CryptoCensus: Cryptographic Posture and Post-Quantum Readiness of Docker Hub

Cristhian Kapelinski[1], **Diego Kreutz**[1]

[1]AI Horizon Labs, Universidade Federal do Pampa (UNIPAMPA)

{cristhianavila.aluno,diegokreutz}@unipampa.edu.br

Abstract. *Under a draft schedule from the National Institute of Standards and Technology (NIST), affected systems would stop using Rivest–Shamir–Adleman (RSA) and elliptic-curve cryptography after 2035. Traffic recorded today can be decrypted once a sufficiently large quantum computer exists. How far published container images have moved toward standardized post-quantum algorithms has not been openly measured. CryptoCensus gives every repository in a 12,716,568-repository crawl the same chance of selection. We scan 11,962 images and catalog 4,211,380 certificates and key files. 100% are quantum-vulnerable and 0 are post-quantum. Although 801 images (about one in fifteen) include a library that supports post-quantum algorithms, none contains a post-quantum key or certificate. The images also contain weak cryptography. 43% of certificates outside trust-store paths use Secure Hash Algorithm 1 (SHA-1) or Message-Digest Algorithm 5 (MD5) signatures, 512-bit RSA keys remain, and 36 private keys in operational paths recur across distinct images. Docker Hub images have yet to begin their post-quantum transition, even where supporting libraries are installed.*

1. Introduction

Containers are a mainstay of modern software deployment. A report on the Cloud Native Computing Foundation's 2024 Annual Survey states that one quarter of respondents use cloud-native techniques for nearly all development and deployment [Silverthorne and Hendrick 2025]. Docker Hub, the largest public registry, holds over 14 million container images, downloaded more than 11 billion times a month [Docker, Inc. 2025]. Container images bundle applications with their dependencies, which can include certificates and private keys. The operational consequence is measurable. Dahlmanns et al. found that 275,269 Transport Layer Security (TLS) and Secure Shell (SSH) hosts used 740 compromised private keys from container images for authentication [Dahlmanns et al. 2023]. Cryptographic files shipped in images can therefore affect Internet-facing services.

This dependence is what a quantum computer threatens. Shor's algorithm breaks RSA cryptography on a sufficiently large machine [Shor 1997], and its extension to elliptic-curve discrete logarithms breaks elliptic-curve cryptography [Proos and Zalka 2003]. The exposure precedes the machine itself, because an adversary can record encrypted traffic today and decrypt it once such a computer exists, so any data with a long confidentiality lifetime is already at risk [Mosca 2018]. Parts of the ecosystem have begun to respond. By late 2025, most human-initiated traffic to Cloudflare already negotiated a hybrid post-quantum key exchange [Westerbaan 2025]. NIST standardized post-quantum replacements in 2024 and now proposes to deprecate affected classical algorithms at the 112-bit security level after 2030 and disallow them entirely after 2035 [NIST

2024, Moody et al. 2024]. Migrating an ecosystem on that schedule requires knowing where the vulnerable cryptography is and how much of it has already moved.

For container images, no open population-level measurement exists. Prior Docker Hub studies measured vulnerabilities, secrets, and malware rather than cryptographic readiness. They scanned 356,218 images [Shu et al. 2017] and 2.2 million images [Liu et al. 2020], and crawled twelve million repositories [Shi et al. 2025]. These studies use vulnerability, malware, or secret scanners rather than a cryptographic inventory. The one study that opens images and extracts key material recovers 52,107 private keys, but focuses on leakage and live-host reuse rather than algorithm strength or quantum safety [Dahlmanns et al. 2023]. Their popularity-based samples describe the images developers reach for rather than the registry as a whole. Our prior ChimangoScan study [Kapelinski et al. 2026] ranks images by exposure, an image's own pull count plus the pulls of every image that inherits its layers, and scans the 52,895 highest-exposure images. In its crawl, 95% of repositories have fewer than 1,000 pulls and are therefore mostly outside that scan. The present work draws from the same crawl with equal probability for every repository to estimate the registry-wide population of pullable `latest` images rather than only the most-pulled images. A free public dashboard began indexing cryptographic risk in popular Docker Hub images in October 2025 [SandboxAQ 2025]. It describes its analysis at a high level but does not release the scanner, sampling frame, or raw results needed to reproduce it.

We close this gap with CryptoCensus, a measurement pipeline and dataset. From our 12,716,568-repository ChimangoScan crawl [Kapelinski et al. 2026], we give every repository the same chance of selection and extract its certificates, keys, libraries, and weak protocol tokens. We separate files under known trust-store paths from material found elsewhere. The 11,962 images we successfully pulled and analyzed provide estimates for repositories with a pullable `latest` image that meet the scan bounds, rather than for the registry's most popular images alone. The census finds no post-quantum key or certificate, including in the images that already contain a supporting library, while the classical cryptography shipped in these images also contains weak signatures, short RSA keys, and reused private keys. This paper contributes, to our knowledge, the first equal-probability census of cryptographic posture and post-quantum readiness on Docker Hub (Section 3); an estimate of readiness that identifies installed but unused post-quantum support (Section 4.2); a measurement of current weaknesses (Section 4.3); and an analysis of key reuse and RSA moduli that share a prime factor (Section 4.4). An open artifact bundles the pipeline, dataset, and sampling frame (Section 5).

2. Related Work

Table 1 positions CryptoCensus against directly comparable studies by the population and sample they inspect, whether they inventory cryptography and post-quantum readiness, and whether their design supports population inference. Counts follow the order in the population column. For [Dahlmanns et al. 2023], the study analyzes 337,171 images assembled from Docker Hub and 8,076 Internet-accessible private registries; the host count is the number of Internet-facing TLS and SSH hosts found using compromised keys, not the total number searched. Network-measurement building blocks and surveys stay in the prose.

Table 1. CryptoCensus versus comparable measurement studies.

Work	Population opened	Measured count	Crypto inventory	Post-quantum	Population inference
[Shu et al. 2017]	Images	356,218	✗	✗	✗
[Liu et al. 2020]	Images	2.2M	✗	✗	✗
[Shi et al. 2025]	Images	33,952	✗	✗	✗
[Kapelinski et al. 2026]	Images	52,895	✗	✗	✗
[Dahlmanns et al. 2023]	Images Registries Hosts	337,171 8,076 275,269	✓	✗	✗
[Strauss et al. 2025]	Apps	4,018	✓	✓	✗
CryptoCensus (this work)	**Images**	**11,962**	✓	✓	✓

The security of the Docker Hub ecosystem has been studied since the scan of 356,218 images that established how vulnerabilities propagate from parent to child images [Shu et al. 2017]. Later work cataloged 2.2 million images and characterized vulnerability and malware risks in popular subsets [Liu et al. 2020]; related the technical lag of Debian-based images to their vulnerabilities [Zerouali et al. 2019]; compared 2,500 images across official, verified, certified, and community tiers [Wist et al. 2021]; and inspected 33,952 influential images, selected by pull count and dependency weight, for vulnerabilities, secrets, and malware [Shi et al. 2025]. ChimangoScan scanned the 52,895 highest-exposure images with six scanners and found that the reported posture depends strongly on the tool [Kapelinski et al. 2026]; companion measurements apply the pipeline to a uniform random sample [Kapelinski and Kreutz 2026b] and to Linux operating-system base images [Kapelinski and Kreutz 2026a]. ChimangoScan also supplies the 12,716,568-repository list from which we draw. These efforts map vulnerabilities and secrets across the ecosystem, while cryptographic posture and post-quantum readiness remain outside their scope. Their popularity- or keyword-based designs describe selected images rather than giving each repository the same chance of inclusion.

Two efforts come closer to our dimension. SandboxAQ's Open Cryptography service [SandboxAQ 2025] inventories libraries, algorithms, keys, certificates, and secrets in popular Docker Hub images and presents risks in a public dashboard. The published description does not provide an executable protocol, sampling frame, scanner implementation, or raw results, leaving reproducible population measurement outside its scope. Closest to our concerns, [Dahlmanns et al. 2023] extract 52,107 private keys from images and analyze certificate trust and validity; their focus is leakage and live-host reuse, checked against Censys [Durumeric et al. 2015], a public search engine over continuous Internet-wide scans of hosts and certificates, leaving algorithm strength and post-quantum readiness outside their scope. CryptoCensus is complementary to both, in dimension (cryptography rather than known vulnerabilities) and in frame (uniform-random rather than popularity).

Measuring deployed cryptography at scale is itself an established line, but aimed at the network rather than the artifact. Internet-wide scanning characterized the TLS and X.509 ecosystem [Holz et al. 2011, Durumeric et al. 2013] and was later generalized into Censys itself [Durumeric et al. 2015]. Early weak-key analysis used the

batch greatest common divisor (batch-GCD) test to compare every pair of RSA moduli efficiently [Heninger et al. 2012]. It flags any two keys that share a prime factor and therefore makes both factorable. Run over eleven million RSA moduli collected in Internet-wide TLS and SSH scans, the test recovered the private keys of half a percent of TLS hosts, with the weak keys concentrated in embedded network devices. The test was later applied to developer SSH keys [Bäumer et al. 2025]. These studies measure reachable hosts or key-hosting platforms, not the cryptography shipped inside software artifacts; we bring their instruments, including the batch-GCD test, to a container-image population. Closer to the artifact, pervasive misuse of cryptographic application programming interfaces (APIs) was found across 17 enterprise Java artifacts [Almeida de Jesus et al. 2025]; we carry that lens from one organization's source to the public registry at population scale.

Post-quantum readiness has so far been studied off the registry. A static analysis of 4,018 Android apps found MD5 and SHA-1 pervasive, no post-quantum cryptography (PQC) in production, and only a few unused imports of PQC libraries [Strauss et al. 2025], the closest analog to our central finding. Post-quantum TLS is already practical in performance terms [Sosnowski et al. 2023], while cryptographic transition and agility remain under-studied relative to algorithm design [Ott et al. 2023]. We report against the CycloneDX Cryptography Bill of Materials (CBOM) standard [CycloneDX 2024], which we also run as an independent cross-check, and estimate readiness for the sampled population of pullable `latest` images.

3. Census Design

Figure 1 presents the measurement design used to estimate cryptographic posture from the Docker Hub crawl. The design draws repositories with equal probability, resolves each reference to an immutable content identifier, downloads and flattens each resolved image, extracts and cross-checks its cryptographic material, and classifies the results.

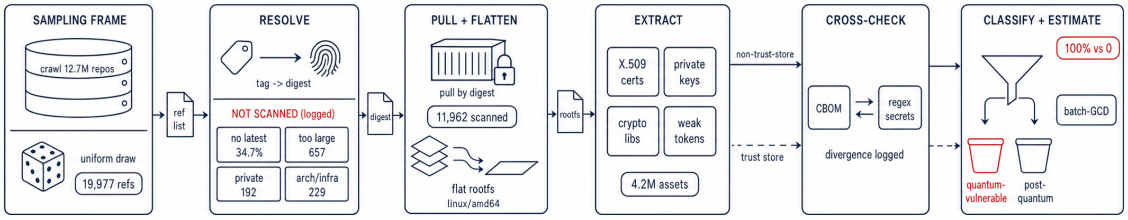

Figure 1. The CryptoCensus pipeline.

Sampling frame and draw. Prior container studies rank images by popularity, which characterizes the images developers use most rather than the registry as a whole. We instead sample with equal probability from a 12,716,568-repository crawl [Kapelinski et al. 2026], pairing each crawled repository with its `latest` tag to form one candidate reference per repository. The crawl enumerates repositories but not their tag lists, so `latest`, the registry's default tag, is the one reference available for every repository. The draw orders this list by the 256-bit Secure Hash Algorithm (SHA-256) hash of `cryptocensus:`*reference* and takes the first 20,000 entries. This budget gives image-level proportions confidence intervals of about one percentage point at feasible download

cost. Of these entries, 19,977 produced a determinate scan record; the remaining 23 returned no result and are excluded. Most repositories in the crawl have few pulls, so equal-probability sampling covers rarely pulled images that popularity-ranked studies largely miss.

Image-level proportions are estimates over the repository frame and use 95% confidence intervals. Asset- and certificate-level proportions are descriptive counts of the extracted material. Because assets are grouped within images, we do not treat them as independent samples or attach binomial confidence intervals to them.

Resolution and pull. A Docker tag is mutable, so a finding keyed to a tag cannot be re-checked later. The census therefore asks the registry to resolve each reference to its content digest and pulls that digest, so the measurement is reproducible to the byte even after tags move. References that do not resolve are logged by registry error and included in the frame-resolution analysis of Section 4.1. The available collection machines used `amd64` processors, so pulls were restricted to the `linux/amd64` image variant, which is also the registry's default platform. Pulls are bounded to 2 GiB of compressed image and 4 GiB of extracted filesystem so that a single oversized image cannot exhaust the pull budget; each image is exported directly from registry blobs as a flat filesystem, and a sampled image is untrusted input that is never executed. The extractor parses regular files up to 2 MB; larger files are skipped, so material inside oversized certificate bundles is undercounted.

Extraction and cross-checks. Three detector families analyze each flattened filesystem. The built-in detector supplies the reported asset counts. Its certificate-and-key routine walks regular files up to 2 MB and uses cryptographic parsers for X.509 certificates and private, public, and SSH keys encoded as Distinguished Encoding Rules (DER), Privacy-Enhanced Mail (PEM), or OpenSSH material. For each parsed object, it records the path, algorithm family, key size, signature hash, and public-key fingerprint. A second built-in routine searches only TLS and SSH configuration files for legacy tokens. These include `RC4`; the Data Encryption Standard (`DES`) and Triple DES (`3DES`); `MD5`; Secure Sockets Layer versions 2 and 3 (`SSLv2`, `SSLv3`); `TLSv1.0` and `TLSv1.1`; and the `NULL` and `EXPORT` cipher classes. A third routine parses the `dpkg` and `apk` package databases and matches cryptographic package names and versions to measure installed post-quantum capability.

Two external detectors cross-check this inventory. Gitleaks, a pattern-based secret scanner in the lineage of [Meli et al. 2019], searches the same filesystem for key-bearing findings in nonstandard locations; its output supports the path audit but does not supply the asset counts. CBOM-Lens independently produces a CycloneDX Cryptography Bill of Materials from the same files. Among the 9,888 images where the built-in detector finds a certificate, CBOM-Lens reports a median of half as many and exceeds the built-in count in 18% of images. This comparison establishes that the detectors have different coverage, not the error rate of either detector.

We classify certificates and keys by path. Files under known trust-store paths (`/etc/ssl/certs`, `/etc/pki`, certificate authority (CA) bundles, and runtime bundles such as Python's `certifi`) form the trust-store category; material found elsewhere forms the non-trust-store category. These path-based categories do not establish who cre-

ated or uses a file. Separating them prevents shared CA bundles from dominating the posture, reuse, and shared-prime analyses.

4. Results

Posture covers the 11,962 images we pulled and fully extracted (4,211,380 public-key assets); frame resolution covers every drawn reference whose status is unambiguous. Table 2 collects the headline figures.

Table 2. Census summary.

Images scanned	11,962	Non-trust-store certificates	518,668
Unresolved refs (no `latest`)	34.7%	weak signature (SHA-1/MD5)	43%
Public-key assets	4,211,380	Non-trust-store RSA <2048-bit	40,720
quantum-vulnerable	100%	Non-trust-store keys	178,455
post-quantum	0	Reused fingerprints (non-trust)	2,412
PQC-capable images (unused)	801	RSA moduli (batch-GCD) / shared-prime	6,116 / 4

4.1. Frame resolution

An equal-probability reference does not always resolve. Of the 19,977 references with a determinate outcome, **34.7%** do not resolve to a `latest` manifest, with a 95% confidence interval (CI) of 34.1–35.4%. Another 11,962 resolve and are scanned, while 1,078 resolve but fall outside posture scope. All but one of the unresolved references return manifest-unknown, meaning that the repository is present but has no `latest` tag; one repository was deleted. Seven weeks after the main scan, a registry API check of 200 randomly selected unresolved references found 193 repositories still present, none with a `latest` tag, and 7 deleted in the interim. Their median last push was in 2022. This unresolved fraction follows from pairing each repository with the default tag. Equal-probability sampling frequently selects repositories that publish only version or build tags. Posture estimates apply to repositories that expose a pullable `latest` image and meet the scan bounds.

The 1,078 out-of-scope references (5.4%) break down as 657 whose extracted filesystems exceed the 4 GiB per-image limit, 192 private images, 148 images without a `linux/amd64` variant, and 81 images affected by transient registry or infrastructure errors. They resolve, and so count as resolved above, but carry no posture measurement; everything below is measured over the 11,962 scanned images.

4.2. Post-quantum readiness

We classify each certificate and key file against quantum-vulnerable families: RSA; finite-field and elliptic-curve Diffie–Hellman; the Digital Signature Algorithm (DSA); ElGamal; the Elliptic Curve Digital Signature Algorithm (ECDSA); and EdDSA, the Edwards-curve Digital Signature Algorithm. The post-quantum families include the Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM), Module-Lattice-Based Digital Signature Algorithm (ML-DSA), Stateless Hash-Based Digital Signature Algorithm (SLH-DSA), and the other NIST candidates and standards listed in the artifact. All **4,211,380** extracted assets are quantum-vulnerable and none are post-quantum. They comprise 518,668 non-trust-store certificates, 178,455 non-trust-store key files, 3,511,725

trust-store certificates, and 2,532 trust-store key files; the figures that follow show the non-trust-store category only. Beyond RSA and generic elliptic-curve keys, the assets include DSA, finite-field Diffie–Hellman, and Ed25519, Ed448, or X25519 material, all of which remain classical. **801** of 11,962 images (6.7%, 95% CI 6.3–7.2%) contain a library package that supports post-quantum algorithms. We detect OpenSSL packages (`openssl`, `libssl`, or `libcrypto`) at version 3.5 or later, which implement ML-KEM, ML-DSA, and SLH-DSA [OpenSSL Library 2025], and `liboqs` packages. Yet no post-quantum key or certificate appears in these images' filesystems. In these 801 images, library availability alone has not led to detectable use. Android apps show a similar gap. The few post-quantum imports found there are unused [Strauss et al. 2025]. These results describe files statically present in images. Runtime observation could reveal additional post-quantum use that this census cannot detect. Because library versions come only from `dpkg` and `apk`, the 801-image count excludes other package managers and statically linked libraries and is therefore a lower bound on detectable installed capability.

4.3. Present-day weaknesses

Independently of the post-quantum question, the classical cryptography shipped in these images is weak (Figure 2). RSA dominates the non-trust-store public-key assets in Figure 2a; elliptic-curve keys are a distant second, and the remaining families are rare. Among the **518,668** certificates outside trust-store paths, **43%** use SHA-1 or MD5 signatures. Collision attacks let an attacker create different inputs with the same hash, undermining the authenticity that a signature is meant to provide. The Internet Engineering Task Force has considered MD5 unacceptable for digital signatures since 2011, while NIST deprecated SHA-1 for generating signatures that year; practical SHA-1 collisions followed [Turner and Chen 2011, NIST 2022, Stevens et al. 2017]. The corresponding descriptive share among the 3,511,725 trust-store certificates is lower, at 32.3%. Key sizes show the same pattern. **40,720** (6.9%) of the 591,617 non-trust-store RSA keys are smaller than 2048 bits, including 5,552 at exactly 512 bits, a size that can be factored for as little as $75 of cloud time [Valenta et al. 2016]. Legacy protocol and cipher strings also occur in configuration files (Figure 2d). `MD5` and `3DES` are most common; less common findings include `SSLv3`, `TLSv1.1`, `RC4`, `NULL`, `DES`, `SSLv2`, `EXPORT`, and `TLSv1.0`. These tokens identify configurations that mention obsolete or deliberately insecure options, but static presence does not show that an option is enabled at runtime.

4.4. Embedded keys and reuse

The census extracts **178,455** key files outside trust-store paths. Of these, **37,077** are private keys or SSH host keys. We classify their likely roles from file paths, assigning each key to the first matching group: test or example material; language-runtime dependencies such as `site-packages` and `node_modules`; SSH host-key paths; system TLS directories; or application directories such as `/app`, `/srv`, and `/opt`. All remaining paths form the libraries and other files group. Figure 3 shows the result. Test or example material bundled with source accounts for 45%, another 45% occurs in libraries and other system files, and 7% occurs in language-runtime dependencies. Operational paths form the smallest classes. Application directories account for 1.2%, SSH host-key paths for 1.5%, and system TLS directories for 0.8%. Of the 7,141 distinct public-key fingerprints in non-trust-store material, **2,412** occur in more than one image. The fingerprint

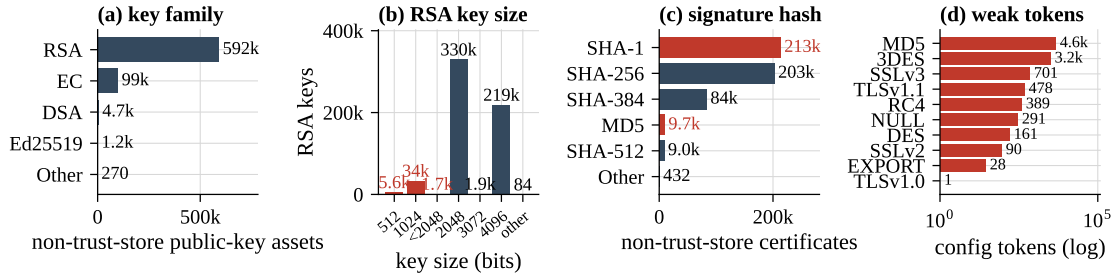

Figure 2. Non-trust-store cryptographic material (red: weak): (a) key family; (b) RSA key size, where "<2048" and "other" are non-standard sizes below and above 2048 bits; (c) certificate signature hash; (d) weak protocol tokens (log scale).

is the SHA-256 hash of the DER-encoded public key, so the same key matches in a key file or certificate. The two most widely shared fingerprints, each present in 4,186 images, are Alpine `apk` package-signing public keys. For the security-relevant reuse count, we retain only private keys in SSH host-key, system TLS, or application paths and exclude tests, examples, documentation, source files, binaries, libraries, and dependency directories. Within this subset, **36** distinct keys recur across multiple images, chiefly as SSH host keys under `/etc/ssh` or TLS server keys inside application images. If a repeated private key is used at runtime, every image carrying it may rely on the same secret.

Figure 3. Non-trust-store private and SSH host keys by path: test/example fixtures; runtime dependencies; SSH host keys; system TLS; application directories; libraries/other files.

Prior work counts private keys exposed in images and matches them to Internet hosts that use the same public keys [Dahlmanns et al. 2023]; here cross-image reuse is one component of shipped posture. We also apply the batch-GCD test of [Heninger et al. 2012] (Section 2) to non-trust-store RSA moduli. The test efficiently finds different RSA moduli that share a prime factor. Of the **6,116** distinct moduli, **4** share a prime and are therefore fully recoverable. This rate is lower than the 0.50% reported for TLS hosts [Heninger et al. 2012]. The test detects only shared-prime failures within this corpus. It cannot find low-entropy keys that do not collide or weaknesses in non-RSA algorithms, so the four keys are a lower bound on this specific failure mode rather than a general audit of key generation.

4.5. Comparison with prior measurements

Table 3 places the container results beside prior measurements of the same property. The populations and denominators differ, so these values provide context rather than direct replications.

Table 3. Container results beside prior measurements of the same property.

Property	Study	Population	Result
Certificate signatures	[Holz et al. 2011]	Live-Web certificates (2009)	17.3% of certificates use MD5
	This work	Non-trust-store image certificates	1.9% MD5; 43% MD5 or SHA-1
Shared-prime RSA	[Heninger et al. 2012]	Internet TLS hosts	0.50% of hosts
	[Bäumer et al. 2025]	Developer RSA keys	0.00056% of keys
	This work	Distinct RSA moduli in images	0.065% of moduli (4/6,116)
Private-key findings	[Dahlmanns et al. 2023]	337,171 container images	52,107 distinct private keys
	This work	11,962 container images	36 distinct operational-path keys recur across images
Post-quantum use	[Strauss et al. 2025]	Production Android apps	None detected
	This work	Image keys and certificates	None detected

Weaknesses. Although MD5 has long been unacceptable for certificate signatures, it signed 17.3% of live-Web certificates in 2009 [Holz et al. 2011] and still signs 1.9% of the non-trust-store certificates here. When SHA-1 is included, the container rate rises to 43% (Figure 2). For shared-prime RSA failures, the 0.065% container-modulus rate is below the 0.50% reported for TLS hosts [Heninger et al. 2012] but two orders of magnitude above the rate for developer SSH keys [Bäumer et al. 2025] (Figure 4b).

Reuse and certificate health. Figure 4a shows, for each reuse count, the share of fingerprints found in at least that many images. Two thirds of the 7,141 fingerprints appear in only one image. A small number spread through base-image inheritance, including two Alpine package-signing keys found in 4,186 images each. Among non-trust-store certificates, 96% are self-signed, 93% carry the certificate-authority basic constraint, and 20% have expired (Figure 4c). Those two shares force an overlap of at least 88%, so most of this material has the shape of a private root: self-signed and marked as a certificate authority. Self-signing alone is not a weakness, because private trust anchors and development certificates may be self-signed. Expiration has a narrower consequence: a verifier checking certificate validity dates should reject an expired certificate if an image uses it at runtime.

Post-quantum adoption. Android apps also showed no production post-quantum cryptography and only a few unused imports [Strauss et al. 2025]. At the transport layer, by contrast, more than half of human-initiated traffic to Cloudflare already negotiates a hybrid post-quantum key exchange [Westerbaan 2025]. We find no post-quantum key or certificate in the sampled container images. These measurements show different adoption levels at the transport and image-filesystem layers, but do not connect the measured images to Cloudflare traffic.

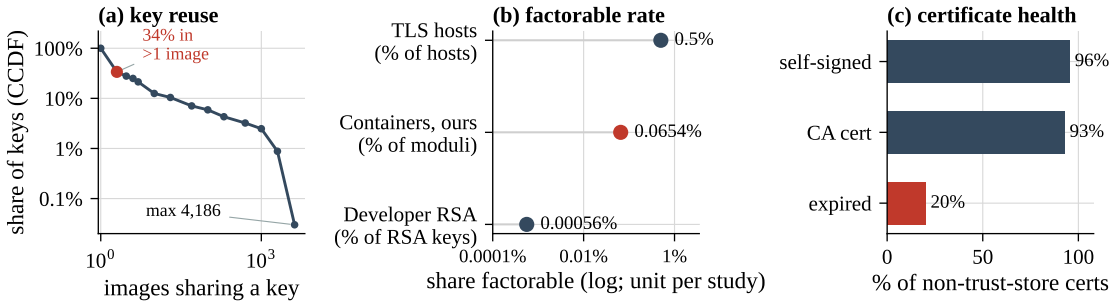

Figure 4. Container measurements and external context (this work in red): (a) key reuse; (b) shared-prime RSA rates; (c) certificate health.

4.6. Operational implications

For practitioners managing containerized services, the main lesson is to treat cryptographic state as deployment data, not image content. Private and host keys should be generated or provisioned uniquely when a container is deployed, through a secret manager or mounted secret, rather than generated during the image build and copied into every instance. Each build should inventory certificates, keys, libraries, and protocol configurations, bind the result to the image digest, and reject short RSA keys, MD5- or SHA-1-signed certificates, and obsolete protocol options intended for production. Base-image updates require the same check because inherited layers can silently introduce cryptographic material. An installed post-quantum-capable library does not complete migration: operators must also configure the protocol, provision post-quantum keys or certificates, and test negotiation at runtime. Finally, findings need context. Trust-store certificates, self-signed development material, and expired files are not equivalent to an exposed operational private key, so deployment policy should use file role and runtime use rather than treating every detected object as a vulnerability.

5. Limitations and Conclusion

Limitations. We cover `linux/amd64` images and files statically present in their filesystems, not material generated, fetched, or negotiated at runtime. Runtime observation could therefore reveal post-quantum use that this census misses. The frame pairs each repository with its `latest` tag, so it excludes repositories that publish only other tags; posture estimates further exclude references that do not resolve or fall outside the scan bounds. Library detection covers `dpkg` and `apk`, making the count of images with detectable post-quantum-capable packages a lower bound on installed capability. The independent extractor comparison reveals coverage differences but does not establish ground-truth error rates. Obfuscated or statically linked cryptography and unsupported key formats may be missed, so the zero count applies only to the formats and files the extractors read. The census is also a single snapshot, so it measures a level rather than a trend; repeating the draw on the same frame would turn post-quantum adoption into a time series.

Ethics and broader impact. We analyze publicly downloadable image files as static, untrusted data. We neither execute images nor test discovered keys against live systems, and report aggregate findings without enabling exploitation.

Conclusion. CryptoCensus's equal-probability sample finds no detectable post-quantum

migration in the inspected image files.[1] All 4,211,380 extracted certificates and keys are quantum-vulnerable, none post-quantum, even in images with supporting libraries. Weak signatures, short RSA keys, and repeated private keys add present-day remediation needs, exposing a gap between installed post-quantum capability and shipped cryptography.

Acknowledgments

This work was partially supported by Capes, CNPq (Grant No. 409743/2025-9), and FAPERGS (Grant Nos. 24/2551-0001368-7 and 24/2551-0000726-1). We thank the anonymous reviewers of SBSeg 2026 for their suggestions and comments. The authors also acknowledge the use of Generative AI as a supporting tool for reviewing and improving the code and manuscript, and the use of OpenAI's GPT Image 2 to generate the schematic pipeline figure (Figure 1).

References

Almeida de Jesus, J., Amaral, L. H. V., and Bonifácio, R. (2025). Cryptographic API misuses in industry: A case study on prevalence and remediation. In *Proc. SBSeg Estendido*, pages 417–425. SBC.

Bäumer, F., Brinkmann, M., Radoy, M., Schwenk, J., and Somorovsky, J. (2025). On the security of SSH client signatures. In *Proc. CCS*, pages 4619–4633.

CycloneDX (2024). Cryptography Bill of Materials (CBOM). `https://cyclonedx.org/capabilities/cbom/`. Standardized as ECMA-424.

Dahlmanns, M., Sander, C., Decker, R., and Wehrle, K. (2023). Secrets revealed in container images: An Internet-wide study on occurrence and impact. In *Proc. ASIA CCS*, pages 797–811.

Docker, Inc. (2025). The World's Largest Container Registry. `https://www.docker.com/products/docker-hub/`.

Durumeric, Z., Adrian, D., Mirian, A., Bailey, M., and Halderman, J. A. (2015). A search engine backed by Internet-wide scanning. In *Proc. CCS*, pages 542–553.

Durumeric, Z., Kasten, J., Bailey, M., and Halderman, J. A. (2013). Analysis of the HTTPS certificate ecosystem. In *Proc. IMC*, pages 291–304.

Heninger, N., Durumeric, Z., Wustrow, E., and Halderman, J. A. (2012). Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *Proc. USENIX Security*, pages 205–220.

Holz, R., Braun, L., Kammenhuber, N., and Carle, G. (2011). The SSL landscape: A thorough analysis of the X.509 PKI using active and passive measurements. In *Proc. IMC*, pages 427–444.

Kapelinski, C. and Kreutz, D. (2026a). A multi-scanner census of the Linux operating-system base images of Docker Hub. In *SBSeg 2026*. SBC.

Kapelinski, C. and Kreutz, D. (2026b). A uniform random-sample security measurement of Docker Hub images. In *SBSeg 2026*. SBC.

[1] `https://github.com/CristhianKapelinski/cryptocensus`

Kapelinski, C., Machado, B., and Kreutz, D. (2026). Vulnerabilities, secrets and misconfiguration in the highest-exposure Docker Hub images. *arXiv preprint arXiv:2608.02669*.

Liu, P., Ji, S., Fu, L., Lu, K., Zhang, X., Lee, W.-H., Lu, T., Chen, W., and Beyah, R. (2020). Understanding the security risks of Docker Hub. In *Proc. ESORICS*, pages 257–276.

Meli, M., McNiece, M. R., and Reaves, B. (2019). How bad can it Git? Characterizing secret leakage in public GitHub repositories. In *Proc. NDSS*.

Moody, D., Perlner, R., Regenscheid, A., Robinson, A., and Cooper, D. (2024). Transition to Post-Quantum Cryptography Standards. `https://csrc.nist.gov/pubs/ir/8547/ipd`. NIST IR 8547, initial public draft.

Mosca, M. (2018). Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security & Privacy*, 16(5):38–41.

NIST (2022). Transitioning away from SHA-1 for all applications. `https://www.nist.gov/news-events/news/2022/12/nist-transitioning-away-sha-1-all-applications`.

NIST (2024). Announcing Approval of Three Federal Information Processing Standards (FIPS) for Post-Quantum Cryptography. `https://csrc.nist.gov/news/2024/postquantum-cryptography-fips-approved`.

OpenSSL Library (2025). OpenSSL 3.5 Final Release. `https://openssl-library.org/post/2025-04-08-openssl-35-final-release/`.

Ott, D., Paterson, K., and Moreau, D. (2023). Where is the research on cryptographic transition and agility? *Commun. ACM*, 66(4):29–32.

Proos, J. and Zalka, C. (2003). Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information & Computation*, 3(4):317–344.

SandboxAQ (2025). Introducing Open Cryptography: A Public Resource for Assessing Cryptographic Risk in Open-Source Software. `https://www.sandboxaq.com/post/introducing-opencryptography`.

Shi, H., Ying, L., Chen, L., Duan, H., Liu, M., and Xue, Z. (2025). Dr. Docker: A large-scale security measurement of Docker image ecosystem. In *Proc. ACM Web Conf. (WWW)*, pages 2813–2823.

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509.

Shu, R., Gu, X., and Enck, W. (2017). A study of security vulnerabilities on Docker Hub. In *Proc. CODASPY*, pages 269–280.

Silverthorne, V. and Hendrick, S. (2025). Cloud Native 2024: Approaching a Decade of Code, Cloud, and Change. `https://www.cncf.io/reports/cncf-annual-survey-2024/`.

Sosnowski, M., Wiedner, F., Hauser, E., Steger, L., Schoinianakis, D., Gallenmüller, S., and Carle, G. (2023). The performance of post-quantum TLS 1.3. In *Proc. CoNEXT Companion*, pages 19–27.

Stevens, M., Bursztein, E., Karpman, P., Albertini, A., and Markov, Y. (2017). The first collision for full SHA-1. In *Proc. CRYPTO*, pages 570–596.

Strauss, J., Upadhyay, K., Siddique, A. B., Baggili, I., and Farooq, U. (2025). Assessing and enhancing quantum readiness in mobile apps. arXiv:2506.00790. Poster, IEEE S&P.

Turner, S. and Chen, L. (2011). Updated security considerations for the MD5 message-digest and the HMAC-MD5 algorithms. Technical Report RFC 6151, Internet Engineering Task Force.

Valenta, L., Cohney, S., Liao, A., Fried, J., Bodduluri, S., and Heninger, N. (2016). Factoring as a service. In *Proc. Financial Cryptography*, pages 321–338.

Westerbaan, B. (2025). State of the post-quantum Internet in 2025. `https://blog.cloudflare.com/pq-2025/`.

Wist, K., Helsem, M., and Gligoroski, D. (2021). Vulnerability analysis of 2500 Docker Hub images. In *Proc. SAM*, pages 307–327. Springer.

Zerouali, A., Mens, T., Robles, G., and González-Barahona, J. M. (2019). On the relation between outdated Docker containers, severity vulnerabilities, and bugs. In *Proc. IEEE SANER*, pages 491–501.