 and AIC reductions of several hundred. At the 2024 finalization, AIC was 528 versus 1,258. Most fits had no detected residual autocorrelation; the 2024 fit was the exception (Ljung–Box $p = 0.002$).

Table VI reports the baseline estimates. The 2016 call had only 192 pre-period engineering events and a monthly growth ratio of 0.92. At the 2022 selection, the engineering level ratio

was 1.17 (95% interval 0.63–2.18). The first-adoption level ratio was 0.69 (0.39–1.22) at lag 0, with intervals below 1 at lags of 3 months (0.30, 0.13–0.72) and 6 months (0.41, 0.17–0.97). The 2024 finalization was associated with increased engineering activity (1.59, 1.09–2.31), but no detectable first-adoption change (1.13, 0.63–2.03). The March 2025 selection was not estimated because it fell within 12 months of the finalization.

The 2024 engineering increase persisted under several exposure and date specifications. Commit and author dates gave ratios of 1.58 and 1.59; capping repository contributions gave 1.38 (1.04–1.84). Active-repository exposure and quarterly aggregation gave 1.53 (1.15–2.03) and 1.56 (1.10–2.22). The Poisson estimate with robust errors was 1.41 (0.90–2.23), whereas exposure based on all selected commits reduced the ratio to 1.17 (0.96–1.41).

Timing and autocorrelation specifications changed the inference. Shifting the interruption by 3 or 6 months gave 0.82 (0.47–1.44) and 0.67 (0.45–1.01). Excluding those phase-in months gave 1.11 (0.67–1.82) and 0.80 (0.54–1.20). Newey–West bandwidths of 6 and 12 months retained the 1.59 estimate, with intervals of 1.11–2.27 and 1.14–2.21. Directly modeling autocorrelation gave 1.47 (0.86–2.54) under AR(1) and 1.09 (0.80–1.47) with a lagged outcome. Both intervals included 1; the lagged-outcome model also removed detected residual autocorrelation (Ljung–Box $p = 0.63$).

First-adoption estimates at the 2024 finalization remained compatible with no change under commit dates (1.19, 0.70–2.02), at-risk repository exposure (0.78, 0.39–1.57), and quarterly aggregation (1.00, 0.48–2.08). Merging predecessor and standardized names reduced first-adoption pairs from 488 to 436, with similar estimates at the 2022 selection (0.76, 0.45–1.29) and 2024 finalization (1.10, 0.63–1.95).

3) *Observation:* Engineering activity and first adoption followed different trajectories. Activity per repository approximately doubled between 2023 and 2024 and again by 2026. The immediate 2024 increase depended on exposure, timing, and autocorrelation treatment; it was not statistically distinguishable under AR(1), lagged-outcome, or delayed-interruption specifications. We found no statistically distinguishable increase in first-adoption rates at the estimable milestones. The lower post-2022 rate depended on the lag. The results were consistent with more intensive work on existing adoption, but did not exclude changes in new scheme adoption.

Scheme names shifted toward standardized variants (Tables VII and VIII). Standardized names appeared in only one of 4,406 events before the 2022 selection, in 19% between selection and finalization, and in 82%–84% afterwards. Predecessor names accounted for 66% between selection and finalization, then 25% and 11% in subsequent eras. Hybrid or both labels rose from 4% to about 11% after selection. Among repositories with both names, predecessor-first sequences predominated in every lineage. For Kyber and ML-KEM, 37 of 50 repositories used Kyber first and 6 used ML-KEM first; 25 repositories still showed only Falcon. These patterns measured names in the source and did not distinguish

algorithm migration from identifier renaming.

RQ3 Summary. Engineering activity increased around the 2024 finalization, but first-adoption rates showed no statistically distinguishable increase at the estimable milestones. The 2024 activity estimate (ratio 1.59, 95% interval 1.09–2.31) was sensitive to exposure, timing, and autocorrelation treatment. Standardized names reached 82%–84% of events after finalization, while predecessor names persisted.

IV. DISCUSSION

PQC adoption involved sustained work beyond initial implementation. Maintenance accounted for 90.6% of scheme labels, compared with 3.8% for new implementations. Recurring build and functional problems in RQ2 identify dependency compatibility and regression testing as continuing maintenance concerns. These proportions describe changes, not effort or cost.

Adoption breadth and commit volume captured different patterns (Figure 2). The temporal analysis showed increased engineering activity around the 2024 finalization without a statistically distinguishable increase in first-adoption rates (Figure 5). The activity estimate varied with exposure, timing, and autocorrelation treatment and did not establish causality. Higher commit volume therefore did not imply wider adoption.

Across projects, build, packaging, and dependency failures and functional defects recurred in 55 and 45 repositories and remained the largest categories under equal repository weights (Figure 4a–b). The taxonomy (Table IV) identifies targets for reusable tests of clean installation, output correctness, and conformance to versioned test vectors, although we did not evaluate these interventions.

Security weaknesses and specification conformance and interoperability each appeared in 38 repositories, compared with 23 for integration despite its higher issue count. Frequency and project coverage thus captured different patterns, neither establishing severity. Size limits (12 issues) and scheme-level breaks (11) remained relevant despite their low counts.

Integration challenges had a longer estimated median time to recorded closure than build failures, 26.9 versus 9.9 days (Figure 4c). These unadjusted estimates measured recorded closure, not confirmed resolution or repair effort.

Standardized names predominated after finalization, but predecessor names persisted in every renamed lineage and coexisted within repositories (Figure 3). A standardized-name-only inventory would omit predecessor activity. Recording both the observed identifier and its lineage preserves distinctions such as Kyber versus ML-KEM for conformance and interoperability analysis, without treating names as proof of compatibility. Deployment configurations also varied. Hybrid or mixed deployment appeared in 51 repositories despite the dominance of pure deployment. Classical partners included X25519 and NIST curves for key exchange and Ed25519 for signatures. These combinations provide concrete interoperability test targets, but their frequency does not establish security or suitability.

V. RELATED WORK

PQC adoption is observable at different layers of the software stack. At the application layer, binary analysis of more than 4,000 Android apps found no adoption of standardized PQC [7]. At the protocol layer, TLS measurements show deployment concentrated on hybrid X25519MLKEM768 and shaped by major infrastructure providers [8], [9]. These findings concern different populations and distinguish support shipped in applications from protocols negotiated in practice. Static analysis brings this assessment to source code by identifying PQC API usage in Python, Java, and Go repositories [21]. Our study extends this repository-level perspective from the presence of PQC support to the changes and discussions through which developers implement and maintain it.

Complementary migration research addresses how to move existing software toward PQC. Readiness and crypto-agility frameworks make cryptographic dependencies and migration needs explicit through assessment and asset inventories [59]–[61]. Migration reviews organize the transition process [62], while toolchains and LLM-assisted code-fragment migration address the changes needed to carry it out [26], [63]. Android benchmarks complement these approaches by evaluating PQC implementations on target devices [64]. This work supports planning and implementing migration. We examine how that work unfolds in open-source projects, identifying recurring technical challenges and relating adoption histories to NIST milestones.

VI. THREATS TO VALIDITY

Construct validity. Keyword matches can miss PQC work or retrieve unrelated changes. Multiple discovery channels broaden coverage, and we assess relevance from patches and threads rather than keywords alone. Adoption labels represent source-level implementation and maintenance, not deployed software. Recorded issue closure likewise does not establish problem resolution.

Internal validity and reliability. Coding and classification errors can affect the findings. We measure agreement among three RQ1 pilot annotators and two RQ2 coders, with a third researcher adjudicating RQ2 disagreements. Frozen codebooks and separate held-out evaluation constrain classification decisions. The artifact preserves prompts, labels, raw outputs, and failures.

Conclusion validity. Rare categories yield less stable estimates. We report support and uncertainty, correct multiple comparisons, and do not infer prevalence from enriched validation samples. Temporal analyses adjust for repository counts and test alternative exposures, lags, and autocorrelation models. They establish associations with NIST milestones, not causal effects.

External validity. Findings describe the sampled public GitHub repositories. Private projects, other platforms, and projects missed by curated lists or search may differ. Corpus B includes only keyword-matched artifacts, so its coverage does not represent all PQC activity in general-purpose software.

TABLE VII

EVENTS BY ERA. EVENT SHARES INDICATE HYBRID/BOTH DEPLOYMENT, STANDARDIZED NAMES (ML-KEM, ML-DSA, SLH-DSA, FN-DSA), AND PREDECESSOR NAMES (KYBER, DILITHIUM, SPHINCS+, FALCON), RESPECTIVELY.

Era	Events	Repos	Hybrid	Standard	Prior
Before 2016 call	213	8	0.9%	0.0%	0.0%
2016 call to 2022 sel.	4,193	49	3.9%	0.0%	34.2%
2022 sel. to 2024 FIPS	2,444	72	11.5%	19.3%	66.3%
2024 FIPS to 2025 HQC	1,815	67	10.9%	81.8%	25.5%
After 2025 HQC	4,764	98	10.6%	83.9%	10.8%

VII. CONCLUSION

We studied PQC adoption across 179 GitHub repositories to characterize scheme implementation and maintenance, recurring technical challenges, and development around NIST milestones. Maintenance accounts for 90.6% of scheme labels, showing that the observed work centers on sustaining existing implementations. Standardized scheme names become dominant after finalization, but predecessor names persist across all renamed lineages.

Our taxonomy identifies 11 recurring technical challenges. Build, packaging, and dependency failures and functional defects lead both in issue frequency and cross-project recurrence, remaining the largest categories when repositories receive equal weight. The temporal analysis distinguishes activity from diffusion. Engineering activity rises around the 2024 finalization, while first adoption by new repository–scheme pairs does not measurably accelerate at the studied milestones. This association is consistent with intensified work among existing adopters, not evidence that standardization caused wider adoption.

These findings position PQC adoption as a continuing software-maintenance effort. Migration plans need to account for build reliability and integration testing alongside algorithm implementation, and adoption inventories must recognize both standardized and predecessor names. The taxonomy provides a basis for organizing these challenges across projects. Our released data, codebooks, annotations, and analysis pipeline support replication and longitudinal study as PQC implementations evolve.

DATA AVAILABILITY STATEMENT

We release an anonymized replication package [65]. It contains the analysis pipeline, the dataset records we analyze, and the prompts, model outputs, codebooks, and annotations behind both analyses.

ACKNOWLEDGMENT

Generative AI tools assisted with language editing, organization, and scripting. The authors directed the work and reviewed all design decisions, analyses, claims, and citations.

TABLE VIII

REPOSITORIES WITH ADOPTION EVIDENCE BY LINEAGE. PRIOR AND STD. DENOTE EXCLUSIVE NAME USE; 1ST COLUMNS COUNT EARLIER INTEGRATIONS AMONG BOTH. COUNTS MATCH RQ1, INCLUDING EVENTS AFTER AUGUST 2026.

Lineage	Prior	Both	Std.	Prior 1st	Std. 1st
Kyber / ML-KEM	17	50	28	37	6
Dilithium / ML-DSA	12	40	33	26	5
SPHINCS+ / SLH-DSA	12	20	20	15	3
Falcon / FN-DSA	25	8	0	4	1

REFERENCES

- [1] P. W. Shor, “Algorithms for quantum computation: discrete logarithms and factoring,” in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, ser. SFCS ’94. USA: IEEE Computer Society, 1994, p. 124–134. [Online]. Available: <https://doi.org/10.1109/SFCS.1994.3657700>
- [2] National Institute of Standards and Technology, “Submission requirements and evaluation criteria for the post-quantum cryptography standardization process,” Dec. 2016, accessed 2026-09-28. [Online]. Available: <https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf>
- [3] —, “NIST releases first 3 finalized post-quantum encryption standards,” Aug. 2024, accessed 2026-09-28. [Online]. Available: <https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards>
- [4] G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, H. Silberg, D. Smith-Tone, and N. Waller, “Status report on the fourth round of the NIST post-quantum cryptography standardization process,” National Institute of Standards and Technology, NIST Internal Report NIST IR 8545, Mar. 2025. [Online]. Available: <https://doi.org/10.6028/NIST.IR.8545>
- [5] M. Mosca, “Cybersecurity in an era with quantum computers: Will we be ready?” *IEEE Security and Privacy*, vol. 16, no. 5, pp. 38–41, Sep. 2018. [Online]. Available: <https://doi.org/10.1109/MSP.2018.3761723>
- [6] J. Blanco-Romero, F. A. Mendoza, C. G. Rubio, C. Campo, and D. D. Sánchez, “On the practical feasibility of harvest-now, decrypt-later attacks,” 2026. [Online]. Available: <https://arxiv.org/abs/2603.01091>
- [7] J. Strauss, K. Upadhyay, A. B. Siddique, I. Baggili, and U. Farooq, “Assessing and enhancing quantum readiness in mobile apps,” 2025. [Online]. Available: <https://arxiv.org/abs/2506.00790>
- [8] K. Loizou and E. Ghadafi, “Measuring post-quantum tls deployment across uk internet sectors,” 2026. [Online]. Available: <https://arxiv.org/abs/2608.02147>
- [9] N. Wickramasinghe, F. Li, S. Jha, and A. Shaghagh, “Mind the gap: Policy vs reality in post-quantum tls deployment,” 2026. [Online]. Available: <https://arxiv.org/abs/2607.29005>
- [10] Meta Engineering, “Post-quantum readiness for TLS at Meta,” May 2024, accessed 2026-09-28. [Online]. Available: <https://engineering.fb.com/2024/05/22/security/post-quantum-readiness-tls-pqr-meta/>
- [11] Meta, “Post-quantum cryptography migration at Meta: Framework, lessons, and takeaways,” Apr. 2026, accessed 2026-09-28. [Online]. Available: <https://engineering.fb.com/2026/04/16/security/post-quantum-cryptography-migration-at-meta-framework-lessons-and-takeaways/>
- [12] Apple Security Engineering and Architecture, “iMessage with PQ3: The new state of the art in quantum-secure messaging,” Feb. 2024, accessed 2026-09-28. [Online]. Available: <https://security.apple.com/blog/imessage-pq3/>
- [13] Amazon Web Services, “AWS post-quantum cryptography migration plan,” accessed 2026-09-28. [Online]. Available: <https://aws.amazon.com/blogs/security/aws-post-quantum-cryptography-migration-plan/>
- [14] Chromium Project, “Post-quantum auth roadmap,” accessed 2026-09-28. [Online]. Available: <https://www.chromium.org/Home/chromium-security/post-quantum-auth-roadmap/>

- [15] Microsoft .NET Team, "Post-quantum cryptography in .NET," 2025, accessed 2026-09-28. [Online]. Available: <https://devblogs.microsoft.com/dotnet/post-quantum-cryptography-in-dotnet/>
- [16] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in android applications," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 73–84. [Online]. Available: <https://doi.org/10.1145/2508859.2516693>
- [17] M. Hazhirpasand, M. Ghafari, and O. Nierstrasz, "Cryptoexplorer: An interactive web platform supporting secure use of cryptography apis," 2020. [Online]. Available: <https://arxiv.org/abs/2001.00773>
- [18] N. C. L. Ramalho, E. A. da Silva, H. A. de Souza, and M. L. Chaim, "Mining quantum software patterns in open-source projects," 2026. [Online]. Available: <https://arxiv.org/abs/2601.06281>
- [19] K. Upadhyay, V. Chhetri, A. Siddique, and U. Farooq, "Analyzing the evolution and maintenance of quantum software repositories," in *2025 IEEE International Conference on Quantum Software (QSW)*. IEEE, Jul. 2025, pp. 173–184. [Online]. Available: <http://dx.doi.org/10.1109/QSW67625.2025.00029>
- [20] K. Upadhyay, M. Fakorede, and U. Farooq, "Understanding bugs in quantum simulators: An empirical study," 2026. [Online]. Available: <https://arxiv.org/abs/2603.22789>
- [21] L. Mehl, "PQC-Readiness of Open-Source Software: An Empirical Study," in *20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), R. Feldt, M. Paasivaara, D. Mendez, S. Wagner, and M. M. Barón, Eds., vol. 394. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2026, pp. 53:1–53:14. [Online]. Available: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ESEM.2026.53>
- [22] PQClean contributors, "PQClean: Clean, portable, tested implementations of post-quantum cryptography," 2026, accessed 2026-09-08. [Online]. Available: <https://github.com/PQClean/PQClean>
- [23] "OpenSSL," accessed 2026-10-11. [Online]. Available: <https://github.com/openssl/openssl>
- [24] "Bouncy Castle open-source cryptographic APIs," accessed 2026-10-11. [Online]. Available: <https://www.bouncycastle.org>
- [25] D. M. German, B. Adams, and A. E. Hassan, "Continuously mining distributed version control systems: An empirical study of how Linux uses Git," *Empirical Software Engineering*, vol. 21, no. 1, pp. 260–299, Feb. 2016. [Online]. Available: <https://doi.org/10.1007/s10664-014-9356-2>
- [26] J. P. de Bonrosto, A. I. González-Tablas, and M. I. G. Vasco, "Empirical evaluation of large language models for migration of code fragments to post-quantum cryptography," 2026. [Online]. Available: <https://arxiv.org/abs/2606.07341>
- [27] Google, "Introducing Gemini 3.8 flash and 3.8 flash cyber," Sep. 2026, accessed 2026-10-11. [Online]. Available: <https://blog.google/innovation-and-ai/models-and-research/gemini-models/3-8-flash-and-3-8-flash-cyber/>
- [28] Anthropic, "Introducing Claude Haiku 5.5," Oct. 2026, accessed 2026-10-11. [Online]. Available: <https://www.anthropic.com/claude-haiku-5-5>
- [29] OpenAI, "Introducing GPT-5.4 mini and nano," Mar. 2026, accessed 2026-10-11. [Online]. Available: <https://openai.com/index/introducing-gpt-5-4-mini-and-nano/>
- [30] Meta AI, "Introducing Muse Spark 1.3," Sep. 2026, accessed 2026-10-11. [Online]. Available: <https://research.meta.ai/blog/introducing-muse-spark-1-3>
- [31] Z.ai, "GLM-5.3: Frontier coding with emergent cyber capabilities," Aug. 2026, accessed 2026-10-11. [Online]. Available: <https://z.ai/blog/glm-5-3>
- [32] Moonshot AI, "Kimi K2.7 Code," Jun. 2026, accessed 2026-10-11. [Online]. Available: <https://www.kimi.com/en/help/kimi-code/whats-new>
- [33] Alibaba Cloud, "Alibaba releases Qwen3.8-Flash with innovative model architecture delivering optimal price-performance," Aug. 2026, accessed 2026-10-11. [Online]. Available: <https://www.alibabacloud.com/blog/603503>
- [34] Mistral AI, "Mistral medium 3.5," Apr. 2026, accessed 2026-10-11. [Online]. Available: <https://docs.mistral.ai/models/mistral-medium-3-5-26-04>
- [35] S. Baltes, F. Angermeir, C. Arora, M. M. Barón, C. Chen, L. Böhme, F. Calefato, N. Ernst, D. Falessi, B. Fitzgerald, D. Fucci, J. He, C. Treude, M. Kalinowski, S. Lambiase, D. Russo, M. Lungu, C. M. Montes, L. Prechelt, P. Ralph, R. van Tonder, and S. Wagner, "Guidelines for empirical studies in software engineering involving large language models," 2026. [Online]. Available: <https://arxiv.org/abs/2508.15503>
- [36] G. C. Cawley and N. L. Talbot, "On over-fitting in model selection and subsequent selection bias in performance evaluation," *J. Mach. Learn. Res.*, vol. 11, pp. 2079–2107, Aug. 2010. [Online]. Available: <https://www.jmlr.org/papers/v11/cawley10a.html>
- [37] T. Ahmed, P. Devanbu, C. Treude, and M. Pradel, "Can llms replace manual annotation of software engineering artifacts?" 2025. [Online]. Available: <https://arxiv.org/abs/2408.05534>
- [38] M. Bano, R. Hoda, D. Zowghi, and C. Treude, "Large language models for qualitative research in software engineering: exploring opportunities and challenges," *Automated Software Engg.*, vol. 31, no. 1, Dec. 2023. [Online]. Available: <https://doi.org/10.1007/s10515-023-00407-8>
- [39] J. Gorostidi, A. Ait, J. Cabot, and J. L. C. Izquierdo, "On the creation of representative samples of software repositories," 2024. [Online]. Available: <https://arxiv.org/abs/2410.00639>
- [40] M. Nagappan, T. Zimmermann, and C. Bird, "Diversity in software engineering research," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2013. New York, NY, USA: Association for Computing Machinery, 2013, pp. 466–476. [Online]. Available: <https://doi.org/10.1145/2491411.2491415>
- [41] N. Calderon, R. Reichart, and R. Dror, "The alternative annotator test for LLM-as-a-judge: How to statistically justify replacing human annotators with LLMs," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vienna, Austria: Association for Computational Linguistics, Jul. 2025, pp. 16051–16081. [Online]. Available: <https://aclanthology.org/2025.acl-long.782/>
- [42] Y. Benjamini and D. Yekutieli, "The control of the false discovery rate in multiple testing under dependency," *The Annals of Statistics*, vol. 29, no. 4, pp. 1165 – 1188, 2001. [Online]. Available: <https://doi.org/10.1214/aos/1013699998>
- [43] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961. [Online]. Available: <http://www.jstor.org/stable/2282330>
- [44] C. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, 1999. [Online]. Available: <https://doi.org/10.1109/32.799955>
- [45] E. H. Bradley, L. A. Curry, and K. J. Devers, "Qualitative data analysis for health services research: Developing taxonomy, themes, and theory," *Health Services Research*, vol. 42, no. 4, pp. 1758–1772, 2007. [Online]. Available: <https://doi.org/10.1111/j.1475-6773.2006.00684.x>
- [46] D. S. Cruzes and T. Dybå, "Recommended steps for thematic synthesis in software engineering," in *2011 International Symposium on Empirical Software Engineering and Measurement*, 2011, pp. 275–284. [Online]. Available: <https://doi.org/10.1109/ESEM.2011.36>
- [47] G. Guest, E. Namey, and M. Chen, "A simple method to assess and report thematic saturation in qualitative research," *PLOS ONE*, vol. 15, no. 5, p. e0232076, 2020. [Online]. Available: <https://doi.org/10.1371/journal.pone.0232076>
- [48] E. L. Kaplan and P. Meier, "Nonparametric estimation from incomplete observations," *Journal of the American Statistical Association*, vol. 53, no. 282, pp. 457–481, 1958. [Online]. Available: <https://doi.org/10.1080/01621459.1958.10501452>
- [49] J. L. Bernal, S. Cummins, and A. Gasparrini, "Interrupted time series regression for the evaluation of public health interventions: a tutorial," *International Journal of Epidemiology*, vol. 46, no. 1, pp. 348–355, Feb. 2017. [Online]. Available: <https://doi.org/10.1093/ije/dyw098>
- [50] J. M. Hilbe, *Negative Binomial Regression*, 2nd ed. Cambridge: Cambridge University Press, 2011. [Online]. Available: <https://www.cambridge.org/9780521198158>
- [51] N. Saarimäki, S. Moreschini, F. Lomio, R. Peñaloza, and V. Lenarduzzi, "Towards a robust approach to analyze time-dependent data in software engineering," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2022, pp. 36–40. [Online]. Available: <https://doi.org/10.1109/SANER53432.2022.00015>
- [52] M. Robredo, N. Saarimäki, R. Peñaloza, and V. Lenarduzzi, "Ignoring time dependence in software engineering data. a mistake," 2023. [Online]. Available: <https://arxiv.org/abs/2311.03114>
- [53] W. K. Newey and K. D. West, "A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix,"

- Econometrica*, vol. 55, no. 3, pp. 703–708, 1987. [Online]. Available: <https://doi.org/10.2307/1913610>
- [54] H. Akaike, “A new look at the statistical model identification,” *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974. [Online]. Available: <https://doi.org/10.1109/TAC.1974.1100705>
 - [55] G. M. Ljung and G. E. P. Box, “On a measure of lack of fit in time series models,” *Biometrika*, vol. 65, no. 2, pp. 297–303, 1978. [Online]. Available: <https://doi.org/10.1093/biomet/65.2.297>
 - [56] J. Lopez Bernal, S. Soumerai, and A. Gasparrini, “A methodological framework for model selection in interrupted time series studies,” *Journal of Clinical Epidemiology*, vol. 103, pp. 82–91, Nov. 2018. [Online]. Available: <https://doi.org/10.1016/j.jclinepi.2018.05.026>
 - [57] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, “An in-depth study of the promises and perils of mining GitHub,” *Empirical Software Engineering*, vol. 21, no. 5, pp. 2035–2071, Oct. 2016. [Online]. Available: <https://doi.org/10.1007/s10664-015-9393-5>
 - [58] C. A. Furia, R. Torkar, and R. Feldt, “Towards causal analysis of empirical software engineering data: The impact of programming languages on coding competitions,” *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 1, 2024. [Online]. Available: <https://doi.org/10.1145/3611667>
 - [59] A. D. B. Costa, “Quantum-safe cryptography: A migration framework for legacy systems toward NIST PQC standards with the crypto-agility readiness score,” *Cryptology ePrint Archive*, Paper 2026/1467, 2026. [Online]. Available: <https://eprint.iacr.org/2026/1467>
 - [60] N. Rameshan and G. Messmer, “An assessment framework for application-level cryptographic agility,” 2026. [Online]. Available: <https://arxiv.org/abs/2606.13425>
 - [61] R. Bögli, J. Spieler, and T. Kehrer, “Bf-cbom: Uncovering cryptographic assets through comparative cbom analysis at scale,” in *Proceedings of the 2026 34th IEEE/ACM International Conference on Program Comprehension*, ser. ICPC ’26. New York, NY, USA: Association for Computing Machinery, 2026, pp. 463–467. [Online]. Available: <https://doi.org/10.1145/3794763.3794831>
 - [62] C. Näther, D. Herzinger, S.-L. Gazdag, J.-P. Steghöfer, S. Daum, and D. Loebenberger, “Migrating software systems toward post-quantum cryptography-a systematic literature review,” *IEEE Access*, vol. 12, pp. 132 107–132 126, 2024. [Online]. Available: <https://doi.org/10.1109/ACCESS.2024.3450306>
 - [63] N. Rattanavipanon, J. Suaboot, and W. Werapun, “A toolchain for assisting migration of software executables towards post-quantum cryptography,” *IEEE Access*, vol. 13, pp. 4368–4380, 2025. [Online]. Available: <http://dx.doi.org/10.1109/ACCESS.2024.3524309>
 - [64] S. Chatterjee, A. Joshi, D. K. N., A. Musku, R. M. P. P. V., R. K. Panda, and B. M. Jadav, “Benchmarking and integration of NIST post-quantum cryptography on Android devices,” *EPJ Web of Conferences*, vol. 360, p. 01027, 2026. [Online]. Available: <https://doi.org/10.1051/epjconf/202636001027>
 - [65] “Anonymized replication package for the study of PQC adoption,” 2026. [Online]. Available: https://anonymous.4open.science/r/pqc_adoption/