

The State of the 2026 Public PKI

Three Migrated Failure Modes from a 44.6-Million-Certificate Audit

ZetaCert Research — Mouhamadou Thioub, Research Lead Version 1.0 — Public release, 2026-10-11
Vendor and operator identities are anonymized in this public edition. All findings were disclosed to the affected parties through coordinated-disclosure channels (vulnerabilities@zetacert.com) prior to publication.

Executive Summary

Between February and April 2026, ZetaCert Research audited 44,642,253 X.509 certificates harvested from the public Certificate Transparency ecosystem and active TLS scanning. The corpus comprises 24.9M RSA and 19.8M ECC certificates, spanning every major commercial CA, enterprise issuer, and a deep tail of embedded/IoT firmware deployments.

We applied four well-established cryptanalytic detectors at corpus scale: batch GCD across 6.6M moduli (Bernstein 2004 / Heninger et al. 2012), Fermat factorization across 25.8M moduli (close-prime sweep at depth $k=100$, plus a deeper $k=2,000,000$ sweep on the IoT-scanned subset), Hidden Number Problem / lattice-based ECDSA nonce-bias recovery across 23 (issuer, curve) groups and 1,436 TLS-handshake-extracted signatures, and a strict CA-misissuance check for non-public namespace CN values. **All four detectors returned clean negatives.** This is itself a noteworthy result: the modern CT-visible public PKI, as operated by major commercial CAs, is robustly implemented against the classical broken-RNG and template-misissuance failure modes.

Cryptographic risk has not disappeared. It has migrated. We document three failure-mode classes that *do* show up in the 2026 corpus, each representing a different generation of mistake:

- **Mode A — RSA migration to the IoT/firmware long-tail.** The classical broken-RNG / weak-keygen pattern documented by Heninger et al. (2012) and by Nemec et al. (ROCA 2017) has migrated out of the commercial CA ecosystem and into vendor firmware defaults. We identified 47 distinct findings affecting consumer routers, enterprise networking gear, NAS appliances, industrial CPEs, and management-plane controllers. One vendor cluster (Vendor C) spans a 12-year SDK firmware default (RSA-2048 with $e=17$ rather than the standard $e=65537$); another vendor cluster (Vendor H) shows 1040-bit-modulus generation bugs surfacing as recently as 2026.
- **Mode B — Voluntary ECDSA key concentration in operator fleets.** A previously-undocumented architectural pattern: six independent operators (registrars, brand-protection providers, lead-generation SaaS, domain-parking platforms) reuse a single ECDSA P-256 (or P-384) private key as the SubjectPublicKeyInfo across their entire customer fleet. The combined exposure is 25,105 certificates concentrated on 9 distinct keys. The largest single cluster signs 18,866 certificates with one key; another signs the certificates of a brand-protection portfolio that includes Fortune 500 clients (entertainment, pharmaceutical, consumer goods, agribusiness). One smaller cluster covers 8 SCADA hosts in a single industrial deployment. A separate mass-issuance “ghost” cluster covers 1.65M certificates on a single key — issuance-only, never deployed — and serves as a methodological observation about CT-visible bulk ACME activity rather than an active-compromise finding.
- **Mode C — Designed-in certificate-lifetime failures.** A third class is neither involuntary (broken RNG) nor voluntary-architectural (operator key reuse), but baked in at the firmware design layer. We identified 775 management-plane certificates across ≈ 721 distinct devices from 4 maintained vendors whose `notAfter` field is hard-coded to 2038-01-19 03:14:08 UTC — the `INT_MAX` boundary of 32-bit signed `time_t`. (An additional 242 single-host certificates share the same expiry value without the fleet-template structure, bringing the total `INT_MAX`-validity cert count in the corpus to 1,017.) The affected vendor-fleet population spans firewalls, consumer routers (28 distinct models from one vendor, Vendor A, alone), security appliances, and IP cameras. At the boundary, the entire fleet’s management certificates expire simultaneously, with no graceful renewal path baked into the firmware. We additionally identified two certificates from a major server-platform vendor (Vendor L) whose

`notAfter` field is *prior to* 1970-01-01 (date overflow in the opposite direction) and one self-signed firewall management certificate (Vendor N) impersonating a retired commercial CA in its issuer DN — both pointing at the same root cause: time-handling and template-construction defects in firmware certificate generation.

The structural lesson across the three modes: the cryptographic primitives are no longer where modern PKI breaks. The breakage has moved up the stack — to entropy at firmware install time (Mode A), to operator architecture (Mode B), and to time/template construction code (Mode C). For CISOs, the operational implication is that a passing TLS scan against your perimeter is no longer evidence that your certificate infrastructure is sound. Vendor firmware defaults, operator-shared keys upstream of your service contract, and design-time time-math errors are all invisible to a standard cipher-suite or vulnerability scanner; they require certificate-content inspection.

This paper presents the methodology, the findings, the disclosure-outcome spectrum across the responding vendors (from “remediation planned” through “best practice acknowledged” through “End of Life — no remediation possible”), and a concrete audit checklist for CISOs and procurement teams.

1. Introduction

The public Web PKI in 2026 sits between two equilibria. Above it, a small number of high-volume commercial CAs (Let’s Encrypt, Google Trust Services, DigiCert, Sectigo, the Microsoft and Amazon issuance estates) have professionalized their key generation, nonce generation, and template handling to a degree that classical large-scale cryptanalytic attacks no longer find purchase against the certificates they issue. Below it, the long tail of certificates — those generated inside firmware, those issued by registrar-side automation pipelines, those bound to management-plane appliances that receive infrequent updates — has not professionalized at the same rate. The cryptographic risk has migrated downward and sideways into this long tail.

This paper reports the results of a four-month audit campaign (February–April 2026) targeting both equilibria. We harvested 33.6M certificates from the public CT logs (Google Xenon and Argon, Cloudflare Nimbus, DigiCert Wyvern and Sphinx, Sectigo Elephant and Tiger, TrustAsia 2026a/b) and supplemented this with 11M certificates obtained from active TLS probing of 24 IoT-relevant ports across the public IPv4 address space, at a polite scan rate of 10–20 connections per second per port. The combined corpus is 44.6M certificates across 24.9M unique RSA moduli and a roughly equal number of distinct ECC public points.

We organize the findings into three failure-mode classes (Sections 3, 4, 5), preceded by Section 2 which documents *what we looked for and did not find*. The negative-results section is methodologically important: it bounds the claims of the positive findings and demonstrates that the ZetaCert Research detection toolchain is well-calibrated against the broader corpus, not just tuned to surface the weaknesses we eventually report.

2. Methodology and Negative Results

2.1 Detectors

Four detectors were run at corpus scale:

Batch GCD (Bernstein 2004). Pairwise greatest-common-divisor sweep across all RSA moduli, designed to surface shared prime factors arising from broken or seeded RNGs at key-generation time. This is the canonical detector that Heninger et al. (2012) used to surface 0.5% weak keys in the 2012 Web PKI. Run on the deduplicated 6.6M-modulus subset of our corpus; runtime ~7 hours on commodity hardware.

Fermat close-prime sweep. Detects RSA moduli generated as $p \times q$ where $|p - q|$ is small enough that Fermat’s algorithm recovers the factorization in seconds. Surfaces a different broken-RNG class than batch GCD: when both primes are drawn from the same biased pool, they are close in absolute value but not shared. Run at depth $k=100$ across 25,804,350 moduli (10 minutes 38 seconds, ~40,500 moduli/second on 12

workers), and additionally at depth $k=2,000,000$ ($20,000\times$ deeper) on the highest-risk subset — the 30,050 moduli sourced from active IoT scanning.

Hidden Number Problem / lattice ECDSA nonce-bias recovery. Lattice-based recovery of an ECDSA signing key from a population of signatures with a small biased nonce, following the classical Boneh-Venkatesan / Nguyen-Shparlinski construction. Run twice: first on 23 (issuer, curve) groups assembled from CT-visible end-entity certificate signatures (61 total lattice attempts at $\text{bias} \in \{2, 4, 8\}$); second on 1,436 signatures extracted live from TLS handshakes against the five operator clusters identified in Section 4.

CA-misissuance scan. Strict-regex scan for publicly-issued certificates whose Subject Common Name resolves to a private namespace (RFC 1918 IPv4, `.local`, `.internal`, `.lan`, `.home`, `.corp`, `.intranet`, `localhost`). Designed to surface CA/B Baseline Requirements §7.1.4.2.1 violations.

2.2 Results

Detector	Population	Hits
Batch GCD	6.6M unique RSA moduli	0 shared factors
Fermat ($k=100$)	25.8M unique RSA moduli	0 close-prime pairs
Fermat ($k=2M$, IoT subset)	22,000 / 30,050 (user-terminated)	0 close-prime pairs
HNP / lattice (CT-extracted sigs)	23 (issuer, curve) groups, 61 attempts	0 recoveries
HNP / lattice (TLS-handshake sigs)	1,436 signatures across 5 operator clusters	0 recoveries
CA misissuance (strict regex)	44.6M certificates	2 false-positive (AWS-internal CA artefacts), 0 genuine

The four detectors produced no positive cryptanalytic recoveries against the modern CT-visible commercial-CA estate. We interpret this as evidence that the major commercial CAs have implemented FIPS-grade HSM-backed key generation, deterministic-or-HSM nonce generation (RFC 6979 or hardware), and template-construction discipline at a level that resists at-scale public-corpus attacks. This is the equilibrium *above* the long tail. The remaining sections document the equilibrium *below* it.

2.3 Methodology lesson — false-positive HNP from a vacuous sanity check

In an early version of the HNP detector, a sanity-check expression at line 132 of `research/ecdsa_nonce.py` read `if k_candidate.bit_length() > n.bit_length() - bias + 2: break.` For curve P-256 (`n.bit_length() = 256`) at `bias = 2`, this evaluates to `> 256`, which is unreachable for any valid `k < n`. The check therefore never broke the loop, and the lattice routine reported every “short” candidate as a recovered key. Initial output included absurd claims of recovered signing keys for CAs known to operate behind FIPS-validated HSMs (Cloudflare TLS Issuing ECC CA 3, several Google Trust Services intermediates, two Microsoft Azure ECC TLS CAs, ZeroSSL ECC, cPanel ECC, TrustAsia, Certainly, nazwaSSL). These “recoveries” were entirely artefacts of the sanity-check bug above: none of the listed CAs exhibited any weakness, each is a correctly-operating, HSM-backed issuer, and we name them only to record which clean populations the vacuous check wrongly flagged. After fixing the bound to `n.bit_length() - bias` (strict) and rejecting `x_candidate ∈ {0, n-1}`, the re-run produced the clean negative result reported above.

We flag this defensively: at least one previously-published “weak-CA HNP” finding may have used template lattice code with a similar latent bug. Anyone running HNP-style detectors at scale should verify their sanity-check bounds against a known-clean CA population *before* drawing conclusions from a positive hit.

3. Mode A — RSA Migration to the IoT/Firmware Long-Tail

The “Mining Your Ps and Qs” finding (Heninger et al., USENIX Security 2012) showed that 0.5% of the then-public Web PKI’s RSA moduli could be factored by simple batch GCD, with the failures concentrated in network appliances and embedded devices that seeded their RNG from low-entropy sources at first boot. Fourteen years later, the underlying class of bug has not disappeared from the Internet — but it has migrated out of the public Web PKI’s CA-issued certificates and into vendor firmware defaults that the public CAs never see and never sign.

The 47 findings in Mode A break down as follows:

- **37 findings** at RSA-1024 modulus length (below the NIST SP 800-131A retirement boundary of 2014). The affected vendor footprint includes a major consumer router brand (Vendor A; 11 findings concentrated in their factory-default management certificates), a US enterprise-networking manufacturer (Vendor B) with End-of-Life devices still in production deployment (3 findings), a Korean (Vendor D) and a Chinese (Vendor E) telecom-equipment vendor, two industrial-CPE manufacturers (Vendor F, Vendor G), an enterprise NAS vendor (Vendor I), an industrial wireless vendor (Vendor J), an enterprise business-automation software suite (Vendor K), a US-headquartered server-platform vendor (Vendor L; 4 findings inside the management-plane controller — the same vendor that produces Mode C’s date-overflow certificates), and a wireless-ISP equipment manufacturer (Vendor H). The full vendor distribution is in Appendix A.
- **10 findings** of non-standard RSA public exponent. The standout case is a German enterprise-networking manufacturer (Vendor C) with 22 certificates spanning 2014 through 2026 — twelve years of firmware releases — that all use $e=17$ rather than the standard $e=65537$. The vendor’s own ECC fleet, by contrast, is fully compliant. This is a localized SDK-branch defect, not an organization-wide cryptographic-discipline gap, and the vendor has acknowledged it as a planned remediation in upcoming firmware releases.

Several findings deserve specific commentary because they illuminate failure modes that purely statistical reporting would obscure:

The 1040-bit modulus. A wireless-ISP vendor (Vendor H) has produced multiple management certificates whose RSA modulus length is 1040 bits — three bits longer than 1024, with the high bits of the modulus consistently structured. This is not a deliberate non-standard size; it is the signature of a key-generation routine that constructs p and q from concatenated-then-corrected RNG output and fails to enforce the modulus length cleanly. Certificates with this signature appear in the corpus as recently as January 2026, indicating that the underlying firmware path is still active in the field.

The retired-CA-impersonating self-signed certificate. A firewall-vendor (Vendor N) management certificate observed during scanning has subject `CN=<firewall-product-redacted>, O=<vendor-redacted>` and issuer `CN=Let's Encrypt Authority X3` — a CA intermediate that was retired in 2021 and that, even in its operational lifetime, never issued one-year-validity certificates with $e=3$ and a generic Common Name. This is a self-signed certificate whose template logic incorrectly retained the previous issuer DN when the device regenerated its own key material. The certificate’s cryptographic content (RSA-2048 with $e=3$) is independently a Mode-A finding; the template construction is a Mode-C finding (Section 5). The case study illustrates how the three failure modes are not orthogonal — the same firmware path can produce defects in multiple categories simultaneously.

Disclosure outcomes (Mode A). Of the seven vendors contacted, one has responded with a planned firmware remediation and an explicit acknowledgement that the report led to internal SDK-branch review; one has responded that the affected hardware has reached End of Life and that no firmware remediation is possible (a structurally different but equally valid coordinated-disclosure outcome); five had not yet responded as of the v0.1 cutoff. Section 6 discusses the policy implications of the End-of-Life response.

4. Mode B — Voluntary ECDSA Key Concentration in Operator Fleets

The Mode-A findings are recognizable as variants of a documented phenomenon (broken-RNG / weak-keygen, dating to 2012). The Mode-B findings are not. We are not aware of a previously-published systematic measurement of the pattern reported here.

We define the pattern as follows. An *operator* — a registrar, a brand-protection vendor, a lead-generation SaaS, a domain-parking platform, a managed-pages provider — terminates TLS for a population of customer domains. In a properly-architected deployment, each terminated domain receives an independently generated key pair (this is the implicit architectural assumption that ACME automation makes). In the operator-concentration anti-pattern, the operator generates *one* ECDSA key pair and reuses it as the SubjectPublicKeyInfo across every certificate it requests on behalf of every customer.

We identified six independent operators exhibiting this pattern in the 2026 corpus:

Operator type	Country	Certificate count	Distinct keys
European registrar (large, ccTLD-focused) — Operator 1	NL	18,866	1
Brand-protection / domain-defense provider (Operator 2)	UK	4,575	1
Marketing / lead-generation SaaS (Operator 3)	BR	1,144	2
Domain-parking and hosting provider (Operator 4)	CA	298	3
Domain-parking provider (Operator 5)	DE	190	1
European managed-pages / hosting provider (Operator 6)	FR	32	1
Total		25,105	9

The brand-protection cluster (Operator 2) is the most structurally significant. The single ECDSA P-256 key signs, simultaneously, certificates for an unaffiliated set of Fortune 500 customers’ brand-defense portfolios — major entertainment-and-media holdings, a global consumer-goods cooperative, a video-game publisher, a defensive-typosquat ring around a major pharmaceutical brand, and an analytical-instruments manufacturer. In the threat model where the operator’s key is recovered (by insider compromise, by hostile-supply-chain access to the operator’s HSM, or by a future cryptanalytic break), the attacker simultaneously gains TLS impersonation capability against thousands of unaffiliated customer domains in a single operation. The customers are not aware of their shared cryptographic fate, because the architectural choice is invisible at the customer-contract layer — the operator’s terms of service describe key management as an operator-internal implementation detail.

We additionally observed two structural sub-cases worth documenting:

The SCADA single-organization cluster. Eight industrial-control hosts (Vendor F; programmable automation controllers, modular communication routers, and supervisory routers) at a single large industrial organization share one ECDSA key. Unlike the operator cases, this is single-tenant — the cryptographic concentration is internal to one organization rather than spanning unaffiliated customers. The case is included as an illustration that the same architectural pattern arises in non-PKI-operator deployments: a procurement

decision to deploy identical industrial-grade certificates from a vendor preset can produce key concentration as a side-effect of operational uniformity.

The mass-issuance ghost cluster. During the audit period, one ECDSA P-256 key was used to generate 1.65M Let’s Encrypt certificates across 1.16M unique subjects in a ten-day window. A random sample of seven of those domains was probed at the TLS layer; none responded. The certificates appear to be issued but not deployed — a CT-visible bulk-ACME footprint with no corresponding active-service surface. We document this as a methodological observation about the disconnect between CT-visible issuance volume and deployed-service reality, rather than as a Mode-B finding in the same risk class as the operator clusters above.

Cryptanalytic depth check. To bound the severity of the operator-concentration finding, we ran the HNP / lattice ECDSA nonce-bias detector against TLS-handshake-extracted signatures from each of the five operator clusters (Section 2). The 1,436-signature, 0-recovery result reported there is the relevant negative: the operator concentration is not compounded by a server-side nonce-generation defect that would make the architectural single-point-of-failure recoverable from passive traffic observation. The risk remains structural — single key, single point of failure, large blast radius — rather than a passive-recovery vulnerability.

Disclosure outcomes (Mode B). All six operators were notified during the v0.1 drafting window via the operator-listed security-contact channel (where one was published) or via the operator’s general-support channel (where no security contact was published). Outcomes will be reported in v1.0 post-grace-expiry.

5. Mode C — Designed-in Certificate-Lifetime Failures

The third failure mode is neither a broken-RNG residue (Mode A) nor an architectural-concentration choice (Mode B) but a defect in the time-handling and template-construction code of firmware certificate-generation routines.

5.1 The Y2038 hard-coded-expiry pattern

The single most operationally consequential finding in this section is a fleet-wide synchronous-expiry event scheduled for 2038-01-19 03:14:08 UTC — the maximum value representable in a signed 32-bit `time_t`. At corpus scope, 1,017 certificates currently in the audit window have `notAfter` set to exactly this value. Of those, 775 (across ≈ 721 distinct probed IPs) concentrate on four maintained vendors whose first-boot templates emit `notAfter = INT_MAX` as a single fleet-wide value:

Vendor category	Distinct devices (IPs)	Cert count	Representative product class
Enterprise firewall / SMB security appliance (Vendor M)	371	399	UTM appliances (default management certificate, single CN template)
Consumer / SOHO networking (Vendor A)	319	340	28 distinct router and modem models from a single vendor (two factory CN templates)
Enterprise firewall (large) — Vendor N	26	31	Next-generation firewall management plane
Network video surveillance (Vendor O)	5	5	IP camera management certificates

Vendor category	Distinct devices (IPs)	Cert count	Representative product class
Identified vendor-fleet subtotal	≈721	775	

The remaining 242 certificates are residual single-host deployments with miscellaneous private-namespace CNs (`localhost`, RFC 1918 prefixes, ad-hoc subjects); they share the Y2038 trigger but not the fleet-amplification structure that makes the four vendor clusters the operational concern of this section.

The structural fact is that, at 03:14:08 UTC on 2038-01-19, the management-plane certificate of every affected device expires *simultaneously* and *worldwide*. Operators of affected devices will, on that morning, find their administrative web interfaces presenting certificate-expired warnings; depending on the management-tool chain, automated configuration deployment may fail closed, monitoring agents may refuse the connection, and SSL-VPN tunnels terminating on the management plane may stop renegotiating. None of the affected vendors’ firmwares contain a graceful renewal path — the certificate is generated once at factory or first-boot time and persists as the management-plane identity for the device’s lifetime.

The root cause across all four vendors is convergent: the firmware generates a self-signed management certificate at first boot and computes its `notAfter` as `INT_MAX` — either explicitly (as the maximum representable 32-bit `time_t` value) or implicitly (as the result of arithmetic on a 32-bit value that overflows to `INT_MAX`). The code path predates the industry’s broad migration to 64-bit time-handling and was not revisited when subsequent firmware releases adopted 64-bit underlying platforms.

5.2 Date-overflow in the opposite direction

A separate finding in the same conceptual class: two management certificates from a major US server-platform vendor (Vendor L) have `notAfter` values *prior to* 1970-01-01. These certificates are technically expired-on-issue from the moment they are generated. The root cause is the mirror image of Y2038: arithmetic on a `time_t` that wraps to a negative value, then is rendered into the certificate’s `GeneralizedTime` field as a pre-epoch date. The same vendor’s product family contributes four findings to Mode A (RSA-1024 management certificates) — both findings live inside the same firmware management-plane controller subsystem.

5.3 The retired-CA-impersonating self-signed certificate

Discussed in Section 3 alongside the Mode-A finding it co-occurs with: the firewall-vendor (Vendor N) management certificate with subject `CN=<firewall-product-redacted>` and issuer `CN=Let's Encrypt Authority X3`. The Mode-C interpretation is that the firmware’s certificate-generation routine retained the issuer-DN field from a previous template version (when the device may have served a Let’s Encrypt-issued certificate via ACME) without clearing it on regeneration. The certificate is cryptographically self-signed but DN-presents as if issued by a defunct commercial intermediate. The defect is in the template-construction code, not the cryptographic primitives — Mode C, not Mode A.

5.4 Why this is a single failure-mode class

Y2038 hard-coded expiry, pre-1970 date-overflow, and template-construction artefacts look superficially unrelated. We argue they are the same class because they share a structural property: the certificate’s *contents* are wrong in a way that the certificate’s *cryptography* would not flag. Each defect is invisible to a key-strength check, a signature-verification check, a chain-validation check, or a cipher-suite check. They become visible only when a tool inspects certificate content with semantic awareness — that the `notAfter` field is implausible, that an issuer DN names a defunct CA, that an identical `notAfter` value appears across certificates from unrelated devices.

This is a category of bug that matures cryptanalysis was not built to find. It is, however, exactly the category of bug that becomes operationally consequential at scale — the Y2038 fleet-wide-simultaneous-expiry alone is

a roadmap to a 2038 outage that is, today, twelve years from inevitability and entirely tractable to remediate through firmware updates issued during that window.

Disclosure outcomes (Mode C). The largest single-vendor finding was disclosed to the vendor’s PSIRT during the v0.1 drafting window with a footprint of 144 IPs at the time of disclosure; continued corpus growth has since brought the observable fleet to ≈ 371 IPs. The disclosure remains accurate as a design-pattern report — the count update is a measurement-window artefact, not a new vulnerability — and will not be re-sent. The cross-vendor expansion ($\approx 319 + 26 + 5$ IPs from three additional vendors) is undergoing per-vendor PSIRT contact verification and disclosure. Outcomes will be reported in v1.0.

6. Disclosure-Outcome Spectrum

A coordinated-disclosure campaign at this scale produces a spectrum of vendor responses. We document the four points on the spectrum represented in the 2026 audit, because they have policy implications beyond the specific findings.

Outcome 1 — Acknowledgement and planned remediation. The German enterprise-networking vendor (Vendor C) whose 22-certificate $e=17$ cluster is documented in Section 3 responded within four days, framed the finding as a “best practice / digital hygiene” matter rather than a vulnerability per se, accepted the attribution-by-fingerprint without questioning the methodology, and committed to adjust the exponent in upcoming firmware releases. This is the cleanest possible disclosure outcome: vendor receives, vendor evaluates, vendor remediates, vendor communicates the timeline.

Outcome 2 — Acknowledgement and structural inability to remediate. The US enterprise-networking vendor (Vendor B) whose 3 RSA-1024 findings are documented in Section 3 responded that the affected hardware has reached End of Life and that no firmware updates are possible. This is not a refusal to remediate — it is a statement that the vendor’s product-lifecycle policy structurally prevents remediation, and the finding therefore exists in a class that coordinated disclosure cannot resolve. The policy implication is that legacy weak-cryptography defects in End-of-Life hardware require regulatory rather than vendor-led remediation paths (extended-support contracts, sunset-crypto requirements at EoL written into compliance frameworks such as the EU Cyber Resilience Act and NIS2).

Outcome 3 — Initial deflection followed by reframing. A bug-bounty-platform-mediated submission produced an initial “does not meet threshold for further review, no evidence of real-world exploitability” response. The reply is a standard bug-bounty-program response to compliance-style findings (a finding’s severity is judged by its exploitability rather than by its policy or compliance impact). Reframing the submission as a coordinated-disclosure-not-bounty matter and providing a fifth corroborating finding (a 1040-bit modulus from 2019, illustrating that the keygen defect pattern is cross-period) changed the platform’s classification of the report.

Outcome 4 — Pending response. Several disclosures sent during the audit window have not produced a response prior to the v0.1 cutoff. This is normal within the 90-day coordinated-disclosure window. v1.0 of this paper will report final outcomes after the 2026-07-25 grace expiry across all batches.

7. Detection Guidance for CISOs and Procurement

The findings in Sections 3–5 are not detectable by standard TLS-cipher-suite scanners or by vulnerability scanners that operate against device service banners. They require certificate-content inspection. The following audit checklist captures the practical detection patterns:

For Mode A (RSA migration). 1. For every TLS-terminating endpoint in your perimeter and management plane, extract the certificate and parse its public-key parameters. Flag any RSA modulus of length other than 2048, 3072, or 4096 (legitimate sizes in 2026 are these three; anything else indicates either a pre-2014 legacy or a keygen defect). Flag any RSA public exponent other than 65537. Flag any modulus length that is

not divisible by 8 (e.g. 1040 bits) as a probable keygen-defect signature. 2. Cross-reference flagged endpoints against your device inventory by vendor and model; firmware-default certificates concentrate by device class, so a single flagged finding is likely indicative of a fleet-wide condition for that model.

For Mode B (operator concentration). 1. For every domain whose TLS termination is delegated to an external operator (registrar-managed certificates, brand-protection service-managed certificates, lead-generation SaaS-managed landing pages, parked-domain managed pages, managed-page hosting), retrieve the certificate served on each domain and extract the `SubjectPublicKeyInfo`. 2. If multiple delegated domains served by the same operator share a `SubjectPublicKeyInfo`, the operator is exhibiting the Mode-B pattern. Engage the operator’s security-contact channel and request architectural confirmation of whether per-domain key generation is in their product roadmap.

For Mode C (designed-in lifetime failures). 1. For every TLS-terminating endpoint, extract the certificate’s `notAfter` field. Flag any `notAfter` equal to 2038-01-19 03:14:07 or 03:14:08 UTC as a Y2038-hard-coded-expiry candidate. Flag any `notAfter` prior to 1970-01-01 as a date-overflow candidate. 2. For every self-signed management certificate, inspect the issuer DN. Flag any issuer DN that names a publicly-recognized CA intermediate (a self-signed certificate must not present a public-CA issuer DN; this is by definition a template-construction defect or a misconfiguration).

ZetaCert operates a free, no-account audit service at audit.zetacert.com that performs all three classes of checks. Submit a hostname or upload a PEM-encoded certificate; the service returns the same detector output that produced the findings reported in this paper.

8. Recommendations

For CISOs and security architects. 1. Add certificate-content inspection (not just cipher-suite scanning) to your perimeter audit cadence. The three failure modes in this paper are invisible to cipher-suite scanners. 2. Add a contractual “key-generation architecture” question to your procurement and vendor-management process for any third-party operator that terminates TLS on your behalf. The Mode-B operator concentration is invisible at the contract layer unless explicitly questioned. 3. Add an inventory query for `notAfter` $\in$ [2038-01-19 03:14:07 UTC, 2038-01-19 03:14:08 UTC] and `notAfter` < 1970-01-01 to your certificate-management tooling. The 2038 cliff is twelve years away and tractable to remediate; pre-emptive identification of affected devices in your fleet now allows you to schedule firmware updates against the vendor remediation windows. 4. For End-of-Life hardware running weak cryptography (Outcome 2 in Section 6), the structural answer is replacement on a schedule that aligns with your compliance framework’s sunset-crypto provisions, not vendor remediation. Maintain a published-and-tracked end-of-cryptographic-life schedule for your appliance inventory.

For vendors generating self-signed certificates in firmware. 1. Audit your certificate-generation code path for time arithmetic that uses 32-bit `time_t`. Any call to `time()` or arithmetic on `time_t` in 32-bit form is a Y2038 hazard regardless of the underlying platform’s 64-bit support. 2. Replace fixed-validity self-signed management certificates with rolling-validity (re-generated periodically on a graceful renewal schedule, with a fallback-to-fresh path on expiry). The single-issue-at-factory model produces fleet-wide synchronous expiry hazards regardless of the chosen expiry date. 3. Audit your template-construction code for retention of issuer-DN fields across certificate regenerations; each regeneration should construct the issuer DN from current device state, not from cached prior-generation values.

For policymakers (CRA, NIS2, equivalent). 1. Compliance frameworks should include explicit sunset-crypto provisions at End of Life: a vendor that ships a device with weak cryptography and subsequently ends firmware support inherits a structural inability to remediate, which the vendor’s product-lifecycle policy alone cannot resolve. The regulatory answer is a defined sunset interval after which weak-cryptography devices may not be operated in regulated contexts.

About ZetaCert Research

ZetaCert is the umbrella company behind PKIFactor (certificate-lifecycle management), ZetaCA (post-quantum-ready certificate authority), and CryptoAudit (free public-key audit service). ZetaCert Research is the team responsible for the audit campaign reported in this paper.

Audit your own certificate fleet at `audit.zetacert.com`. The service is free, requires no account, and applies the same detector toolchain reported in Section 2 against an uploaded certificate or a probed hostname.

For coordinated-disclosure inquiries: `vulnerabilities@zetacert.com`. For commercial inquiries (PKIFactor, ZetaCA): `sales@zetacert.com` — handled separately from any disclosure correspondence.

Version 1.0 — public release, 2026-10-11. Vendor and operator identities are anonymized throughout this edition; each affected party was independently notified through coordinated-disclosure channels (`vulnerabilities@zetacert.com`) ahead of publication. Certificate-subject fields that would identify a vendor are shown redacted for the same reason.

Changelog v0.1 → v0.2: Mode C Y2038 figures refreshed against the live corpus. Identified vendor-fleet count rose from 582 certs / 515 devices to 775 certs / ≈ 721 IPs as scanning expanded the observed footprint of all four templates; 242 residual single-host INT_MAX certificates outside the four vendor templates are reported separately, bringing the total INT_MAX-validity cert count in the corpus to 1,017.