Affine-Padding Rabin-Oracle Factoring in $L_n\left(\frac{1}{3}, \sqrt[3]{\frac{32}{9}}\right)$

Alexandra-Ioana Buzățoiu[1,2], Diana Maimuț[3,4], and David Naccache[1,5]

[1] ENS, CNRS, PSL Research University, Paris, France
david.naccache@ens.fr
[2] Military Technical Academy "Ferdinand I", Romania
ioana.buzatoiu@mta.ro
[3] Advanced Technologies Institute
021101 Bucharest, Romania
maimut.diana@dcti.ro
[4] University of Science and Technology POLITEHNICA Bucharest
060042 Bucharest, Romania
[5] Fakultet inženjerskih nauka, Univerziteta u Kragujevcu
6 Sestre Janjić, 34000 Kragujevac, Serbia
david@kg.ac.rs

Abstract. The 2007 algorithm of Joux, Naccache, and Thomé (JNT) shows that, with subexponential access to an e-th root oracle, one can forge RSA signatures for *odd* e in time close to the special number field sieve, without factoring the modulus; a recent JNT implementation by Shea et al. [2] carried that attack to 1024-bit RSA. We revisit the JNT construction at $e = 2$, the Rabin case, where an e-th root oracle is a square-root oracle. A *raw* square-root oracle factors $n = pq$ in one query, so it is of no interest; the interesting object is a *redundancy-constrained* (affine-padded) Rabin oracle that returns a single canonical root and thereby resists the one-query attack. We show that a one-sided number field sieve against such an oracle produces a congruence of squares $X^2 \equiv Y^2 \pmod{n}$ with $X \not\equiv \pm Y$ with probability $\frac{1}{2}$ per dependency, and hence *factors* n in special-number-field-sieve time

$$L_n\left(\frac{1}{3},\ \sqrt[3]{\frac{32}{9}}\right) \simeq L_n(1/3,\ 1.526\ldots)\,,$$

strictly below the general number field sieve's

$$L_n\left(\frac{1}{3},\ \sqrt[3]{\frac{64}{9}}\right) \simeq L_n(1/3,\ 1.923\ldots)\,.$$

The mechanism is a pleasant inversion: the sign ambiguity that the odd-e algorithm must suppress becomes, at $e = 2$, the very quantity that *leaks the factorization*. We give the algorithm in full — LLL polynomial selection for a quadratic field, a line sieve producing prime-ideal relations, quadratic characters, $\mathbb{F}_2$ linear algebra, and an exact number-field square root — prove the half-rate

splitting, and analyse the complexity, including the precise reason the attack collapses when the padding is a hash rather than an affine function of an attacker-chosen message. An open-source implementation realises every step. A small toy example run of it, from the padded modulus to the recovered factors, is given in Appendix A. Our implementation confirms a clear performance improvement in factoring speed.

Keywords: RSA · Rabin · Factoring · Number field sieve · Blind signatures · Delayed-target problems · Padding oracles

1 Introduction

It has been known since the invention of RSA that computing e-th roots modulo an integer $n = pq$ of unknown factorization need not be as hard as factoring n. For *odd* e, Joux, Naccache, and Thomé (JNT) [1] made this quantitative in a strong sense: given temporary, subexponential access to an e-th root oracle for a fixed public key (n, e), an attacker can afterwards forge the signature (or decrypt the ciphertext) of an arbitrary target, offline, in time close to that of the *special* number field sieve $L_n\left(1/3,\ \sqrt[3]{32/9} + o(1)\right)$, well below the $L_n\left(1/3,\ \sqrt[3]{64/9} + o(1)\right)$ that factoring n with the *general* number field sieve would cost. The setting — an oracle available only during a precomputation phase, then a challenge target — is the *delayed-target* problem in the terminology of Koblitz and Menezes [4], and a recent large-scale computation has carried the odd-e attack to 1024-bit RSA [2].

The present paper looks at the boundary case $e = 2$, where the modulus is a *Rabin* modulus [3] and an e-th root oracle is a square-root oracle. Two facts make this case special and, we think, instructive.

- A *raw* square-root oracle — one that will root any quadratic residue — factors n with probability $\frac{1}{2}$ per query, hence after two queries in expectation, because a random square is returned with a root other than $\pm r$ half of the time. So at $e = 2$ the entire number field machinery is pointless *unless* the oracle is constrained.
- Under such a constraint the odd-e algorithm inverts in a satisfying way. For odd e one takes pains to suppress a sign ambiguity in the number-field root; at $e = 2$ that very ambiguity is an element of the 2-torsion of $\mathbb{Z}_n^*$, and *that is the factorization*. The obstruction becomes the payload.

We study the natural constrained oracle — natural, yet never used in practice: a *redundancy-padded* Rabin signer that roots only values carrying a fixed affine pad and returns a single canonical root. This is exactly the affine-modular-root oracle of [1], specialised to $e = 2$. We show that a one-sided number field sieve against it yields, per $\mathbb{F}_2$ dependency, a congruence of squares that splits n with probability $\frac{1}{2}$, and hence *factors* n in time $L_n\left(1/3,\ \sqrt[3]{32/9} + o(1)\right)$ — SNFS speed, below GNFS — the deliverable now being the factorization itself rather than a forgery.

Contributions.

1. We isolate the redundancy-constrained square-root oracle AMR_2 (Section 2) and observe that it resists the trivial one-query attack yet falls to one-sided sieving.
2. We give the $e = 2$ number field sieve in full (Section 3): LLL polynomial selection for a quadratic field with $\alpha \mapsto c \bmod n$, a line sieve producing prime-ideal relations, quadratic characters, $\mathbb{F}_2$ linear algebra, and a number-field square root recovered exactly, by trace-and-norm in even degree and by Couveignes' method in odd degree.
3. We prove the half-rate splitting under the standard independence heuristic, and analyse the complexity: $L_n\Big(1/3,\ \sqrt[3]{32/9} + o(1)\Big)$, attained because factoring requires *no descent* (Section 4).
4. We report an open-source implementation. Appendix A exhibits a complete run of it.
5. We explain precisely why a *hashed* pad (PKCS#1 v1.5 [30]) defeats the attack even though it is affine in form (Section 6).

The contribution is not a new asymptotic record — the ingredients are classical and the mechanism is a JNT variant — but a clean oracle model in which a sub-GNFS *factoring* attack exists, together with an exact, runnable witness that every step composes.

2 Preliminaries

Throughout, $n = pq$ is a *Blum integer*: $p \equiv q \equiv 3 \pmod 4$. Then -1 is a quadratic non-residue modulo p and modulo q, squaring is a 4-to-1 map on $\mathbb{Z}_n^*$, and each quadratic residue has four square roots $\{\pm r, \pm r'\}$ with $r' \equiv r \pmod p$ and $r' \equiv -r \pmod q$. We write $L_n(1/3,\, c) = \exp\big((c + o(1))\, \sqrt[3]{(\ln n)(\ln \ln n)^2}\big)$.

Definition 1 (Affine modular-root oracle AMR_2). *Fix a public pad constant $c \in \mathbb{Z}_n$ and a message space $\mathcal{M} \subseteq \{0, \dots, M-1\}$ with $M \ll n$ (the letter B is reserved for the smoothness bound). A redundancy-padded Rabin signer is the deterministic oracle*

$$\mathcal{O}(\mu) = \begin{cases} \mathsf{canon}\big(\sqrt{c+\mu} \bmod n\big), & c+\mu \in (\mathbb{Z}_n^*)^2, \\ \bot, & \text{otherwise,} \end{cases} \qquad \mu \in \mathcal{M},$$

where canon *selects one distinguished root (the "redundancy pick": the unique root carrying a valid padding pattern, or — in our implementation — the numerically smallest root). Inputs outside $c + \mathcal{M}$ are refused.*

The affine shape $c + \mu$ with μ small, contiguous and attacker-chosen is that of a raw or blind signer under a fixed redundancy frame. It is a model rather than a deployed encoding: as Appendix C sets out, the standardised paddings either hash the message or scatter it across digit positions, and either is enough to defeat the sieve. What matters is not that the input be padded but that the attacker can *steer* μ over an interval; we return to this in Section 6. That the constraint is necessary at all is the content of the next observation.

Algorithm 1 Affine-Rabin factoring by a one-sided NFS

1: **Input:** Blum modulus $n = pq$, pad c, oracle $\mathcal{O}$, bounds B, S.
2: **Polynomial selection.** By LLL [27] find monic f of degree d with small coefficients and $f(c) \equiv 0 \pmod{n}$; set $K = \mathbb{Q}(\alpha)$, $f(\alpha) = 0$.
3: **Line sieve.** Over an interval of $2S$ consecutive $x \in \mathcal{M}$ collect relations $(\alpha + x)$ whose algebraic norm $|f(-x)|$ is B-smooth; factor each into prime ideals (p, r).
4: **Queries.** For each relation obtain $s_x = \mathcal{O}(x)$, so $s_x^2 \equiv c + x$; keep the relation iff $c + x$ is a residue (the oracle answers).
5: **Linear algebra.** Over $\mathbb{F}_2$, with quadratic-character columns, find a subset S^* with $\prod_{x \in S^*} (\alpha + x) = \beta^2$ a square in K.
6: **Number-field root.** Compute $\beta \in K$ exactly (Section 3.4); set $X = \varphi(\beta) = \beta(c) \bmod n$.
7: **Congruence.** $Y \leftarrow \prod_{x \in S^*} s_x \bmod n$; then $X^2 \equiv Y^2 \pmod{n}$. If $\gcd(X \mp Y, n)$ is nontrivial, **return** it; else try the next subset.

Proposition 2 (Why a raw oracle is trivial). *An unconstrained square-root oracle splits n with probability $\frac{1}{2}$ per query, hence after two queries in expectation. Choosing $r \in \mathbb{Z}_n^*$ at random and querying r^2, one receives $s \equiv \varepsilon r$ with ε in the two-torsion subgroup T, of order four; $\varepsilon = 1$ gives $\gcd(s - r, n) = n$, $\varepsilon = -1$ gives 1, and the remaining two values split. Consequently a* non-leaking *deterministic square-root oracle must restrict its domain; the affine restriction of Definition 1 is the natural such choice, and the one that fixed-pattern padding realises.*

The *delayed-target* game is as in [1,4]: the oracle is available during a precomputation phase depending only on (n, c); then a challenge is posed and the oracle withdrawn. For $e = 2$ the challenge is trivial (we factor n), so the "target" is n itself.

3 The Algorithm

We run a one-sided number field sieve in which the oracle supplies the second side. Let $K = \mathbb{Q}(\alpha)$ be the number field defined by a monic polynomial f with $f(c) \equiv 0 \pmod{n}$, and let $\varphi\colon \mathbb{Z}[\alpha] \to \mathbb{Z}_n$, $\alpha \mapsto c$, be the induced ring homomorphism (well defined because $f(c) \equiv 0$). The rational image of $\alpha + x$ is $\varphi(\alpha + x) = c + x$, which the oracle roots. Algorithm 1 summarises the attack; the subsections detail each step, at the degree $d = 2$ used in the implementation (the optimal degree grows only slowly with n; see Section 4).

3.1 Polynomial Selection

For $d = 2$ we seek a monic $f(x) = x^2 + a_1 x + a_0$ with small a_0, a_1 and $f(c) \equiv 0 \pmod{n}$, i.e. $a_0 + a_1 c \equiv -c^2 \pmod{n}$. The integer solutions of the homogeneous

congruence form a rank-2 lattice of index n,

$$\Lambda = \big\{(a_0, a_1) \in \mathbb{Z}^2 : a_0 + a_1 c \equiv 0 \pmod{n}\big\}, \qquad \Lambda = \big\langle (n, 0),\ (-c \bmod n,\ 1)\big\rangle,$$

The reduction must be *skewed*, and this is not an optimisation but a requirement. What controls the norm over $x \in [-S, S]$ is not $\max_i |a_i|$ but $\max_i |a_i|\sigma^i$ for a skew σ matched to the sieve interval, so we LLL-reduce Λ in the coordinates scaled by $(1, \sigma, \ldots, \sigma^{d-1})$ and round a particular solution against that basis; the implementation takes $\sigma = n^{1/2d}$. For $d = 2$ the weighted lattice has determinant $n\sigma$, so $|a_0| \lesssim \sqrt{n\sigma}$ and $|a_1| \lesssim \sqrt{n/\sigma}$, and

$$|f(-x)| = |x^2 - a_1 x + a_0| \ \lesssim\ \max\big(S^2,\ \sqrt{nS}\big) \qquad (\sigma \asymp S).$$

An *unweighted* reduction would instead return $|a_0|, |a_1| \simeq \sqrt{n}$ and a norm of order $\sqrt{n}\,S$, larger by $\sqrt{S}$; Remark 7 shows that this single factor is exactly the difference between the constant of Theorem 5 and that of GNFS. The homomorphism is $\alpha \mapsto c$ and $K = \mathbb{Q}(\alpha)$ is imaginary or real quadratic according to the sign of $\operatorname{disc}(f) = a_1^2 - 4a_0$.

3.2 One-Sided Line Sieve and Prime-Ideal Relations

The factor base is the set of degree-one prime ideals of norm below B, i.e. pairs (p, r) with $p < B$ prime, $p \nmid \operatorname{disc}(f)$ and $f(r) \equiv 0 \pmod{p}$. A prime (p, r) divides $(\alpha + x)$ exactly when $x \equiv -r \pmod{p}$. We therefore line-sieve over an interval of $2S$ consecutive elements of $\mathcal{M}$; since the oracle answers only on $c + \mu$ with $\mu \in \mathcal{M}$, that interval must lie *inside* $\mathcal{M}$, and we write it $[-S, S]$ throughout, having absorbed its midpoint into the pad constant c — which is legitimate because $c + x_0 + x$ is again an affine function of the message. Over that interval we accumulate $\log p$ at every $x \equiv -r \pmod{p}$, and retain the x whose accumulated logarithm is close to $\log |f(-x)|$. Each survivor is confirmed by factoring the single integer $|f(-x)|$ over the factor base and reading off the ideal valuations; unlike the two-sided sieve of GNFS, *only the algebraic side is sieved* — the rational side $c + x$ carries no smoothness condition because the oracle will root it. This one-sidedness is the source of the SNFS-like complexity (Section 4).

3.3 Characters and Linear Algebra

Stacking the ideal valuations modulo 2 gives an $\mathbb{F}_2$ matrix whose null space encodes subsets S^* with all valuations even, i.e. $\prod_{x \in S^*} (\alpha + x)$ a square *ideal*.

To force a square *element* we adjoin a handful of quadratic-character columns $\chi_{(q,s)}(\alpha + x) = \left(\frac{x+s}{q}\right)$ for auxiliary primes (q, s), together with *one sign column per real place*: for a real quadratic K the norm $f(-x) = (\alpha_1 + x)(\alpha_2 + x)$ is the product of the two archimedean signs, so a single column recording $\operatorname{sign} f(-x)$ would pass a dependency negative at both places, which is not a square; we therefore record $\operatorname{sign}(\alpha_i + x)$ at each real embedding. For imaginary K there is no real place and the columns are absent. A null-space vector then yields, heuristically, a product that is a genuine square in K; because the number-field root of Section 3.4 verifies its output exactly, a false square is simply discarded and the next dependency tried, so soundness never depends on the completeness of the character set.

3.4 The Number-Field Square Root

Given S^* we must extract β with $\beta^2 = \Pi := \prod_{x \in S^*} (\alpha + x) \in \mathcal{O}_K$.

This is the step that decides whether the attack is practical, because a dependency contains $|S^*| = \Theta(|F|)$ factors: at cryptographic sizes this runs to 10^4 terms and Π has coefficients of order 10^6 bits.

The archimedean route of Montgomery [5], surveyed with its alternatives by Thomé [7] — take the complex embeddings $\Pi_j = \prod_x (\alpha_j + x)$, take complex square roots, solve the $d \times d$ Vandermonde system and round — is correct but needs a working precision of $\frac{1}{2}\log_2 |\Pi|$, and implemented naively — forming the embeddings by a sequential product at full precision — it costs more than the sieve and the linear algebra together. A careful implementation of [5] would not; we chose the exact routes because they need no precision analysis at all and certify their own output. We therefore use two exact routes, according to the parity of d, and never leave $\mathbb{Z}$. In both, Π is first formed *exactly* in $\mathbb{Z}[\alpha]$ by a balanced product tree, which keeps the integer multiplications balanced and costs $O(M(|\Pi|)\log|S^*|)$.

One caveat is standard and we adopt the standard remedy. A square root of an element of $\mathbb{Z}[\alpha]$ lies in $\mathcal{O}_K$ but need not lie in the order $\mathbb{Z}[\alpha]$, which may be strictly smaller; we therefore extract $\sqrt{f'(\alpha)^2\,\Pi}$ and divide by $f'(\alpha)$ at the end, so that the root sought is guaranteed to have integer coordinates in the power basis. Primes dividing $\mathrm{disc}(f)$ are excluded from the factor base (Section 3.2), and a relation whose norm is divisible by one is discarded rather than represented.

Degree two: trace and norm. For $d = 2$ write $f = Z^2 + f_1 Z + f_0$ and $\beta = u + v\alpha$. From $\beta^2 = \Pi$,

$$N(\beta)^2 = N(\Pi), \qquad \mathrm{Tr}(\beta)^2 = \mathrm{Tr}(\beta^2) + 2N(\beta) = \mathrm{Tr}(\Pi) + 2N(\beta),$$

and, writing $t = \mathrm{Tr}(\beta) = 2u - vf_1$ and $\nu = N(\beta) = u^2 - uvf_1 + v^2 f_0$, eliminating u gives $v^2\,(f_1^2 - 4f_0) = t^2 - 4\nu$. So β is recovered from three *integer* square roots — ν from $N(\Pi)$, t from $\mathrm{Tr}(\Pi) + 2\nu$, and v from $(t^2 - 4\nu)/\,\mathrm{disc}(f)$ — followed by $u = (t + vf_1)/2$. Each is a $\Theta(10^6)$-bit `isqrt`, which GMP does in milliseconds. The sign choices are enumerated (there are at most eight, most of them killed by a non-square or a non-division) and each surviving β is checked against $\beta^2 = \Pi$ in $\mathbb{Z}[\alpha]$.

Two symmetric functions determine β only when $d = 2$: for even $d \geq 4$ the trace and the norm leave $d - 2$ coordinates undetermined, and there the general route is Montgomery's method [5]. Since the optimal degree of Theorem 5 grows, the route below is the one that matters asymptotically.

Odd degree: Couveignes. For odd d we use Couveignes' method [6], which is exactly the case where it applies. Let P be a prime inert in K, so that $\mathcal{O}_K/P \cong \mathbb{F}_{P^d}$; reduce Π there and take its square root in the residue field. Of the two roots $\pm r$ only one can be $\beta \bmod P$, and the ambiguity is resolved *without any lifting*: since d is odd, $N(-r) = -N(r)$, so the correct root is the one whose residue-field norm matches

$N(\beta) = \sqrt{N(\Pi)}$, an integer we already have exactly. Collecting enough such P that

$$\prod P > 2 \max_k |\beta_k|$$

and combining the residues by a divide-and-conquer CRT returns β; choosing $P \equiv 3 \pmod 4$ makes each residue-field root a single exponentiation, since then $P^d \equiv 3 \pmod 4$. The inert primes depend only on f, so they are found once and reused across dependencies. For $d = 3$ irreducibility mod P reduces to the single test $\gcd(Z^P - Z, f) = 1$, since a reducible cubic must have a root; for $d \geq 5$ that test is not sufficient — a quintic can split as $2 + 3$ with no root — and one needs the full distinct-degree factorisation. The procedure is self-certifying: the reconstructed β is accepted only after $\beta^2 = \Pi$ is verified exactly in $\mathbb{Z}[\alpha]$, so no bound on $|\beta|$ need be proved — an insufficient set of primes simply fails the check and more are added.
Finally, $X = \varphi(\beta) = \big(\sum_k \beta_k c^k\big) \bmod n$.

3.5 Congruence of Squares

By construction $X^2 \equiv \varphi(\Pi) = \prod_{x \in S^*} (c + x) \pmod n$, and the oracle products give a second root $Y = \prod_{x \in S^*} s_x$ of the same value, so $X^2 \equiv Y^2 \pmod n$.
A nontrivial $\gcd(X - Y, n)$ (or $\gcd(X + Y, n)$) splits n. The next section shows this happens with probability $\frac{1}{2}$ per dependency, so a split is found after $O(1)$ of them.

4 Analysis

4.1 Half-Rate Splitting

Proposition 3. *Fix a dependency S^* and let X, Y be as in Section 3.5, so $\varepsilon := YX^{-1}$ lies in the two-torsion $T \cong \{\pm 1\}^2$ under $\mathbb{Z}_n^* \cong \mathbb{Z}_p^* \times \mathbb{Z}_q^*$. Modelling the CRT sign pattern $(\sigma_{x,p}, \sigma_{x,q}) \in \{\pm 1\}^2$ of each canonical oracle root s_x, relative to the number-field root's branch, as independent and uniform on $\{\pm 1\}^2$, the element ε is uniform on T, and $\gcd(X - Y, n)$ is a nontrivial factor with probability exactly $\frac{1}{2}$.*

Proof. Write the honest number-field root as $X \leftrightarrow (X_p, X_q)$, and let

$$R = \prod_{x \in S^*} (c + x), \qquad \sqrt{R}_p = \tau_p X_p, \qquad \sqrt{R}_q = \tau_q X_q,$$

recording its branch $\tau_p, \tau_q \in \{\pm 1\}$ (the letter P is reserved for the inert primes of Section 3.4). Each oracle root contributes $(\sigma_{x,p}, \sigma_{x,q})$, so $\varepsilon_p = \Sigma_p \tau_p$ and $\varepsilon_q = \Sigma_q \tau_q$ with $\Sigma_p = \prod_x \sigma_{x,p}$, $\Sigma_q = \prod_x \sigma_{x,q}$.
For ± 1 variables each Σ is uniform, and $\mathbb{E}[\Sigma_p \Sigma_q] = \prod_x \mathbb{E}[\sigma_{x,p}\sigma_{x,q}] = 0 = \mathbb{E}[\Sigma_p]\,\mathbb{E}[\Sigma_q]$, so (Σ_p, Σ_q) is jointly uniform on $\{\pm 1\}^2$ for any nonempty S^*. Multiplying by the fixed (τ_p, τ_q) permutes T; hence $(\varepsilon_p, \varepsilon_q)$ is uniform on T. The gcd splits iff $\varepsilon_p \neq \varepsilon_q$, of probability $\frac{1}{2}$. □

Remark 4 (Which canonicalisations Proposition 3 covers). The independence model is a hypothesis about canon, and not every choice satisfies it. For a Blum integer $|(\mathbb{Z}_n^*)^2| = \frac{p-1}{2} \cdot \frac{q-1}{2}$ is odd, so squaring is an automorphism of the group of quadratic residues and the *principal-root* map — its inverse — is multiplicative. A signer returning principal roots would therefore give $\prod_x \mathsf{canon}(c+x) = \mathsf{canon}\big(\prod_x (c+x)\big)$, every $\sigma_{x,\cdot}$ would equal $+1$, and the proof above would be vacuous: splitting would then hinge on the number-field branch alone and need a different heuristic. Proposition 3 applies to a canon that is *not* a homomorphism on $(\mathbb{Z}_n^*)^2$ — the smallest-root and fixed-pattern picks used here and in ISO 9796-1 are of that kind — and the Rabin–Williams principal-root variant is precisely the exception.

The quadratic characters of Section 3.3 force a square *in the number field* but say nothing about the $\mathbb{Z}_n$ signs, which remain free; that freedom is exactly what survives to leak p, q. This is the mirror image of odd e, where the root is unique in the residue field and the ambiguity can — and must — be avoided.

4.2 Complexity

Theorem 5. *Against an AMR$_2$ oracle whose pad is affine in an attacker-chosen message and whose message space satisfies $|\mathcal{M}| \geq L_n\Big(1/3,\ \sqrt[3]{32/9} + o(1)\Big)$ — the minimum of Proposition 9, without which no sieve range exists — the algorithm factors n in time $L_n\Big(1/3,\ \sqrt[3]{32/9} + o(1)\Big) \simeq L_n(1/3,\ 1.526\ldots)$, using $L_n\Big(1/3,\ \sqrt[3]{4/9} + o(1)\Big) \simeq L_n(1/3,\ 0.763\ldots)$ oracle queries.*

Proof (Sketch). With degree $d = \delta \sqrt[3]{\ln n / \ln\ln n}$, sieve range $S = L_n(1/3,\ \theta b)$ and factor-base bound $B = L_n(1/3,\ b)$, the skew-weighted selection of Section 3.1 returns a *monic* f — its leading coefficient is 1 — whose weighted coefficients satisfy $\max_i |a_i|\sigma^i \simeq n^{1/d}\sigma^{(d-1)/2}$ with $\sigma \asymp S$. The norm over $x \in [-S, S]$ therefore has size $L_n(2/3, \gamma)$ with $\gamma = \frac{1}{\delta} + \frac{\theta}{2}\delta b$, and is B-smooth with probability $L_n(1/3,\ -\eta)$, $\eta = \gamma/(3b) = \frac{1}{3}(\frac{1}{\delta b} + \frac{\theta\delta}{2})$. Two conditions close the system: (i) enough survivors, $\theta b - \eta = b$; and (ii) sieving balanced against linear algebra, $L_n(1/3, \theta b) = L_n(1/3, 2b)$, i.e. $\theta = 2$. Optimising in δ yields $\delta = \sqrt[3]{3/2}$, $b = \sqrt[3]{4/9}$, and total cost $L_n(1/3,\ 2b) = L_n\Big(1/3,\ \sqrt[3]{32/9}\Big)$; the query count is $L_n(1/3,\ b) = L_n\Big(1/3,\ \sqrt[3]{4/9}\Big)$, subexponential and far below the run time. Crucially, factoring needs *no descent*: no external target is embedded, so the initialisation step whose cost inflates the arbitrary-target constant to $L_n\Big(1/3,\ \sqrt[3]{6}\Big)$ of the affine-oracle forgery is absent, and only one-sided relation collection plus an $\mathbb{F}_2$ dependency remain, exactly as in NFS factoring. □

Remark 6 (Why $\sqrt[3]{32/9}$ and not $\sqrt[3]{6}$). It is worth separating two AMR results of [1], since they are easily conflated. JNT state, on the one hand, that one may *selectively* produce new affine modular roots in $L_n\Big(1/3,\ \sqrt[3]{32/9}\Big)$ — matching SNFS — and, on the other, that computing an *arbitrary* e-th root from an AMR oracle costs $L_n\Big(1/3,\ \sqrt[3]{6}\Big) \simeq$

$L_n(1/3, 1.817\ldots)$. The $\sqrt[3]{6}$ penalty is therefore the price of embedding an *externally supplied* target under the AMR constraint, not the price of one-sided relation collection. Factoring supplies no external target: any $\mathbb{F}_2$ dependency whatsoever yields a congruence of squares, so the attack of this paper lies in the *selective* regime and attains $\sqrt[3]{32/9}$. This is precisely what the derivation above shows, and it is consistent with the remark of [2] that JNT "does not achieve SNFS complexity when the oracle is restricted to AMR queries", which concerns the arbitrary-target forgery that is their subject, not the selective setting.

Remark 7 (Why the skew is load-bearing). Repeating the optimisation for an *unweighted* selection — balanced coefficients $|a_i| \simeq n^{1/d}$, hence a norm of size $L_n(2/3, \frac{1}{\delta} + \theta\delta b)$ and $\eta = \frac{1}{3}(\frac{1}{\delta b} + \theta\delta)$ — gives $3\delta b^2 - 2\delta^2 b - 1 = 0$ with $\partial b/\partial\delta = 0$, so $b = \frac{4}{3}\delta$, $\delta^3 = \frac{3}{8}$, $b = \sqrt[3]{8/9}$ and $2b = \sqrt[3]{64/9}$: exactly the GNFS constant, and no advantage whatsoever. The whole margin of Theorem 5 is the factor $\sqrt{S}$ that the skew-weighted reduction of Section 3.1 removes from the norm.

Remark 8 (Optimality within this framework). It is natural to ask whether the constant can be pushed further. Repeating the derivation of Theorem 5 with its three structural quantities left free — ρ, the fraction of the norm that must be smooth; w, the weight the sieve range carries in that norm; and κ, the number of simultaneous root conditions imposed on f — the whole optimisation closes in one line. With $\gamma = \kappa/\delta + w\theta\delta b$ and $\eta = \rho\gamma/(3b)$, the two conditions $\theta b - \eta = b$ and $\theta = 2$ give $3\delta b^2 - 2w\rho\delta^2 b - \rho\kappa = 0$, and $\partial b/\partial\delta = 0$ yields

$$\delta^3 = \frac{3\kappa}{8w^2\rho}, \qquad b\,\delta^2 = \frac{\kappa}{2w}, \qquad (2b)^3 = \frac{64\,w\,\rho^2\kappa}{9}.$$

Theorem 5 is $(w, \rho, \kappa) = (\frac{1}{2}, 1, 1)$ and Remark 7 is $(1, 1, 1)$. Each of the three is pinned, and by a different mechanism.

$\rho = 1$, *by the oracle's interface.* The rational value $c + x$ is of the size of n — it is $L_n(1, 1)$, not $L_n(2/3, \cdot)$ — so no constant fraction of it can be sieved at all, and the oracle is not performing half of a two-sided descent but something no sieve of any size could do. In the other direction the oracle accepts only inputs of the form $c + \mu$, so it cannot be handed a cofactor of the algebraic norm. Hence ρ is not a tunable fraction: it is 1, or there is no attack.

$w \geq \frac{1}{2}$, *by the lattice determinant.* The selection lattice $\{a : \sum a_i c^i \equiv 0 \pmod n\}$ has determinant n; weighting coordinate i by σ^i makes it $n\sigma^{d(d-1)/2}$ in dimension d, so a reduced vector satisfies $|a_i|\sigma^i \lesssim n^{1/d}\sigma^{(d-1)/2}$, the exponent being the *mean* of $0, \ldots, d-1$. Releasing the leading coefficient does not help: the lattice becomes $(d+1)$-dimensional with determinant $n\sigma^{d(d+1)/2}$, giving $n^{1/(d+1)}\sigma^{d/2}$ — the mean of $0, \ldots, d$, hence the same $w = \frac{1}{2}$ up to $o(1)$. The identity above says the same thing from the other side: monicity requires the leading term S^d to be dominated by $n^{1/d}S^{d/2}$, which is exactly $b\delta^2 \leq 1$, i.e. $w \geq \frac{1}{2}$. At the optimum $b\delta^2 = 1$ holds with equality, so the leading term has precisely caught up and there is no slack to trade.

$\theta = 2$, *by the linear algebra.* The matrix has $L_n(1/3, b)$ columns and its nullspace costs $L_n(1/3, 2b)$, so the sieve is balanced against it at $\theta = 2$ and the total cannot fall below $L_n(1/3, 2b)$ whatever θ is chosen.

Two further routes close for structural reasons. The affine padding pins the sieved element to $\alpha + x$: a two-dimensional $a - b\alpha$ would need $(b-1)c \bmod n$ to be small, which forces $b \asymp n/S$ and an astronomical norm. A one-dimensional sieve over S points carries norms S^d where a two-dimensional one carries $S^{d/2}$, so the oracle's interface costs exactly the factor that the skew of Section 3.1 buys back — which is what $w = \frac{1}{2}$ *is*. And the multiple-number-field trick that improves GNFS slightly gains nothing here: it amortises a shared *expensive* side over several algebraic ones, whereas our shared side is the oracle and is free, so k polynomials give k times the relations and k times the columns and cancel exactly.

The result sits strictly below GNFS, $\sqrt[3]{32/9} \simeq 1.526\ldots < 1.923\ldots \simeq \sqrt[3]{64/9}$: the gap is the whole SNFS margin, but the deliverable is the factorization of n, obtained without ever computing a private key.

4.3 The Message Slot Is the Sieve Range

The affine oracle answers only on $c + \mu$ with $\mu \in \mathcal{M}$, so the sieve of Section 3.2 may only examine $x \in \mathcal{M}$: the message slot *is* the sieve range. This makes the padding length a hard structural parameter rather than a tuning knob, and gives the following bound.

Proposition 9 (Minimum message slot). *With the parameters of Theorem 5 the sieve must range over* $S = L_n(1/3,\, 2b) = L_n\Big(1/3,\, \sqrt[3]{32/9}\Big)$ *values, so the attack requires*

$$\log_2 |\mathcal{M}| \;\gtrsim\; \frac{\sqrt[3]{\frac{32}{9}}}{\ln 2}\, \sqrt[3]{(\ln n)(\ln\ln n)^2},$$

i.e. a pad of at most $\log_2 n - \left\lceil \sqrt[3]{32/9}\, \sqrt[3]{(\ln n)(\ln\ln n)^2}/\ln 2 \right\rceil$ *bits. Below that size the attack fails outright.*

The essential point is not the constant but the *irreversibility*: a longer pad shrinks $\mathcal{M}$, and the lost relations cannot be bought back with more computation, because there are no further candidates to sieve. Each extra pad bit halves the available candidates. Lowering B reduces the number of columns to beat but lowers the smoothness probability at the same time, so it postpones the failure only marginally. Numerically the bound grows slowly: 19 bits of message at $\log_2 n = 64$, 27 at 128, 37 at 256 and 69 at 1024.

5 Experimental Evaluation

5.1 What to Compare Against, and Why

The claim of Section 4 is about an *exponent*. It says that collecting relations under one smoothness condition costs $L_n\Big(1/3,\, \sqrt[3]{32/9}\Big)$ where collecting them under two costs $L_n\Big(1/3,\, \sqrt[3]{64/9}\Big)$, and it says nothing whatever about constant factors. An experiment that timed our implementation against a mature general-purpose factoring program

would therefore measure the wrong thing. Such a program carries two decades of tuning — large primes, lattice sieving with special-q, batch cofactorisation, a filtering chain, block Wiedemann tuned to its matrices — none of which is in the exponent and all of which is in the constant. A fresh implementation begins tens of times behind on that account alone, and since the exponent gap buys only a factor $\exp\big((\sqrt[3]{64/9} - \sqrt[3]{32/9})\,\Delta\big)$ with Δ the increment of $(\ln n)^{1/3}(\ln\ln n)^{2/3}$ — about 1.3 per 24 bits in the range a single core can reach — a constant-factor handicap of that size is not repaid until several hundred bits beyond anything such a campaign could measure. A race lost for reasons the theorem does not address is not evidence against the theorem, and a race won for them would not be evidence for it.

The comparison that does test the claim is the algorithm against *itself*. We therefore implement the same number field sieve twice, in one program, differing in exactly one respect: whether the rational side is sieved.

One-sided. Algorithm 1: f monic of degree d with $f(c) \equiv 0 \pmod n$, and x ranging over the message slot. A relation is an x whose algebraic norm $f(-x) = \pm N(\alpha + x)$ is B-smooth. The rational value $c + x$ is never sieved — the $\textsc{amr}_2$ oracle returns its square root — which is the one-sided regime of Section 3.2.

Two-sided. The same sieve with the oracle removed. Since $c + x$ is of the size of n, no sieve can make it smooth, so dropping the oracle forces the classical setting: f monic of degree d with $f(m) \equiv 0 \pmod n$ for $m \sim n^{1/d}$, and coprime pairs (a, b) for which *both* $a - bm$ and $F(a, b) = \sum_i f_i a^i b^{d-i}$ are smooth. A dependency then gives $\prod(a - bm) = y^2$ in $\mathbb{Z}$ and $\prod(a - b\alpha) = \beta^2$ in $\mathbb{Z}[\alpha]$, so that $\varphi(\beta)^2 \equiv y^2 \pmod n$ under $\varphi : \alpha \mapsto m$, and $\gcd(\varphi(\beta) - y, n)$ splits n. This is the general number field sieve [26], at $\sqrt[3]{64/9}$.

The point of building the control this way is that almost nothing differs between the two runs. They share the LLL selection lattice of Section 3.1, pointed at c in one case and at m in the other; the root finding modulo the factor-base primes; the logarithmic sieve and the exact cofactoring of its survivors; the quadratic characters; the assembly of the $\mathbb{F}_2$ matrix; the nullspace routine; and the exact number-field square root of Section 3.4. The two-sided siever is the one-sided siever with a second array, a second threshold and a second resieve. Whatever our implementation does badly, it does badly on both sides and cancels; what does not cancel is the second smoothness condition, which is precisely the quantity under test.

5.2 Protocol

Instances. For each size $\log_2 n$ one modulus $n = pq$ is generated from a fixed seed with $p \equiv q \equiv 3 \pmod 4$, and the affine pad c is the decimal digits of π in binary-coded decimal, so that the pad is manifestly attacker-independent and every instance is reproducible from the seed alone. Both engines are given the same n.

The oracle. The AMR_2 oracle is simulated: it is handed p and q, computes the two square roots by exponentiation modulo each prime, recombines them by the Chinese remainder theorem, and returns the single canonical root fixed by the convention of Section 2, refusing any query that is not a residue. It returns one root and never the other three, so nothing in the measurement benefits from the ambiguity the attack is supposed to have to reconstruct.

One core each. Every measurement is a separate job pinned to a single CPU by `taskset`, with all thread pools fixed at one. A job that owns one core is sequential, so its wall-clock and its CPU time coincide and neither engine can be favoured by the scheduler or credited with a parallel speed-up it did not have. Jobs run several at a time, one per core. We report CPU seconds, and we also record each job's elapsed time, which makes a violation visible: a job whose sieve took longer in elapsed time than the job consumed in CPU time did not have its core to itself, and is flagged rather than quietly kept.

Verification. A returned pair counts only if it is an exact factorisation of n into two primes *and* matches the primes the modulus was built from. Anything else, including a run that exhausts its time budget, is recorded as a failure and never as a timing, so a curve can be short but never optimistic.

Parameters. Neither engine is hand-tuned per instance. Each starts from a rule in $\log_2 n$ alone — the smoothness bound, the degree and the sieve range — and, if the pass does not yield enough relations, doubles and repeats. The parameters that finally succeeded are recorded with the timing, so the tuning is visible in the data rather than hidden in it.

5.3 Results

Figure 1 is the raw measurement: CPU seconds against $\log_2 n$, on a logarithmic axis, for both engines.
Two curves on a logarithmic axis separating steadily is suggestive but not yet a measurement of an exponent. The quantity claimed is recovered by changing abscissa: a cost $L_n(1/3,\, c)$ is, by definition, $\exp\big(c\,(\ln n)^{1/3}(\ln\ln n)^{2/3}(1+o(1))\big)$, so plotting $\ln(\text{cost})$ against $(\ln n)^{1/3}(\ln\ln n)^{2/3}$ turns each engine into a straight line whose *slope is its constant*. Figure 2 is that plot, with the two least-squares lines.

What is measured without fitting anything. Two statements follow from Figure 1 directly, and neither depends on a regression. Over the range where both engines ran, 88–144 bits, removing the oracle multiplies the cost by between 21 and 327, and the multiplier grows with n. And at equal cost the two are 88 bits apart: the attack factors a 191-bit modulus in 65 core-seconds, which is what the control spends on 103 bits. Since the two runs are the same program on the same moduli, differing only in whether the rational side is sieved, that gap is the second smoothness condition and nothing else.

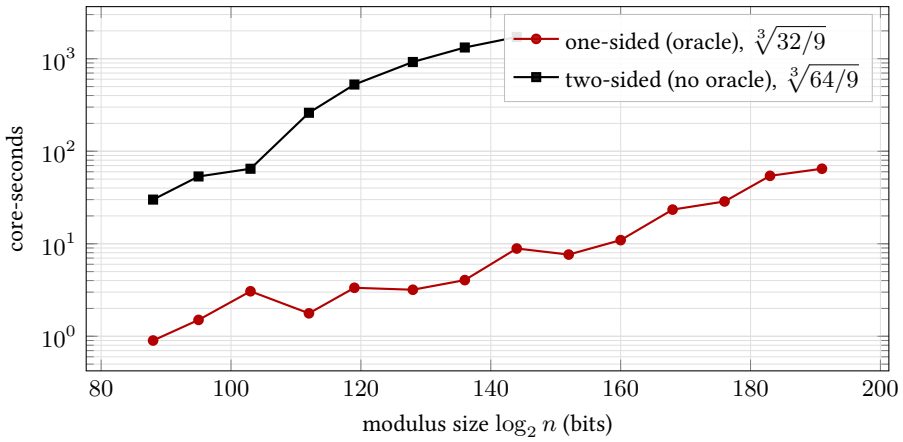

Fig. 1. Core-seconds against modulus size for the one-sided attack and for the same code with the oracle removed. Both engines run on one core, on the same moduli, and every point is a factorisation verified against the generated primes.

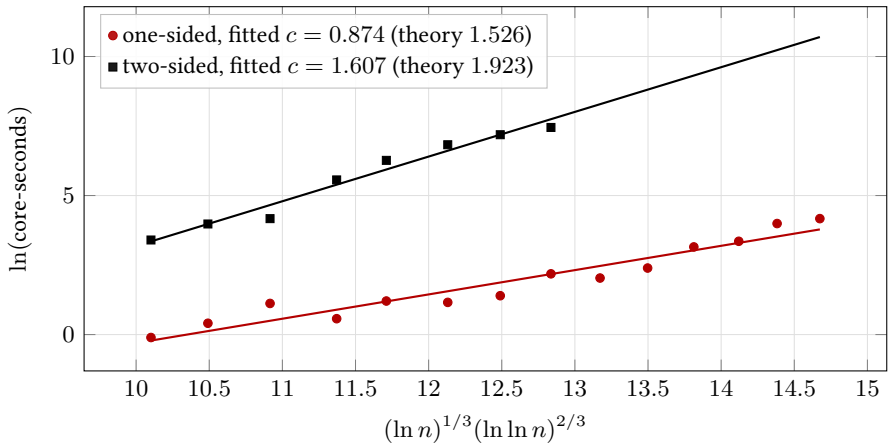

Fig. 2. The same measurements against $(\ln n)^{1/3}(\ln \ln n)^{2/3}$, the abscissa in which an $L_n(1/3, c)$ cost is a line of slope c. The fitted slopes are the measured exponent constants; the predictions are $\sqrt[3]{32/9} \simeq 1.526$ and $\sqrt[3]{64/9} \simeq 1.923$.

Reading the fit. The exponent itself is the harder quantity, and the honest report is that this campaign does not yet pin it. Fitted over all the data, the attack gives $c = 0.874$ ($R^2 = 0.929$, 14 points) against the control's $c = 1.607$ ($R^2 = 0.970$, 8 points), a separation of 0.732 where theory predicts 0.397. But the separation is not stable: refitting from each measured size upward, over 6 nested windows, it ranges from -0.33 to $+0.74$. A quantity whose sign depends on where the window starts has not been measured, and we prefer to say so rather than quote the window that agrees best.

Why the slope is the harder quantity here. Three effects, all in the implementation and none in the algorithm, dominate the ends of the range.

1. *The attack is too fast at the bottom.* Its cost at 88 bits is under a second, of which less than half is sieving; the rest is process start-up, polynomial selection and matrix assembly, which are constants in n. A constant added to both lines cancels in the separation, but it does not cancel within a single fit, and at the bottom of this range it flattens the attack's line towards zero. The sieve's share climbs from 44% at 88 bits to 95% at 191; only the upper part of the range measures the algorithm at all.
2. *The control is not sieving at the top.* Its relation collection is the whole cost up to 128 bits, but at 136 bits the number-field square root is 55% of the run and at 144 bits the linear algebra is 60%. The nullspace is dense Gaussian elimination over $\mathbb{F}_2$, cubic in the number of columns, whereas the analysis of Section 4 assumes a linear-algebra cost subordinate to relation collection. It is the same routine on both sides, but the control's matrices are roughly three times as wide, so it reaches them first. A cost with the wrong exponent, arriving on one side sooner than the other, is exactly what a slope measurement cannot absorb.
3. *Shared cores.* At three control sizes the sieve's elapsed time exceeds the job's own CPU time, which can only happen if the job did not have its core to itself. The campaign was run with more workers than the machine had physical cores, so those points are inflated by contention rather than by arithmetic.

The first is cured by measuring further up, which the attack makes cheap: it is still only a minute at 191 bits. The third is a scheduling setting. The second is structural and is the one that decides the experiment: a block Lanczos or block Wiedemann nullspace, whose cost is essentially that of the sparse matrix–vector products, would restore the assumption the analysis makes and let the control follow the attack up the range. Until then Figure 2 should be read as two straight lines with the predicted ordering, and not as a measurement of $\sqrt[3]{32/9}$ against $\sqrt[3]{64/9}$.

Where the sweep stops. The control arm stops at 144 bits and the attack's runs to 191, because that is where each stops being affordable: the single control run at 144 bits costs more than the attack's entire fourteen-point arm put together. That asymmetry is not an artefact of the schedule, it is the result. No arm is truncated on the basis of what it was about to show; the budget is fixed in advance and a job that exceeds it is reported as a failure, never as a timing.

Reproducing. The implementation is a notebook that writes out every source file it uses — the two sievers in C over GMP, and the Python driving them — builds them, and runs the sweep, checkpointing each measurement as it lands. There is no external solver to install and nothing is fetched at run time. The campaign's raw output is a single CSV, one row per modulus size, from which `mkbenchcsv.py` derives both figures of this section and every fitted number quoted in it.

6 Why a Hash Defeats the Attack

PKCS#1 v1.5 signature encoding is affine: the encoded message is `00 01 FF..FF 00` ‖ ASN.1 ‖ $H(m) = c + H(m)$ for a fixed constant c and a small variable $H(m)$. In form this is an AMR$_2$ oracle, and the affine-oracle variant of [1] applies at asymptotic constant $\sqrt[3]{6} \simeq 1.817\ldots < 1.923\ldots$ — below GNFS. Yet the attack does not run in practice, for a reason worth stating precisely. The sieve of Section 3.2 *steers* x: it chooses smooth values and queries the oracle on those. With $x = H(m)$ the attacker cannot steer, because hitting a chosen smooth x^* would require a preimage $H(m) = x^*$. It can only *sample*: draw m, keep $H(m)$ if the norm happens to be smooth. Two consequences follow, and only the second is decisive.

The first is a cost, and it is not enough. Sampling replaces amortised sieving by per-candidate smoothness testing; but by the relation-count condition $\theta b - \eta = b$ one has $b + \eta = \theta b$, so the number of candidates inspected is $L_n(1/3,\, 2b)$ either way — exactly the sieve range — and an elliptic-curve smoothness test at $L_B(1/2, \sqrt{2}) = L_n(1/6, \cdot)$ per candidate is subexponentially small against $L_n(1/3, \cdot)$ and does not move the exponent. On cost alone a hashed oracle would still sit at $\sqrt[3]{6}$, below GNFS.

What is decisive is the *shape* of the reachable set. The one-sided norm bound of Section 3.2 holds because x runs over an *interval* of length $2S = L_n(1/3,\, 2b)$ — some 2^{69} values at $\log_2 n = 1024$ — and the skew of Section 3.1 is tuned to exactly that interval. The attainable values $H(m)$ form no interval: they are spread pseudorandomly across the full hash range, 2^{160} or 2^{256} wide. For x of that size the algebraic norm $|f(-x)| \simeq |x|^d$ is larger than the sieve ever contemplates and the smoothness probability collapses; and confining the search to messages with $H(m) < S$ costs 2^{256-69} hash evaluations, which is preimage work in all but name. Proposition 9 asks for a message slot of a given length, but it asks for an interval: a hash supplies a set of ample cardinality and of the wrong shape, and it is the shape the sieve consumes. Thus affine *structure* is necessary but not sufficient: the attacker also needs *steering* — an interval of attainable values — which a raw or fixed-pattern oracle grants and a hashed one denies. Rabin-PSS [28], being randomised, is not even affine.

7 Conclusion

At $e = 2$ the delayed-target attack of Joux–Naccache–Thomé specialises to a factoring algorithm: against a redundancy-constrained Rabin oracle it splits n in special-number-field- sieve time, below the general number field sieve, with the two-torsion sign that the odd-e algorithm suppresses now serving as the factoring signal. A short, exact

implementation witnesses every step. The moral is the one that recurs throughout this line of work: the security of a number-theoretic primitive under oracle access can be markedly lower than a factoring-based estimate suggests, and the gap follows the factor base — present wherever there is smoothness and a steerable oracle, absent the moment the pad becomes a hash.

8 Acknowledgements

This work was supported by RExQTCS: Romanian Excellence Quantum Technologies enhancing Cybersecurity, a grant of the Ministry of Education and Research from Romania - National Authority for Research and UEFISCDI, project number PN-IV-P6-6.1-CoEx-2024-0214, within PNCDI IV.

References

1. Joux, A., Naccache, D., Thomé, E.: When e-th roots become easier than factoring. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 13–28. Springer (2007). https://doi.org/10.1007/978-3-540-76900-2_2
2. Shea, L., Haller, M., Suhl, A., Heninger, N., Thomé, E.: Forging 1024-bit RSA signatures in nearly SNFS time. Cryptology ePrint Archive, Report 2026/2131 (2026), https://eprint.iacr.org/2026/2131
3. Rabin, M.O.: Digitalized signatures and public-key functions as intractable as factorization. Tech. Rep. MIT/LCS/TR-212, MIT Laboratory for Computer Science (1979)
4. Koblitz, N., Menezes, A.: Another look at non-standard discrete log and Diffie–Hellman problems. J. Math. Cryptol. **2**(4), 311–326 (2008). https://doi.org/10.1515/JMC.2008.014
5. Montgomery, P.L.: Square roots of products of algebraic numbers. In: Mathematics of Computation 1943–1993, Proc. Sympos. Appl. Math., vol. 48, pp. 567–571. AMS (1994)
6. Couveignes, J.-M.: Computing a square root for the number field sieve. In: Lenstra, A.K., Lenstra, H.W. (eds.) The Development of the Number Field Sieve. LNM, vol. 1554, pp. 95–102. Springer (1993). https://doi.org/10.1007/BFb0091541
7. Thomé, E.: Square root algorithms for the number field sieve. In: Arithmetic of Finite Fields (WAIFI 2012). LNCS, vol. 7369, pp. 208–224. Springer (2012). https://doi.org/10.1007/978-3-642-31662-3_15
8. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT '99. LNCS, vol. 1592, pp. 223–238. Springer (1999). https://doi.org/10.1007/3-540-48910-X_16
9. Damgård, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer (2001). https://doi.org/10.1007/3-540-44586-2_9
10. Okamoto, T., Uchiyama, S.: A New Public-Key Cryptosystem as Secure as Factoring. In: Nyberg, K. (ed.) EUROCRYPT '98. LNCS, vol. 1403, pp. 308–318. Springer (1998). https://doi.org/10.1007/BFb0054135
11. Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Comput. Syst. Sci. **28**(2), 270–299 (1984). https://doi.org/10.1016/0022-0000(84)90070-9
12. Benaloh, J.D.C.: Verifiable Secret-Ballot Elections. Ph.D. thesis, Yale University, Department of Computer Science, Technical Report YALEU/DCS/TR-561 (1987)

13. Naccache, D., Stern, J.: A New Public Key Cryptosystem Based on Higher Residues. In: CCS '98: Proceedings of the 5th ACM Conference on Computer and Communications Security, pp. 59–66. ACM (1998). https://doi.org/10.1145/288090.288106
14. Fiat, A., Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO '86. LNCS, vol. 263, pp. 186–194. Springer (1987). https://doi.org/10.1007/3-540-47721-7_12
15. Feige, U., Fiat, A., Shamir, A.: Zero-Knowledge Proofs of Identity. J. Cryptology **1**(2), 77–94 (1988). https://doi.org/10.1007/BF02351717
16. Guillou, L.C., Quisquater, J.-J.: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. In: Günther, C.G. (ed.) EUROCRYPT '88. LNCS, vol. 330, pp. 123–128. Springer (1988). https://doi.org/10.1007/3-540-45961-8_11
17. Koyama, K., Maurer, U.M., Okamoto, T., Vanstone, S.A.: New Public-Key Schemes Based on Elliptic Curves over the Ring $\mathbb{Z}_n$. In: Feigenbaum, J. (ed.) CRYPTO '91. LNCS, vol. 576, pp. 252–266. Springer (1992). https://doi.org/10.1007/3-540-46766-1_20
18. Demytko, N.: A New Elliptic Curve Based Analogue of RSA. In: Helleseth, T. (ed.) EUROCRYPT '93. LNCS, vol. 765, pp. 40–49. Springer (1994). https://doi.org/10.1007/3-540-48285-7_4
19. Okamoto, T.: A Fast Signature Scheme Based on Congruential Polynomial Operations. IEEE Trans. Inf. Theory **36**(1), 47–53 (1990). https://doi.org/10.1109/18.50373
20. Blum, M., Goldwasser, S.: An Efficient Probabilistic Public-Key Encryption Scheme Which Hides All Partial Information. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO '84. LNCS, vol. 196, pp. 289–299. Springer (1985). https://doi.org/10.1007/3-540-39568-7_23
21. Williams, H.C.: A Modification of the RSA Public-Key Encryption Procedure. IEEE Trans. Inf. Theory **26**(6), 726–729 (1980). https://doi.org/10.1109/TIT.1980.1056264
22. Coron, J.-S., Naccache, D., Stern, J.P.: On the Security of RSA Padding. In: Wiener, M. (ed.) CRYPTO '99. LNCS, vol. 1666, pp. 1–18. Springer (1999). https://doi.org/10.1007/3-540-48405-1_1
23. Coron, J.-S., Naccache, D., Tibouchi, M., Weinmann, R.-P.: Practical Cryptanalysis of ISO/IEC 9796-2 and EMV Signatures. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 428–444. Springer (2009). https://doi.org/10.1007/978-3-642-03356-8_25
24. Grieu, F.: A Chosen Messages Attack on the ISO/IEC 9796-1 Signature Scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 70–80. Springer (2000). https://doi.org/10.1007/3-540-45539-6_5
25. Desmedt, Y., Odlyzko, A.M.: A Chosen Text Attack on the RSA Cryptosystem and Some Discrete Logarithm Schemes. In: Williams, H.C. (ed.) CRYPTO '85. LNCS, vol. 218, pp. 516–522. Springer (1986). https://doi.org/10.1007/3-540-39799-X_40
26. Buhler, J.P., Lenstra, H.W., Pomerance, C.: Factoring integers with the number field sieve. In: Lenstra, A.K., Lenstra, H.W. (eds.) The Development of the Number Field Sieve. LNM, vol. 1554, pp. 50–94. Springer (1993). https://doi.org/10.1007/BFb0091539
27. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring Polynomials with Rational Coefficients. Math. Ann. **261**(4), 515–534 (1982). https://doi.org/10.1007/BF01457454
28. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures — How to Sign with RSA and Rabin. In: Maurer, U. (ed.) EUROCRYPT '96. LNCS, vol. 1070, pp. 399–416. Springer (1996). https://doi.org/10.1007/3-540-68339-9_34
29. Bellare, M., Rogaway, P.: Optimal Asymmetric Encryption. In: De Santis, A. (ed.) EUROCRYPT '94. LNCS, vol. 950, pp. 92–111. Springer (1995). https://doi.org/10.1007/BFb0053428
30. Kaliski, B.: PKCS #1: RSA Encryption Version 1.5. RFC 2313, RFC Editor (March 1998). https://doi.org/10.17487/RFC2313
31. ISO/IEC 9796-1:1999: Information technology — Security techniques — Digital signature schemes giving message recovery — Part 1: Mechanisms using redundancy. International Organization for Standardization, Geneva (1999), withdrawn

32. ISO/IEC 9796-2:2002: Information technology — Security techniques — Digital signature schemes giving message recovery — Part 2: Integer factorization based mechanisms. International Organization for Standardization, Geneva (2002)

A A Worked Toy Example

The following is a genuine run of the implementation at 24-bit primes, under a fixed seed, with the affine pad taken from the digits of π. Every quantity is printed in hexadecimal; note that the pad c reads off the decimal digits of π directly.

24-bit Primes $\rightarrow$ 48-bit Rabin Modulus ($e = 2$)	
secret p	`0xf36e53`
secret q	`0xba9ddb`
modulus n	`0xb17443364801`
pad c	`0x31415926535` (digits of π)
field $f(z)$	$z^2 + \texttt{0x111f3}\, z - \texttt{0xd22cf81}$ (skew 3737, real quadratic)
smoothness bound B	4096
factor base $\lvert F\rvert$	542 prime ideals
sieve interval	2^{19} values of x
relations	4614 (one answered oracle query each)
after filtering	703 rows, 535 columns
dependencies	64, of which 34 split (rate 0.53)
recovered p	`0xf36e53`
recovered q	`0xba9ddb`
check	$p \cdot q = n$, matches the secret primes [factored]

The oracle performed only black-box square-root (Rabin signing) operations; the factorization was recovered from the two-torsion sign of the congruence $\beta(c)^2 \equiv \big(\prod_{x \in S^*} s_x\big)^2 \pmod{n}$, never from the private key. The observed split rate $34/64 = 0.53$ is within one standard deviation of the $\frac{1}{2}$ of Proposition 3. Relations outnumber the factor base by a wide margin because the filtering step must also discard singletons: 4614 relations reduce to 703 rows over 535 surviving columns, and it is that excess, not the raw count, that determines the dependency count.

Two honest caveats. First, at 48 bits the pad leaves almost the whole residue range free — $n - c$ is itself 48 bits — so this example exercises the *machinery* end to end but not the redundancy *constraint*; the constraint is what the padding campaign of the accompanying notebook measures. Second, the degree is fixed at 2 here, which is optimal at this size but not at larger sizes.

B Applicability to Other Factoring-Based Schemes

The attack of this paper is one instance of the delayed-target, oracle-assisted number field sieve. It is worth asking which *other* factoring-based primitives expose an oracle

that the same one-sided sieve can consume. Stripped to its load-bearing parts, the technique needs three things at once.

(C1) A public fixed-exponent root on $\mathbb{Z}_n^*$. The oracle must invert the *publicly known* power map $w \mapsto w^e$ on its image: on input w it returns some s with $s^e \equiv w$. For e coprime to $\varphi(n)$ that inverse is the single value w^d, $ed \equiv 1 \pmod{\varphi}$; for $e = 2$ it is one of four roots, and the resulting ambiguity is not a defect but the signal this paper exploits. What (C1) buys is that oracle answers on factor-base primes *multiply* into an answer on any product of them, which is what makes the linear algebra a system modulo e. In the forgery setting the attacker must also know e to form the target relation $t \cdot m^e = \prod_i p_i^{a_i}$; in the factoring setting of this paper there is no external target and that requirement lapses (Remark 6).

(C2) A factor base. The group on which the oracle acts must admit subexponential index calculus, i.e. a notion of smoothness: $\mathbb{Z}_n^*$, or a number field mapping into it.

(C3) Steering over an interval. The attacker must choose *which* w is rooted, and must be able to reach a whole *interval* of values: raw, or affine with the message contiguous, so that a sieve range falls inside the queryable set. Cardinality alone does not suffice — a set of the right size but the wrong shape fails, whether it is scattered by a digit interleaving (§C.1) or by a hash (Section 6). A per-query randomiser removes steering outright.

Two structural consequences organise the whole landscape. First, steering power is monotone — raw ⊇ affine ⊇ hash — so a scheme can only *lose* applicability as the padding hardens; the per-row pattern is always one of ✓ ✓ ✗, ✓ ✗ ✗, or ✗ ✗ ✗. Second, a full hash defeats (C3) universally, so *the hash column is crossed for every scheme*: any vulnerability lives only in the raw or affine variant. Table 1 classifies the genuinely distinct families across the three padding variants; the subsections that follow explain each case.

RSA, Odd e

The original construction of [1]. It satisfies (C1) with the public verification exponent and (C2) on $\mathbb{Z}_n^*$. Raw (a signing/decryption oracle in `CKM_RSA_X_509` mode, or blind RSA) and affine (ISO 9796 message recovery) give full steering, so both are attackable; PKCS#1 v1.5 and PSS/FDH/OAEP [29] hash the message and remove steering (Appendix C).

Rabin, Rabin–Williams, Blum–Goldwasser ($e = 2$) [3,21,20]

All three expose a square-root oracle: Rabin signing, the Williams principal-root tweak, and Blum–Goldwasser decryption (which is Rabin decryption of the Blum–Blum–Shub seed). They satisfy (C1) with $e = 2$ and (C2). The raw variant is trivially fatal — a random square is rooted to an independent root and $\gcd$ splits n in one query — so the number field sieve is superfluous there; the *affine* variant is the subject of this paper, where the one-sided sieve factors n at SNFS speed. Hashed Rabin (FDH) is safe.

ESIGN (p^2q, Public e) [19]

The closest structural miss. ESIGN has a *public* exponent, satisfying (C1)'s exponent requirement, and a factor base exists ((C2) holds; the number field sieve works over p^2q). It is nonetheless immune, and for a reason orthogonal to padding: the trapdoor produces only an *approximate* e-th root (only the high-order bits of $s^e \bmod n$ are pinned to the message), so no exact multiplicative identity holds, and the sieve has no exact relations to collect. Its crosses in Table 1 are therefore padding-independent. An exact-root variant "RSA modulo p^2q" would instead read ✓ ✓ ✗.

Paillier (Encryption and Signature) [8]

Immune, and instructively so: the obstruction survives even when every condition is granted.

The immediate reason is (C1). The trapdoor is a *secret* exponent $\lambda = \mathrm{lcm}(p-1, q-1)$ acting on $\mathbb{Z}_{n^2}^*$, post-composed with the easy base-$(1+n)$ logarithm. A decryption oracle inverts a fixed-base, variable-exponent map — it returns the class m of $c = g^m r^n$ and discards r altogether — whereas the technique needs a fixed-exponent, variable-base root $w \mapsto w^{1/e}$, so that answers on factor-base primes *multiply* into the answer on the target. The target relation of (C1) cannot even be written. Note that steering, (C3), was never the difficulty: a decryption oracle accepts every element of $\mathbb{Z}_{n^2}^*$, so Paillier enjoys unrestricted steering whether or not it is randomised, and there is no padding column to lose.

The deterministic variant. Fixing r therefore relaxes the one condition Paillier already satisfied and leaves untouched the one it fails. It does break the scheme, but in the wrong direction and without any cryptanalysis: with the standard generator $g = 1+n$ one has $c = (1+n)^m = 1+mn \bmod n^2$, so $m = (c-1)/n$ is read off with no key at all. The ciphertext is then an affine function of the message — exactly the shape this paper exploits — but the affine map is public and invertible, so no trapdoor remains to attack. Confidentiality collapses; n is untouched. For a general g of order αn the deterministic scheme is a plain encoding, vulnerable to the usual dictionary and meet-in-the-middle recovery on a small message space, and again with no consequence for n.

Granting (C1) anyway. Paillier does contain one genuine public fixed-exponent power map, namely $r \mapsto r^n \bmod n^2$ inside the encryption. An oracle returning the randomness r from a ciphertext — randomness recovery, which occurs in proof-of-correct-decryption and some CCA settings — is an AMR$_2$-type oracle with public exponent $e = n$ over $\mathbb{Z}_{n^2}^*$, satisfying (C1) and (C2), and with (C3) unrestricted. Even so the technique yields nothing, for an arithmetical reason with no slack in it.

Proposition 10 (Squaring the modulus annihilates the gain). *Let the delayed-target one-sided sieve of Theorem 5 be run against a randomness-recovery oracle for Paillier, i.e. with working modulus n^2. Its cost, expressed in the size of n, is*

$$L_{n^2}\left(\frac{1}{3}, \sqrt[3]{\frac{32}{9}}\right) = L_n\left(\frac{1}{3}, \sqrt[3]{2}\cdot\sqrt[3]{\frac{32}{9}}\right) = L_n\left(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right),$$

which is exactly the cost of running GNFS *on* n *itself.*

Proof. $\ln n^2 = 2\ln n$, so to leading order $\sqrt[3]{(\ln n^2)(\ln\ln n^2)^2} = \sqrt[3]{2}\,\sqrt[3]{(\ln n)(\ln\ln n)^2}\,(1+o(1))$, and $\sqrt[3]{2}\cdot\sqrt[3]{32/9} = \sqrt[3]{64/9}$.

The attack is in fact strictly dominated. At $e = n$ it returns a root, that is a single delayed decryption, whereas GNFS at the same cost returns the factorisation of n and with it the private key. An adversary who can afford one can afford the other and should prefer the latter. This is a stronger justification for the crossed row than the failure of (C1), because it does not depend on (C1) at all.

Corollary 11 (Damgård–Jurik [9]). *With working modulus* n^{s+1} *the effective constant against* n *is* $\sqrt[3]{32(s+1)/9}$*, which is below the* GNFS *constant* $\sqrt[3]{64/9}$ *only for* $s+1<2$ *— a value no member of the family attains, since Damgård–Jurik is defined for* $s\geq 1$*. Paillier,* $s=1$*, sits exactly at break-even, and every higher level is strictly worse. The degenerate* $s=0$*, which the formula would reward, is not a member of the family: it is the modulus* n *itself, i.e. the* RSA *setting of [1].*

The whole Damgård–Jurik family therefore lies at or beyond the break-even point by construction, and not by accident: the advantage $\sqrt[3]{64/9}\to\sqrt[3]{32/9}$ that the one-sided sieve extracts is worth precisely a factor $\sqrt[3]{2}$, which is what one level of modulus inflation costs.

Okamoto–Uchiyama (p^2q) [10]

The trapdoor is a p-adic logarithm modulo p^2, a secret-exponent map, so (C1) fails. The p^2q modulus additionally changes the polynomial-selection regime of the underlying sieve, so even the factoring baseline differs; but the decisive obstruction is again the absence of a public root.

Goldwasser–Micali [11]

Decryption is a quadratic-residuosity *decision*: the oracle returns a single bit, not a root. Nothing multiplies, and a residuosity-deciding oracle is not known to factor. (C1) fails at the most basic level.

Benaloh and Naccache–Stern (Higher Residues) [12,13]

Decryption recovers a message by a discrete logarithm among small prime-residue classes using the secret factorization — a secret-exponent/log structure, not a public root. (C1) fails, independently of padding.

Fiat–Shamir, Feige–Fiat–Shamir, Guillou–Quisquater [14,15,16]

These *do* reveal roots — square roots for (F)FS, RSA roots for GQ — so (C1) appears met.

They fail on (C3) for an intrinsic reason: the honest prover always reveals $\sqrt{r^2 \prod_i v_i^{c_i}}$ (respectively $r \cdot s^c$) with a *fresh* per-proof randomiser r, which masks the rooted value. Steering is gone even in the raw and affine columns; this randomiser is the identification-protocol analogue of the PSS salt.

KMOV and Demytko (Elliptic Curves Modulo n) [17,18]

Factoring-based, yet immune for the cleanest reason of all: the oracle acts on $E(\mathbb{Z}_n)$, and elliptic-curve groups have no subexponential index calculus. (C2) fails outright — there is no factor base to sieve, so the one-sided relation collection has nothing to collect. The trick follows the factor base, not the modulus.

Table 1. Applicability of the delayed-target one-sided NFS across factoring-based schemes and padding variants. A checkmark (✓) marks that the technique yields a sub-GNFS forgery, or a factorization at $e = 2$; a cross (✗) marks that it does not. The hash column is crossed for all schemes (a hash removes steering), and each row is monotone left-to-right.

Scheme (Trapdoor)	Raw	Affine	Hash	Obstruction when ✗
RSA, odd e: JNT (coded by Shea et al.)	✓	✓	✗	— (hash removes steering, C3)
Rabin / Rabin–Williams / Blum–Goldwasser (e=2)	✓†	✓	✗	— (raw is trivial; hash safe)
ESIGN (p^2q, public e)	✗‡	✗‡	✗	approximate root: no exact relations
Paillier (encryption & signature)	✗	✗	✗	secret exponent λ (C1)‡‡
Okamoto–Uchiyama (p^2q)	✗	✗	✗	secret p-adic log (C1)
Goldwasser–Micali	✗	✗	✗	oracle returns a bit, not a root (C1)
Benaloh / Naccache–Stern	✗	✗	✗	secret residue-class log (C1)
Fiat–Shamir / Feige–FS / GQ	✗§	✗§	✗	prover randomiser r^2 (C3)
KMOV / Demytko (EC modulo n)	✗	✗	✗	no index calculus / factor base (C2)

† Raw Rabin splits n with probability $\frac{1}{2}$ per query; the checkmark holds but the sieve is unnecessary, and the *affine* cell is where the technique does real work. The row covers signers whose canonical root is *not* multiplicative on the quadratic residues; a Rabin–Williams signer returning principal roots falls outside Proposition 3 and needs the separate argument of Remark 4.
‡ ESIGN meets (C1)'s exponent and (C2), and is defeated *only* by the approximateness of its root; its crosses are independent of the padding column.
§ (F)FS and GQ reveal roots, but always under a fresh randomiser r, so steering is absent even without a hash.
‡‡ Paillier's row is padding-independent for a second, stronger reason: even granting a public-exponent oracle over $\mathbb{Z}^*_{n^2}$, the working modulus n^2 costs a factor $\sqrt[3]{2}$, which is exactly the advantage the one-sided sieve extracts (Proposition 10).

The reading of Table 1 is stark: only two trapdoors — RSA (odd e) and the Rabin family ($e = 2$) — carry any checkmark, and only in the raw or affine variants. Every other scheme is crossed for one of five orthogonal reasons: a *secret exponent* (Paillier, Okamoto–Uchiyama, Benaloh, Naccache–Stern), an oracle that is *not a root* (Goldwasser–Micali), a root that is *not exact* (ESIGN), the *absence of a factor base* (KMOV, Demytko), or an *inflated working modulus* that costs exactly what the one-sided sieve

saves (Paillier again, and the whole Damgård–Jurik family, by Proposition 10 and Corollary 11), with the Fiat–Shamir family removed by intrinsic randomization. The wholly crossed hash column and the two-scheme monopoly on the checkmark are the compact statement of the entire applicability question.

C ISO 9796 [31,32] and PKCS#1 v1.5 [30] Signature Padding When Applied to $e = 2$

Since the attack lives or dies by whether the message-to-be-rooted is *steerable*, the concrete signature-padding standards deserve individual treatment. Table 2 summarises the RSA padding options; the subsections that follow explain each, in order of decreasing exposure.

Note that all the following assumes a setting where the concerned padding schemes are applied to Rabin. We are not aware of such deployments but mention these for their theoretical interest.

Remark 12 (The padded value must not be reduced). Everything below, and the whole analysis of Section 4, presupposes that the padded value satisfies $c + \mu < n$, so that the signer roots it as an integer rather than as a residue. An implementation that instead lets $c + \mu$ exceed n and reduces is broken by elementary means, and the one-sided sieve is beside the point for it: for μ just past $n - c$ the value $(c + \mu) \bmod n = c + \mu - n$ is *small*, so the oracle is returning square roots of small integers. Trial-dividing those over a modest factor base, taking an $\mathbb{F}_2$ dependency among their exponent vectors and multiplying the corresponding oracle answers gives $X^2 \equiv Y^2 \pmod{n}$ directly — no polynomial selection, no number field, no square root beyond an integer one. In our implementation this factors a 1024-bit modulus from 581 oracle queries in a few seconds, and the query count is governed by the width of the wrap rather than by $\log_2 n$: it is unchanged at 512, 1024 and 2048 bits. The affine analysis therefore assumes a signer that *rejects* an out-of-range encoding rather than reducing it, and this is a property of the implementation, not of the padding format.

Table 2. RSA signature/encryption padding options against the delayed-target one-sided NFS. A checkmark (✓) in "Attack Applies" means the message-to-be-rooted is steerable and the sieve can be run; a cross (✗) means it cannot. Only a raw or blind signer is exposed: no standardised padding here leaves the message-to-be-rooted steerable over an interval.

Padding	Encoded Value	Steerable?	Attack Applies?	Note
Raw / `X_509` / blind	μ chosen directly	✓	✓	full steering; constant $\sqrt[3]{32/9}$
ISO 9796-1 (recovery)	Π-interleaved nibbles	✗	✗	not affine over $\mathbb{Z}$; slot scattered (§C.1)
ISO 9796-2 (recovery)	hdr $\|\|$ μ $\|\|$ $H(\mu)$ $\|\|$ BC	✗	✗	full hash; short-hash attack separate
PKCS#1 v1.5 (signature)	$c + H(\mu)$ (affine skeleton)	✗	✗	preimage barrier; would-be $\sqrt[3]{6}$
Rabin-FDH	$H(\mu)$ full-domain	✗	✗	no steering
Rabin-PSS	randomised (salt)	✗	✗	salt removes steering *and* structure
RSA-OAEP (encryption)	randomised	✗	✗	randomised; not a fixed root

C.1 ISO/IEC 9796-1 (Message Recovery)

The withdrawn ISO 9796-1 scheme is the obvious candidate for an affine target, and it fails on both halves of (C3). Its encoding maps each message byte $x \,\|\, y$ through a local injection [24]

$$F_i(x \,\|\, y) = \Pi(x) \,\|\, \Pi(y) \,\|\, x \,\|\, y,$$

where Π is a fixed permutation of the nibbles. First, the encoded value is therefore *not* an affine image of μ over $\mathbb{Z}$: Π is affine over $\mathbb{F}_2$ but not over the integers — $\Pi(0), \Pi(1), \Pi(2), \Pi(3) = \texttt{E}, \texttt{3}, \texttt{5}, \texttt{8}$ have successive differences $-11, +2, +3$ — so no constant c makes the encoding $c + \mu$.

Second, and independently of affinity, the attacker-controlled digits are *scattered*. Each message byte governs four nibbles of the encoding, so a message space of cardinality 2^{8z} is spread over a span of 2^{16z}: the reachable set has cardinality S but magnitude S^2. Since the skew of Section 3.1 is tuned to the magnitude and not to the cardinality, this is fatal to the margin of Theorem 5. Writing w_{eff} for the exponent measured from $|f(-x)| \simeq n^{1/d} S^{w_{\text{eff}} d}$ with S the cardinality, our implementation gives, at $\log_2 n = 384$ and $d = 4$:

$\log_2 \lvert\mathcal{M}\rvert$	24	32	40	48
contiguous slot	0.36	0.37	0.40	0.50
interleaved slot	0.83	1.12	1.30	1.42

At the operating point the contiguous slot reproduces the predicted $w = \frac{1}{2}$, which calibrates the measurement; the interleaved slot gives $w_{\text{eff}} \simeq 1.42$, and by Remark 8 that is a constant $\sqrt[3]{64(1.42)/9} \simeq 2.16$ — *above* $\sqrt[3]{64/9} \simeq 1.923$. Against ISO 9796-1 the one-sided sieve would therefore be slower than simply factoring n with the general number field sieve, and the technique of this paper does not apply. What defeats it is the same thing that a hash does, and for the same reason (Section 6): the reachable set has ample cardinality and the wrong shape. Cardinality is not the binding resource; contiguity is.

ISO 9796-1 was of course broken, by the Coron–Naccache–Stern [22] and Grieu [24] attacks, which exploit multiplicative relations among specially structured encodings rather than smoothness of an algebraic norm; it was withdrawn in 1999. That those attacks succeed where this one does not is a useful reminder that the two lines of cryptanalysis ask for different structure.

C.2 ISO/IEC 9796-2 (Hash-Based Message Recovery)

The successor standard replaced the affine redundancy with a *hash*: the encoded value carries a header, a portion of the message, the hash $H(\mu)$, and the trailer `0xBC`. Because the encoded integer now depends on μ through $H(\mu)$, the attacker can no longer steer it to a smooth value without inverting the hash. Two regimes must be distinguished. In *partial message recovery* with a full-width hash the scheme behaves, for our purposes, like a hashed padding and is not steerable. The historically relevant weakness of 9796-2 was different in nature — the Coron–Naccache–Tibouchi–Weinmann forgeries [23] exploited *short* hash lengths and Desmedt–Odlyzko-style [25] multiplicative relations

among hashed values — and while that is also a smoothness attack, it is not the one-sided *oracle* sieve of this paper: it needs many honestly generated signatures and small hashes, and it does not place the padded value in the affine, attacker-steered position that (C3) requires. With a full-length hash, 9796-2 falls in the crossed hash column of Table 1.

C.3 PKCS#1 v1.5 Signatures When $e = 2$

The v1.5 encoding is `0x00 0x01`, a run of `0xFF` octets, `0x00`, then the ASN.1 DigestInfo wrapping $H(\mu)$. Structurally this is affine — $\mathrm{EM} = c + H(\mu)$ with c the fixed high-order frame and $H(\mu)$ the low-order variable — so in *form* it is an AMR_2 oracle, and the affine-oracle variant would apply at the (already marginal) constant $\sqrt[3]{6} \simeq 1.817\ldots$. The reason it is safe in practice is precisely the hash: the low part is $H(\mu)$, not an attacker-chosen μ, so hitting a chosen smooth low part x^* would require a preimage $H(\mu) = x^*$. The attacker is reduced to *sampling* messages and keeping those whose encoded norm happens to be smooth. As Section 6 shows, the extra per-candidate cost of that is not by itself enough to move the exponent; what kills the attack is that $H(\mu)$ ranges over the whole hash output space rather than over a short interval, so the norm bound the one-sided sieve relies on simply does not hold. Thus PKCS#1 v1.5 exhibits the affine *skeleton* without the affine *exposure*: the structure is present but the steering is not, and it sits in the hash column.

C.4 Rabin-PSS

For completeness, PSS is not merely hashed but *randomised*: a fresh salt enters the encoding, so even two signatures of the same message root different values. This removes both steering (C3) and any static multiplicative structure, and is the strongest of the standard choices against the present technique. It is the padding we recommend where a raw or affine signer would otherwise be exposed.