A Vacuous Binding Theorem, and Its Repair

the generic-group security of KZG evaluation binding, mechanized in Lean

ArkLib `d72f8392` · Lean v4.31.0
all Lean claims `sorry`-free against upstream

ArkLib — the Ethereum Foundation's Lean 4 library of succinct-argument building blocks — proves evaluation binding for the KZG polynomial commitment conditionally on a `t`-SDH hardness assumption. As stated, that assumption is unconditionally false: it quantifies over adversaries with **no resource bound**, and an adversary definable with `Classical.choice` reads the trapdoor out of the public reference string and wins with probability 1 — so the binding theorem holds vacuously and carries no information at any parameter. Lean's axiom check reports it clean regardless: axiom-cleanliness never inspects whether a theorem's hypotheses are satisfiable. This spread presents the mechanized refutation, an unconditional restatement of the reduction that no longer rests on the false assumption, and a generic-group-model security bound for KZG evaluation binding, proved in both standard formulations of the model at the identical bound.

Research note — a **formalization-soundness** finding in a public, in-development library, not a vulnerability in any deployed system. KZG, `t`-SDH, and the reduction between them are sound as normally stated; the issue is a Lean quantifier.

I · The finding: a theorem that proves nothing

KZG and its binding theorem

Fix prime-order groups G_1, G_2, G_T (order p) with a bilinear pairing $e : G_1 \times G_2 \to G_T$ and generators g_1, g_2. The KZG polynomial commitment publishes a **structured reference string** (SRS) generated from a secret trapdoor τ:

$$\mathrm{srs} = \left(\left(g_1, g_1^{\tau}, \ldots, g_1^{\tau^D}\right),\ (g_2, g_2^{\tau})\right).$$

A polynomial f of degree $\le D$ is committed as $C = g_1^{f(\tau)}$, computable from the SRS without knowing τ; an opening of C at a point z to a value v is checked by one pairing equation. **Evaluation binding** — no adversary can open one commitment at one point to two different values — is what a verifier relies on, and it is classically proved by reduction to the `t`-Strong Diffie–Hellman (`t`-SDH) assumption: given $\left(g, g^{\tau}, \ldots, g^{\tau^D}\right)$, it is hard to output a pair $\left(c, g^{1/(\tau+c)}\right)$. ArkLib mechanizes this reduction in `KZG.CommitmentScheme.binding`, and the reduction itself is constructive and algebraically sound. The problem is the assumption it consumes.

An assumption with no resource bound

```
abbrev tSdhAdversary (D : ℕ) :=
  Vector G₁ (D+1) × Vector G₂ 2 → StateT unifSpec.QueryCache ProbComp (Option (ZMod p × G₁))

def tSdhAssumption (D : ℕ) (error : ℝ≥0) : Prop :=
  ∀ (adversary : tSdhAdversary D), tSdhExperiment D adversary ≤ error
```

The adversary is a **plain function type**. `ProbComp` is a free monad over oracle queries: only `query` nodes cost anything, and pure computation is unmetered. Nothing bounds running time, and the function body may be any term of the right type — including a noncomputable one built with `Classical.choice`.

The adversary that empties it

The SRS hands the adversary the verifier leg g_2^{τ}, which determines τ whenever $g_2 \neq 1$; and the discrete logarithm in a prime-order group is `Classical.choice`-definable, via ArkLib's own lemma `exists_zmod_power_of_generator`. So an adversary recovers τ and returns the `t`-SDH solution at offset zero:

```
noncomputable def tauExtractingAdversary (hg₂ : g₂ ≠ 1) (D : ℕ) : tSdhAdversary D :=
  fun srs => pure (some (0, g₁ ^ (1 / dlogOf hg₂ srs.2[1]).val))

theorem tSdhExperiment_tauExtractingAdversary : tSdhExperiment D (…) = 1
```

It wins with probability **exactly** 1 — the trapdoor sampler avoids 0, so $\tau + 0 \neq 0$ on the whole support — and it makes **zero** oracle queries: all of its work happens under `pure`, so any query-counting resource bound constrains something it never does. A canary confirms the experiment discriminates: the adversary that returns `none` scores exactly 0.

No content at any parameter

```
theorem not_tSdhAssumption               (herr : error < 1) : ¬ tSdhAssumption D error
theorem tSdhAssumption_trivial_of_one_le (1 ≤ error)        : tSdhAssumption D error
```

Below 1 the assumption is refuted; at ≥ 1 its conclusion is the triviality "a probability is ≤ 1." And `binding` cannot dodge the hypothesis the extractor needs: its own premise `hpair : pairing g₁ g₂ ≠ 0` **forces** $g_2 \neq 1$, because a bilinear map kills the identity. So `binding`'s hypotheses are jointly unsatisfiable for any error below 1 (`binding_hypotheses_unsatisfiable`), and its conclusion is free at ≥ 1: the theorem says nothing at any parameter. The sibling ARSDH assumption behind `function_binding` has the identical unrestricted quantifier and falls identically. Nor is the shape specific to `t`-SDH: a `q`-strong-DLOG assumption stated in the same idiom — recover τ from the power SRS — is voided by the same extraction (`not_qDlogAssumption`).

Axiom checks cannot see this

The vacuous theorem, the winning adversary, and the refutation all print the **same** clean axiom closure:

```
#print axioms not_tSdhAssumption   -- [propext, Classical.choice, Quot.sound]
```

No `sorryAx`, no custom axiom — nothing a `#print axioms` gate would flag. **Axiom-clean and vacuous coexist.** The blindness is structural: an axiom check reports the axioms in a proof term's **closure**; it never asks whether the theorem's **hypotheses** are jointly satisfiable, and a theorem with an unsatisfiable hypothesis is axiom-clean and content-free at once. The point generalizes: any named hard-problem `def Foo : Prop` used as a **hypothesis** is an assumption no axiom check ever inspects. The reliable test is adversarial — try to inhabit the assumption's **negation**, which is precisely what `tauExtractingAdversary` does.

II · The repair, and the bound that grounds it

The unconditional restatement

ArkLib's binding proof is a five-step `calc`: four unconditional steps rewrite the binding-game success probability into the `t`-SDH success probability of an **explicitly constructed** reduction adversary, and only the fifth applies the assumption. Splitting at that one step gives the primary theorem:

```
theorem binding_reduces_to_tSdh … (adversary : KzgBindingAdversary …) :
    bindingExperiment … adversary ≤ tSdhExperiment g₁ g₂ n (bindingReduction … adversary)
```

This takes **no hardness assumption** — both sides are concrete probabilities and the inequality holds at every parameter, with nothing for `Classical.choice` to inhabit. The change is small (+41/−14 in one file), preserves the original reduction verbatim, and keeps `binding` as a one-line corollary. The restatement **provably coexists with the attack** (`repair_survives_attack`): in one `sorry`-free closure the trapdoor-extracting adversary still refutes `tSdhAssumption` below 1 while the restated bound holds regardless. What remains is one obligation — bound the constructed reduction adversary's success — which the generic-group model supplies.

The generic-group model, and why $1/(X+c)$ is out of reach

In the generic-group model the adversary never sees a group element: it holds opaque **handles** and combines them only through oracles. Behind each handle the model keeps an **ordinary polynomial** in $\mathbb{Z}_p[X]$, with X the formal trapdoor: the SRS seeds $1, X, \ldots, X^D$, and the group operation adds or subtracts exponent polynomials — inversion **negates** the exponent, it never introduces X^{-1}, so the exponent ring is $\mathbb{Z}_p[X]$, not Laurent. A `t`-SDH win needs a handle whose exponent equals $1/(X+c)$ — not a polynomial, hence unrepresentable. The best a generic adversary can do is output some polynomial f of degree $\le D$ that happens to satisfy $f(\tau)\cdot(\tau+c) = 1$ at the specific random τ: every winning τ is a root of the **nonzero**, degree-$\le D+1$ polynomial $f\cdot(X+c) - 1$, so Schwartz–Zippel caps the winning trapdoors at $D+1$ out of $p-1$. Adding the standard collision bad event — two formally distinct handle polynomials evaluating equal at τ, handled by an identical-until-bad simulation — yields the full bound.

One bound, both standard models — wired to ArkLib

The literature fixes two standard formulations of "the adversary cannot see group elements," differing only in **how it learns equalities among the handles it holds**. Both are mechanized, `sorry`-free, and both bound ArkLib's **own** `Groups.tSdhExperiment` — the very experiment of Panel I, restated nowhere — at the **identical** bound: for an adversary running at most q generic steps against a degree-D SRS,

$$\Pr[\mathrm{break}] \ \le\ \frac{\binom{q+D+4}{2}\cdot D + (D+1)}{p-1},$$

a genuine < 1 whenever the numerator is below $p-1$. Each is proved for the image of an embedding of the generic strategy into ArkLib's adversary type:

```
tSdh_ggm_sound       : tSdhExperiment D (embed strat)      ≤ (C(q+D+4,2)·D + (D+1))/(p−1)
shoup_tSdh_ggm_sound : tSdhExperiment D (embedShoup strat) ≤ (C(q+D+4,2)·D + (D+1))/(p−1)
```

Maurer (explicit equality). An equality is learned only by spending an explicit query (`Move.query`), so only queried pairs can collide and the all-pairs count is a sound **over-count**; `embed` answers each query by real group equality. **Shoup (random encodings).** Random **encodings** under an injection $\sigma : \mathbb{Z}_p \hookrightarrow E$ are compared **for free** — no query move; the adversary sees the full pairwise-equality matrix at each step, so every held pair is a live candidate and the count is **tight**. Its `embedShoup` realizes this in the concrete group — the group-equality matrix of the held handles equals the symbolic one off the collision event, by prime-order injectivity of $a \mapsto g_1^a$ — so free comparison is **discharged, not assumed**. Both bounds share the side-conditions $1 \le D$ (at $D = 0$ the statement is genuinely false — a pairing-free G_1 adversary cannot form g_1^{τ}), $2 \le p$, orderOf $g_1 = p$ ($g_1, g_2 \neq 1$), and ArkLib's own `SampleableType` instance.

Why the bound escapes the vacuity. Each class receives only equality information — an explicit boolean, or the free matrix — never a group element, so it realizes only $g_1^{f(\tau)}$ with $\deg f \le D$, and the trapdoor-extracting adversary of Panel I **cannot even be expressed** in it. Over the full unrestricted type the same statement is provably false; the restriction is where the number comes from.

What is mechanized (all `sorry`-free; axiom closure exactly `[propext, Classical.choice, Quot.sound]`)

- **The refutations** — `t`-SDH, ARSDH, and `q`-DLOG, each with a discriminating canary, against genuine upstream ArkLib (imported, nothing redefined; whole tree builds).
- **The restatement** — `binding_reduces_to_tSdh` and its coexistence with the attack, `repair_survives_attack`.
- **The two-model bound, full spine** — the Schwartz–Zippel root count; the identical-until-bad simulation, **proven by induction, not assumed**, in both the explicit-query and free-comparison forms; the all-pairs collision count with the handle-table size a theorem; degree bounds proved on the **actual** oracle (ArkLib's `t`-SDH adversary is granted no pairing map, so the oracle is linear and every exponent stays degree $\le D$); the field-to-group transport onto ArkLib's real win condition `tSdhCondition`, by prime-order injectivity; the probability-monad threading that collapses ArkLib's game to a `Finset` count; and the two embeddings of generic strategies into ArkLib's adversary type — explicit-query (`embed`) and free-comparison matrix (`embedShoup`) — each carrying its model's bound onto ArkLib's own experiment.

Artifacts (this directory; all `sorry`-free against ArkLib `d72f8392`): `KzgVacuity.lean` (the refutations) · `binding-repair.patch` (`binding_reduces_to_tSdh`) · `RepairSurvives.lean` · `candidates/GgmEndToEnd.lean` & `GgmShoupEmbed.lean` (the Maurer & Shoup bounds on ArkLib's experiment, `tSdh_ggm_sound` / `shoup_tSdh_ggm_sound`) · `candidates/GgmShoup.lean` (Shoup symbolic core) · supporting spine `candidates/Ggm{Candidate, Adaptive, RandomEncoding, DegreeDischarge, ArkLibTransport, ProbThreading, Embed}.lean` · `candidates/KzgQDlogVacuity.lean`. **Reproduce the vacuity:** drop `KzgVacuity.lean` into `ArkLib/Scratch/`, `lake build`, `#print axioms`. **References:** [KZG10] Kate–Zaverucha–Goldberg · [BB04/BB08] Boneh–Boyen · [Sho97] Shoup · [Mau05] Maurer · [FKL18] Fuchsbauer–Kiltz–Loss · [CGKY25] Chiesa–Guan–Knabenhans–Yu.