"Less Than 256" Is Not the Weak-Key Set: A Counterexample and a One-Sentence Correction for the ECDSA Small-Scalar Rule Proposed for the CA/Browser Forum TLS Baseline Requirements, with a Lean Proof

Nobuo Otoi, independent researcher, ba-cybersecurity.com, Tokyo, Japan

November 2026 (version 1, draft; not yet published)

Abstract

After eight eMudhra intermediate CA certificates were found to have P-384 private scalars below 256 (Mozilla Bugzilla 2083449, October 2026), a draft ballot for the CA/Browser Forum TLS Baseline Requirements (servercert pull request #683, 2026-10-22) adds to Section 6.1.6 the requirement that the CA "ensure that the public key does not correspond to a Private Key whose private scalar is less than 256". We show that this sentence does not describe the set of keys that the search it alludes to recovers. For every k in 1..255 the scalar $n - k$, where n is the group order, is not "less than 256", yet its public key $-kG$ has the x-coordinate of kG and is found by the same 255-step x-only comparison; and the scalar $k^{-1} \bmod n$ is not "less than 256" either, yet its public key Q satisfies $kQ = G$ and is found by 255 additions of Q. We verify both families against the eight real eMudhra public keys (recovering the published scalars 4, 56, 66, 115, 155, 156, 230, 237, and from each building a key the draft accepts). We propose to replace the sentence by "ensure that neither the private scalar d nor $n - d$ is less than 2^{16}, and that no integer k less than 2^{16} satisfies $k \cdot Q = \pm G$", where Q is the public key; the bound 2^{16} matches the default of the lint that zlint added for the incident, the check costs the same as the draft's, and the probability of rejecting an honestly generated key is below 2^{-366}. In Lean 4 (core library; axioms propext and Quot.sound only; no sorry) we prove, on an abstract prime-order curve group, that the proposed scalar clause holds exactly for the keys an x-only search of that size recovers, that the draft clause is a strict special case, and that the inverse clause is needed; the P-384 instances are checked by kernel arithmetic. We are careful about what is new: the negation family is already handled by the zlint lint (e_ecdsa_small_private_key, merged 2026-10-25), whose authors evidently know it; what is new is the observation that the ballot text, which is what a CA will be audited against, does not say so, and the inverse family, which to our knowledge no current linter or text covers.

1. The text under test

TLS Baseline Requirements v2.3.1 (2026-10-04), Section 6.1.6, today only recommends ECDSA public-key validation (the NIST SP 800-56A routines). Pull request #683 ("Reorganize key quality checking and add ECDSA small private scalar check", Aaron Gable, commit 844896cd, 2026-10-22; posted to the Server Certificate Working Group list on 2026-10-23 and on the agenda of the 2026-11-05 call as a draft under consideration) rewrites the section and adds, under "For ECDSA public keys, the CA MUST":

> Effective 2027-01-15, ensure that the public key does not correspond to a Private Key whose private scalar is less than 256.

The ballot's summary says the requirement "would have prevented the recent compromise of 8 ECDSA Subordinate CA Certificate Private Keys whose private scalars all fit in one byte each". The sentence is a MUST; after the effective date it is an audit criterion, and a CA that implements it literally, by computing kG for k = 1..255 and comparing with the public key, has complied.

2. Counterexamples

Let the curve have prime order n and base point G, and let d be a private scalar with public key Q = dG. Write x(P) for the x-coordinate of a point P. Two facts about affine Weierstrass curves are used: (n − d)G = −(dG), and −P has the same x-coordinate as P (and is the only other point with it).

A. Negation. Take k in 1..255 and d = n − k. Then $d \geq n - 255 > 2^{383}$, so "d is less than 256" is false and the draft rule accepts the key. But Q = (n − k)G = −kG, so x(Q) = x(kG). The search that recovers the eMudhra keys, "compare x(Q) with x(kG) for k = 1..255", returns k; the private key is then n − k. The cost is identical. (A CA that compares whole points rather than x-coordinates recovers nothing here, which is exactly the implementation the sentence invites.)

B. Inverse. Take k in 1..255 and $d = k^{-1} \bmod n$. Again $d > 2^{383}$ (for P-384 and k = 4, d = (n+1)/4) and the draft rule accepts the key. But $kQ = k \cdot k^{-1} \cdot G = G$, so computing Q, 2Q, 3Q, ... and comparing with G finds k after at most 255 additions; the private key is $k^{-1} \bmod n$. The same holds for $-k^{-1}$ (kQ = −G). Neither the draft sentence, nor the scalar form of our fix, nor any linter we examined covers this family.

Verification against the real keys. `verify.py` (Python 3, no dependencies, about two seconds) embeds the eight eMudhra public keys, taken from the certificates in the public proof-of-compromise bundle; their SHA-256 fingerprints match the Bugzilla report. It recovers the eight scalars by the 255-step x-only search ({4, 56, 66, 115, 155, 156, 230, 237}, as stated in Bugzilla 2083449 comment 634), and for each k builds d = n − k and $d = k^{-1} \bmod n$, checks that the draft rule accepts both, that x((n−k)G) = x(kG), that $k \cdot (k^{-1}G) = G$, and that each is recovered in at most 255 group operations. Every number in this note is a line of that output.

3. Correction

Replace the sentence by:

> Effective 2027-01-15, ensure that neither the private scalar d nor n − d is less than 2^{16}, where n is the order of the curve, and that no integer k less than 2^{16} satisfies $k \cdot Q = \pm G$, where Q is the Subject Public Key and G the base point.

Remarks. (i) The bound 2^{16} is the default of zlint's `e_ecdsa_small_private_key` (configurable up to 2^{32}) and costs 2^{16} point additions per key for each clause with a table, or a baby-step giant-step of about 2^8 steps; 256 is the smallest bound that covers the incident, not a security margin. (ii) The three clauses together reject at most $4 \cdot (2^{16} - 1)$ of the n − 1 scalars, so an honestly generated key is rejected with probability below $2^{18}/2^{384} = 2^{-366}$. (iii) The sentence names the check, not a tool, so a CA can implement it from the text; a sentence of the form "using a tool such as ..." would be an acceptable alternative provided the tool's coverage (including both families) is stated. (iv) The general principle is that a weak-key rule should be stated in terms of what an attacker can enumerate, not in terms of the integer value of d; the two families here are the ones with the same cost as the draft's own check.

4. What is already known, and what is not

The attack class "small scalar, up to sign" is known to the zlint maintainers: the lint added in response to the incident (zlint PR #1117, merged 2026-10-25, community source) searches d·G = ±Q by baby-step giant-step and its description says "or the negation of such a d". Google's paranoid_crypto also finds structured scalars (low or high 32 bits, repeated words) by batch discrete logarithms. We claim no novelty for the negation family as an attack. The two points we add are: first, the ballot text is what will be audited, and it does not say "or n − d"; neither the PR, its review comments, the working-group thread, nor the Bugzilla discussion mentions the gap (we checked each as of 2026-11-05). Second, the inverse family, k·Q = ±G, is covered by none of the ballot text, zlint's lint (which searches for d·G = ±Q only), paranoid_crypto's forms, badkeys (blocklist only for EC keys) or pkilint, and costs the attacker no more than the draft's check. We do not claim that any flawed generator is known to produce inverse scalars; the point is that a rule written as a set of integers admits keys as cheap to break as the ones it bans, and that the correction closes exactly that set at the same cost.

5. Lean 4

`proof/Weak.lean` (Lean 4.33.1, core library only) models the rule on an abstract prime-order curve group. The structure `CurveModel` carries as hypotheses only the two facts of Section 2 (scalars act mod n with (n − d)G = −(dG); negation preserves x and is the only way two points share an x-coordinate) together with injectivity of d ↦ dG on 0..n−1 and k·(dG) = (kd)G. Over this model:

- `gap`: for every k in 1..255 with 256 + k ≤ n, n − k is not rejected by the draft rule and is recovered by the x-only search over 1..255.
- `fix_iff`: for 0 < d < n and B ≤ n, "d < B or n − d < B" holds if and only if some k in 1..B−1 has x(dG) = x(kG). So the scalar clause of the correction says exactly what the x-only search tests; `draft_le_fix` shows the draft rule is a special case.
- `inv_recoverable_of_mul`: k·d ≡ ±1 (mod n) with 0 < k < B implies k·(dG) = ±G, the inverse family.
- For P-384 by kernel arithmetic: n − 4 and 4^{-1} = (n+1)/4 are both not "less than 256"; the scalar clause with B = 2^{16} rejects n − 4 but not 4^{-1}, which is why the inverse clause is separate.
- `Toy.model`: a cyclic group of order 23 satisfies the hypotheses (finite parts by `decide`), so the model is not vacuous.

The curve arithmetic itself is not formalized; the two hypotheses are standard and are exercised numerically on P-384 by `verify.py`. The file compiles with Lean 4.33.1 in under a second; `#print axioms` reports `propext` and `Quot.sound` only (the P-384 numeric facts depend on no axiom), and no `sorry` or `Classical.choice` appears.

6. Scope and limitations

This note concerns one sentence of a draft ballot that has not yet been proposed for a vote; the text may change before or during discussion. We make no claim about key generators, HSMs or

the eMudhra root cause. We make no claim that the remaining text of PR #683 (the RSA checks, the Debian weak-key clause, the catch-all for "demonstrated or proven methods") is wrong. The bound 2^{16} is a proposal; any bound that a CA can afford works with the same wording.

7. Reproduction and artifacts

- `verify.py`: `python3 verify.py` prints Sections 1-4 of the checks above and exits 0 only if all pass.
- `proof/Weak.lean`: `lean Weak.lean` (Lean 4.33.1); the only output is the `#print axioms` lines.
- Sources verified against: cabforum/servercert `docs/BR.md` v2.3.1 (commit 3574ae8) and PR #683 at commit 844896cd (2026-10-22); zlint commit 2f53a6a (2026-11-01); Bugzilla 2083449 as of 2026-11-05.

Disclosure

The analysis, code and proof were produced with substantial assistance from an AI system (Claude, Anthropic), directed and checked by the author. The author chose the target, decided the claims, ran the code and is responsible for the content. The keys examined are already public and revoked; the constructed keys are synthetic and not associated with any certificate. No CA, working group member or researcher was contacted in preparing this note.

References

- CA/Browser Forum, "Baseline Requirements for the Issuance and Management of Publicly-Trusted TLS Server Certificates", v2.3.1, 2026-10-04. https://github.com/cabforum/servercert/blob/main/doc
- A. Gable, "SC-XX: Reorganize key quality checking and add ECDSA small private scalar check", cabforum/servercert pull request #683, 2026-10-22. https://github.com/cabforum/servercert/pull/683
- Mozilla Bugzilla 2083449, "eMudhra: Private key compromise of Intermediate Certificates", opened 2026-10-18. https://bugzilla.mozilla.org/show_bug.cgi?id=2083449
- C. Henderson, "Add e_ecdsa_small_private_key", zmap/zlint pull request #1117, merged 2026-10-25. https://github.com/zmap/zlint/pull/1117
- Google, paranoid_crypto, `lib/ec_single_checks.py` (CheckWeakECPrivateKey). https://github.com/goog
- NIST, FIPS 186-5 / SP 800-186 (curve P-384); SEC 2 v2.0 (secp384r1).
- L. de Moura and S. Ullrich, "The Lean 4 Theorem Prover and Programming Language", CADE 2021.