KeyFuse: Falsification-First Cache-Key Auditing with Minimal Collision Witnesses and Conservative Key Repair

DeepForge Research

September 18, 2026 · keyfuse

Technical report · Artifact · Not peer reviewed

Abstract

A build cache key describes the inputs a project declared, not the inputs the build actually read. When a task reads an undeclared slot — an environment variable, the working directory, the locale, the timezone, a clock, an RNG seed, a config file pulled in by a plugin — two builds can share a key and produce different outputs, and the cache serves the wrong artifact. Public incidents in Nx, Metro, Gradle, and Turborepo document the shape. KeyFuse is a falsification-first, read-only auditor over a finite typed slot universe. It probes a task with five strategies from a baseline, records the rows whose output moves, and reports (1) a minimal collision witness — a same-declared-key pair differing at exactly one slot whose removal restores the baseline output, verified per-slot at the recorded witness context — and (2) a conservative repaired key over declared union implicated. On a frozen 24-task corpus with brute-force ground truth (408 binary assignments, 651 full product assignments, 1,533 enumerated same-key collision pairs), the exact default arm matched ground truth in 69 of 72 task-by-strength cells with 3 explicit nondeterminism refusals; the conservative fallback's attributed detections were a subset of the exact detections with all 53 absences classified; 83 minimized witnesses were produced, 81 verified 1-minimal with 2 flagged and reported; and repair separated all 101 recorded collision witnesses in the gate's separating check with 0 unseparated, while the documented anchor failure is reported separately (1 documented minimal pair, 12 Hamming-1 pairs, 55 exhaustive residual pairs). Masking is demonstrated, not assumed: a pinned adversarial covering array misses the (a,b) pair of f = (a AND b) XOR c and the miss is recorded. The certificate is exactly "no detected <=t-support effect at covered tuples" — detection is not soundness, a repaired key is conservative rather than complete, and the anchor condition is an assumption. The shipped product is a browser-local, read-only lab with no storage, no grading, and no network.

Keywords: build cache, cache key, undeclared inputs, combinatorial interaction testing, delta debugging, minimal witness, key repair, reproducibility

---

1. Introduction & motivation

A build cache is a bet that the declared key names everything the build reads. The Nx team's "Can You Trust Your Build Cache?" describes remote cache poisoning and stale-hit classes that follow from keying on declared inputs; expo/expo issue #30930 and facebook/metro issue #918 document Metro cache-key collisions around custom transformers and external inputs; gradle/gradle issue #16144 documents build-cache key normalization problems; and the Gradle and Turborepo documentation devotes entire pages to environment variables that silently enter or fail to enter keys. In each case the failure shape is the same: two builds share a key, produce different outputs, and the cache is confident.

KeyFuse is a falsification-first auditor for that shape. It does not attempt to synthesize a correct key and it does not claim one. Given a deterministic oracle over a finite typed slot universe, a baseline assignment, and a declared input list, it runs one of five probe strategies, records every row whose output differs from the baseline, and reports what it detected: a minimal collision witness pair whose two assignments differ at exactly one slot and whose outputs differ, with per-slot necessity checked at the recorded witness context, and a repaired key over declared union implicated. The witness is the product; the repair is the intervention that follows from detected dependence when the inputs cannot be declared properly — a third-party plugin, a generated config, a vendored tool.

- Typed slot universe: file, env, cwd, locale, timezone, rng, clock, with frozen name forms and baseline/top/sentinel values; slots outside the universe are misses, not defaults.
- Five probe strategies: baseline-toggle, single-trace, ca, ca-ddmin (fallback), cover-with-defaults (default), all deterministic and budget-bounded at 4,096 oracle calls.
- Minimal collision witness: same declared key, different outputs, differing set of size 1, necessity verified at the witness context; 1-minimality of minimized supports verified by exhaustive single-coordinate removal.
- Conservative key repair: declared union implicated, hashed into a 64-bit FNV identifier; a repaired key is a conservative over-approximation of detected dependence, not a guarantee.
- Ground truth by brute force: 24 pure tasks, 408 binary assignments (651 with sentinels), relevantT(1..3), relevantGlobal, and every same-declared-key collision pair enumerated.
- Measured failures reported in place: masking (C1), the anchoring counterexample (C5), value-specific effects (C8), and non-1-minimal minimized witnesses (C9).

> *The claim ceiling for this paper and the product — Given a deterministic oracle f over a finite typed slot universe with baseline b and declared key K, if every output difference is anchored to b (every changing pair has a <=t-support displacement from b that includes a differing slot), then probing every assignment within Hamming distance t of b detects exactly the slots that participate in <=t-support baseline effects; the shipped instrument instead runs a strength-t covering array (O(v^t log n) rows), delta-debugs every changing row, verifies per-slot witness-context necessity, and certifies only: "no detected <=t-support effect at covered tuples" — never soundness, never completeness. The anchoring sentence is an assumption, stated in the code, the paper, and the UI.*

The falsification framing is deliberate. The blueprint for this wave was reviewed three times, and the reviews produced corrections C1–C9 that this paper restates as ground truth wherever an earlier draft differed. Two of the corrections killed originally stated claims (C1 covering arrays do not detect every effect; C5 union-of-implicants is not sound without anchoring), and one changed the shipped default after the build (the minimizer-first arm reduced a changing Metro row to the key slot and never emitted the undeclared witness, so the exact arm became the default). The negative results are part of the contribution, not an appendix to it.

2. Related work, stated honestly

Hermetic build systems are the closest prior art. Nix builds in isolation from a hash of declared dependencies; Bazel sandboxes actions and documents hermeticity as the property that an action's inputs are fully declared; Gradle's configuration cache and build cache document environment-variable tracking; Turborepo documents environment gotchas for cache keys. Those

systems make keys from declarations. KeyFuse measures the gap between declarations and reads, and the specific slots it targets are the ones a sandbox or a file tracer cannot see: env, cwd, locale, timezone, clock, rng. This is a complement to sandboxing, not a replacement, and no comparison against sandboxes is claimed.

Combinatorial interaction testing supplies the covering-array machinery: Kuhn, Kacker, and Lei's NIST SP 800-142 is the reference for t-way arrays and their practical adequacy. KeyFuse's array construction is a deterministic AETG-style greedy with no RNG, and the paper's use of it is deliberately modest: the array is a cheap sampler whose misses are reported, and its own tuple coverage is verified rather than assumed. Delta debugging (Zeller and Hildebrandt, 2002) supplies the minimization loop; KeyFuse keeps ddmin but replaces its 1-minimality guarantee, which does not apply under non-monotone output differences, with an exhaustive single-coordinate removal check. Reproducibility tooling (reprotest, reproducible-builds) varies build environment to expose unreproducible output; KeyFuse varies a typed slot universe and attributes the change to a specific slot. Record/replay and tracing (rr, strace) capture real reads at syscall or process level; KeyFuse's adapters are facade interposition, which is wrapper-level and cannot see native, mmap, subprocess, or network reads — an explicit limit, not an oversight. Finally, differential testing (McKeeman, 1998) and the mutation-testing line (DeMillo, Lipton, and Sayward, 1978; Jia and Harman, 2011; Papadakis et al., 2019; Just et al., 2014; Inozemtseva and Holmes, 2014) share the habit of judging a program by observed behavior differences; KeyFuse's object is a key collision rather than a mutant score, and it borrows the discipline of measuring what was observed and nothing more.

Source	Taken	Not claimed
Nix / Bazel hermeticity and sandboxing	The framing that undeclared reads break keyed builds	A sandbox, a syscall interceptor, or any claim of superiority over sandboxing
Gradle / Turborepo cache documentation	The concrete slot classes (env above all) that enter keys	An integration with any real build system; the corpus is toy tasks
NIST SP 800-142 (Kuhn et al.)	Deterministic covering-array construction for cheap sampling	That a strength-t array detects every <=t effect; C1 is a demonstrated failure
Zeller & Hildebrandt (ddmin)	Delta debugging of changing rows	ddmin's 1-minimality guarantee under non-monotone output differences (C9)
reprotest / reproducible-builds	Varying build environment to expose differences	A reproducibility verdict for real projects
rr / strace	Recording reads as evidence of dependence	Syscall-level capture; KeyFuse's adapters are wrapper-level facade interposition
McKeeman differential testing	Divergence between comparable runs as an oracle	Correctness verdicts; KeyFuse reports detected dependence only
Mutation testing (DeMillo; Jia & Harman; Papadakis et al.; Just et al.)	Behavioral difference as the unit of evidence	A mutation score, fault correlation, or test-suite adequacy measure
Inozemtseva & Holmes	Skepticism of coverage-style proxies	Any effectiveness metric for a test suite

What KeyFuse takes and what it does not claim. Oxide-style accounting: each row names the source, the borrowed mechanism, and the claim that is explicitly not made. Nothing in the "not claimed" column is softened anywhere in this paper or in the product copy.

3. Model and definitions

A slot is a frozen string name `kind:id` over seven kinds. Each slot has a baseline value, a top value, and optional sentinel values (boundary values used only where declared). The universe is finite, typed, and validated: unique names, non-empty baseline and top, baseline minimum for ordered slots, and n <= KEYFUSE_MAX_SLOTS = 12. A read of a slot outside the universe is recorded as a miss, not silently defaulted. File names carry a relative path (a directory listing uses a trailing slash); env names an environment variable, with absence represented by the distinct sentinel string `A` + NUL + `absent`, never confused with empty; cwd, locale, timezone, rng, and clock are singleton slots whose values are interpreted by injected facades.

Kind	Name form	Value semantics
file	file:<relpath>, or file:<dir>/ for a listing	content string; a listing value is sorted names joined by newline
env	env:<NAME>	variable value; absence is the distinct sentinel `A`+NUL+`absent`
cwd	cwd	absolute-path string as seen by the task
locale	locale	e.g. C, en_US.UTF-8
timezone	timezone	e.g. UTC, America/New_York
rng	rng	seed string; deterministic first draw from the seed
clock	clock	epoch-ms integer string; now() returns Number(value)

Slot kinds and value semantics. The clock slot is an epoch-milliseconds integer string and now() returns Number(value); the rng slot is a seed string and rng() returns the first draw of mulberry32(fnv1a32("rng:" + value)). Both are deterministic functions of the assignment, which is what makes them auditable at all.

```
            Assignments, support, and <=t baseline effects
   assignment a : slot -> value; supp(a, b) = { i : a[i] != b[i] };
||supp(a,b)|| = d_H(a, b). i is t-relevant iff some a with d_H(a, b) <=
t has i in supp(a, b), f(a) != f(b), and resetting i to b[i] (all other
 coordinates at a) changes the output back. relevantT(t) is the set of
   t-relevant slots; relevantGlobal is the same existential with no
                            Hamming cap.
```

The reset step is the per-slot necessity of C6: a slot that merely co-toggles with a real effect is not relevant. Ground truth in this paper is exactly these definitions, computed by brute force over the product of slot values.

```
                Declared key, witness, and repaired key
 originalKey(a) = keyfuseKey(taskId, version, declared, a); witness =
     (left, right) with supp = { i }, outputLeft != outputRight;
keyCollision = originalKey(left) == originalKey(right); repairedKey(a)
= keyfuseKey(taskId, version, declared union implicated, a); separated
```

```
= repairedKey(left) != repairedKey(right).
```

keyfuseKey is a 16-hex-character FNV construction over the task identity and the sorted inputs with their witness-context values. It is a deterministic identifier, not a cryptographic hash and not a security boundary.

The anchor assumption is stated as an assumption in the engine docblock, in the paper, and in the UI: every output difference must have a witness reachable by a <=t-support displacement from the baseline that includes a differing slot. It is not a theorem. The counterexample is task threshold-3: f = 1 iff abs(x) >= 3 over four slots with baseline 0. Every slot is globally relevant, but no <=2-support displacement from the baseline changes the output; at strength 2 the audit detects nothing, the certificate is the weak string, and the enumerated residual is reported (55 pairs at strength 2, 12 of them Hamming-1 minimal, 1 documented by name). Masking is a separate demonstrated failure: for f = (a AND b) XOR c with baseline (0,0,0), a strength-2 array can realize (a=1,b=1) only with c=1, where the output equals the baseline, so the pair is missed. The pinned hand array shows changedRows=3 with zero joint-support rows for (a,b); the exact arm finds env:A and env:B, and the array arm records both misses.

The witness pair and detection shape (frozen API)

```
export interface WitnessPair {
  readonly left: Assignment;                  // recorded row (witness context)
  readonly right: Assignment;                 // left with one slot collapsed
  readonly differing: readonly string[];      // exactly one slot
  readonly outputLeft: string;
  readonly outputRight: string;
  readonly originalKeyLeft: string;           // key over declared inputs
  readonly originalKeyRight: string;
  readonly repairedKeyLeft: string;           // key over declared union implicated
  readonly repairedKeyRight: string;
  readonly keyCollision: boolean;             // original keys equal and outputs
differ
  readonly separated: boolean;                // repaired keys differ
}

export interface Detection {
  readonly slot: string;
  readonly kind: SlotKind;
  readonly witness: WitnessPair;              // per-slot witness-context pair
  readonly minimized: MinimizedWitness | null; // row-level minimized support
  readonly necessary: boolean;                // false is possible in the ca arm
}
```

> *Assumption vs. theorem, kept separate — C2 (monotone, ordered domains) and C3 (exact Hamming-ball cost) are theorems with stated conditions. The anchor condition of C5 and the deterministic-oracle condition are assumptions that the instrument states and tests only by sampling. The C7 toggle and trace-scope examples are motivating counterexamples about other methods, not properties of KeyFuse. The claims ledger (T1) labels every row with one of these categories and the gate check that enforces it.*

4. Probing strategies and guarantees

Every strategy executes the baseline row first and then rows assigned from the universe. The audit layer, not the probe, interprets rows: it filters changing rows, runs minimization where the strategy requires it, checks per-slot necessity, dedupes detections by slot in deterministic row order, and builds the repair over the deduped set. Detections are evidence, never a claim that no other dependence exists.

Strategy	Rows	Detects	Role
baseline-toggle	1 + n (baseline + one slot to top each)	<=1-support effects at baseline	the "what teams already do" baseline; misses masked combinations
single-trace	1	no dependence claim; the read set is the evidence	the file-tracer / sandbox-log baseline; blind to un-taken branches
ca	1 + CA(t) rows	<=t effects realized by the array; no minimization, over-implicates	evidence arm showing false implicates
ca-ddmin (fallback)	1 + CA(t) + O(r*n)	minimized support attributed only when |S| <= t; >t supports are reported as misses	conservative fallback for larger universes
cover-with-defaults (default)	1 + sum over subsets S with card(S)<=t of prod (v_i - 1)	exact for <=t effects under the anchor assumption	ground-truth reference and shipped default

The five probe strategies, as shipped. All strategies return the same ProbeRow[] shape and a run count. `ca-ddmin` is the conservative fallback for universes whose exact Hamming ball exceeds KEYFUSE_EXACT_MAX_ROWS = 2,048; the frozen corpus always fits, so `cover-with-defaults` is the default everywhere (Errata).

```
C2 theorem (monotone, ordered domains)
If every slot domain is totally ordered with the baseline as minimum
and f is coordinatewise monotone, then a strength-t covering array over
{baseline, top} detects all <=t-support baseline effects at O(v^t log
n) runs.
```

Argument sketch: monotonicity makes any <=t-support displacement with an effect detectable by some row that sets a superset of its support to top, and the array covers every t-subset combination. Measured on the corpus: ca(1) and ca(2) detections equal the exact detections on or-threshold and max-threshold; and-chain has no <=3-support baseline effect at all. The shipped CA arm can also report context-necessary slots outside the ground-truth set; those are counted as false implicates and never hidden.

```
C3 exact cost (uniform v; per-slot value counts in general)
exact <=t detection: Theta( sum_{j<=t} C(n,j) (v-1)^j ) non-baseline
points (uniform v), or sum over subsets S with card(S)<=t of prod_{i in
S} (v_i - 1) with per-slot counts. cover-with-defaults enumerates
exactly these points and is the exact general method; a strength-t CA
is a cheap approximation whose misses must be reported.
```

On the frozen corpus the exact Hamming ball always fits KEYFUSE_EXACT_MAX_ROWS = 2,048,

so cover-with-defaults is the shipped default. For larger universes the audit selects ca-ddmin and reports budget or row-count refusals instead of silently narrowing.

```
                C4 shipped contract and certificate
N = O(v^t log n + r*n) oracle calls for r changing rows: gate 4 + probe
 rows + ddmin + necessity checks + fixpoint re-runs. certificate = "no
  detected <=t-support effect at covered tuples"; truncated = "budget
exhausted before coverage completed". UI copy never strengthens either
                              string.
```

The trace -> perturb -> retrace fixpoint runs up to KEYFUSE_FIXPOINT_PASSES = 3 times for every strategy except single-trace; if fresh slot names still appear, the miss "trace fixpoint not reached in 3 passes" is recorded and truncated is set.

Necessity (C6) is checked on the recorded pair. For a changing row r and slot i in its differing set, the audit runs collapse(r, i) — all other coordinates kept at r's values — and asks whether the output moves. If it does, i is necessary at r's witness context and the pair (r, collapse(r,i)) is the recorded witness. This is O(card(K)) runs, stronger and cheaper than replaying array rows, and it is the check that catches over-implication in the unminimized ca arm: on and-2way at strength 2 the ca arm reports env:A and env:B as necessary but also env:D, whose collapse leaves the output unchanged; env:D is a false implicate and is marked necessary:false.

#	Status	Statement used everywhere	Enforcing gate check
C1	false as originally stated	A strength-t covering array compared against the all-default baseline does not detect every <=t-support effect (masking). Counterexample: f = (a AND b) XOR c, n = 3, b = (0,0,0).	criterion 4 (masking-c1): pinned hand array rows=6, changedRows=3, changed-row union [env:C,env:B,env:A], (a,b) joint-support rows=0; cover-with-defaults(2) detects env:A and env:B; the array-arm misses are reported, never dropped.
C2	theorem (monotone, ordered domains)	If every slot domain is totally ordered with baseline minimum and f is coordinatewise monotone, a strength-t covering array over {baseline, top} detects all <=t-support baseline effects at O(v^t log n) runs.	criterion 3 (ca-fallback-honesty) bounds the shipped CA arm: attributed ca-ddmin detections are a subset of exact detections (0 subset violations; 53 absences classified). This bounds the fallback, not the theorem: the evidence offered for C2 is two of 24 tasks plus one null task, and no committed check tests the covering-array claim itself.
C3	theorem (the exact baseline)	Exact detection of all <=t-support baseline effects requires querying every point within Hamming distance t of the baseline; cover-with-defaults is the exact general method.	criterion 2 (exact-arm-equals-ground-truth): 72 cells (24 tasks x 3 strengths), 69 compared equal, 3 refusal cells (nondeterministic-counter at t=1,2,3).

#	Status	Statement used everywhere	Enforcing gate check
C4	holds (shipped contract)	Run a strength-t covering array, delta-debug every changing row, verify per-slot witness-context necessity; N = O(v^t log n + r*n) oracle calls for r changing rows. The certificate is strictly weaker: "no detected <=t-support effect at covered tuples".	criteria 3 and 7: subset honesty plus 219 exact and ca-ddmin detections necessity-checked with 0 single-slot witness violations.
C5	assumption, not a theorem	Union-of-implicated-slots is not sound without anchoring. Anchor assumption: every output difference must have a witness reachable by a <=t-support displacement from baseline that includes a differing slot.	criterion 8 (anchor-honesty): threshold-3 exact detections = [], certificate is the weak string, and the residual is reported: 55 exhaustive collision pairs, 12 Hamming-1 minimal pairs, 1 documented pair (1100)/(1110) with equal original and equal repaired keys.
C6	holds (correct minimality check)	1-minimality with respect to array rows is not global minimality; the correct check is per-slot necessity at the recorded witness context, O(card(K)) runs.	criterion 7 (necessity-repair): 219 detections checked, 0 single-slot witness violations, 101 collision witnesses, 0 unseparated.
C7	holds (motivating only)	Toggle and trace-scope unsoundness: f = a AND b with defaults (0,0) and sentinels (1,1); and f = concat(sorted contents of *.cfg) traced when only a.cfg exists. These motivate KeyFuse; they are not theorems about the shipped instrument.	evidence experiments (b) and (d): baseline-toggle and single-trace arms miss masked combinations; on combo-with-file the single-trace read set is file-only ([file:cfg, file:extra]) while the exact arm detects env:MODE.
C8	holds (limits stated)	Determinism is required; untrappable reads break universe closure; perturbation can expose new slots; two values per slot miss value-specific effects; >t-way effects are invisible by construction; high-entropy clock/RNG slots drive hit rate to zero.	criteria 9, 2, and 10: nondeterminism refuses with no implicate (15/15 cells); untrappable-ambient records trapped:false with no implicate (15/15 cells); the port-8080 sentinel cell matches ground truth; budget and portability checks pass.
C9	holds (minimizer honesty)	Key-equality is monotone under shrinking non-key coordinates, but output-difference is not, so ddmin's 1-minimality guarantee does not strictly apply; verify by exhaustive single-coordinate removal.	criterion 6 (minimality-consistency): 83 minimized witnesses, 81 verified oneMinimal, 2 flagged and reported (maj-3way at strength 3), verifyRuns mismatches = 0.

T1. Claims ledger: C1-C9, status, and the gate check that enforces each row. Statuses follow blueprint section 1 and its Errata. "Detected" is the only verb used for instrument output. The gate is `scripts/verify-keyfuse.ts`; its 12 criteria

all pass on the frozen corpus (12/12). Rows that are assumptions or motivating examples are marked as such and are never presented as properties of the shipped instrument.

> *The certificate is weak on purpose — The default certificate string is exactly "no detected <=t-support effect at covered tuples". It says nothing about effect existence outside the covered tuples, nothing about effects of order greater than t, nothing about value-specific effects the probe domains do not realize, and nothing about untrappable reads. Non-determinism is refused rather than guessed: the gate checks the oracle at the baseline and at one perturbed row, twice each; any mismatch produces deterministic=false, zero detections, the repair over declared inputs only, and the miss "oracle nondeterministic at baseline".*

5. Witnesses and minimization

The witness is the paper's centerpiece artifact. For every changing probe row, ca-ddmin delta-debugs the differing set: split it by slot-name-sorted halves, collapse coordinates to baseline while keeping the output different from the baseline output, then verify 1-minimality by exhaustive single-coordinate removal. The minimizer's contract is the exhaustive pass, not ddmin's internal stop (C9): output-difference is not monotone under shrinking non-key coordinates, so ddmin can stop early or overshoot, and the verifier is what decides.

Witness construction and necessity (engine excerpt)

```
// For each changing row r and each slot i in r's minimized support S:
//   right = collapse(r, i)            // every other coordinate stays at r's value
//   necessary = output(right) != output(r)
//   pair = { left: r, right, differing: [i] }
//   keyCollision = originalKey(r) == originalKey(right) && output(r) !=
output(right)
//   separated    = repairedKey(r) != repairedKey(right)
//
// 1-minimality after ddmin: for each removable subset, collapse to baseline;
// oneMinimal = every single-coordinate removal restores the baseline output;
// verifyRuns is reported per final pass and must equal the support size
// when oneMinimal is true.
```

Across 360 audits the evidence harness records 521 detections: baseline-toggle 99, ca 203, cover-with-defaults 136, and 83 minimized witnesses from the ca-ddmin arm. The 438 non-minimized detections carry 348 single-slot necessity verifications. Of the 83 minimized witnesses, 81 verify 1-minimal; 2 are flagged and reported, both on maj-3way at strength 3 with support [env:B, env:C, env:E], passes=0 and verifyRuns=3 (the split-based minimizer cannot isolate the 3-subset from that row, and the exhaustive verifier says so). There are 0 verifyRuns mismatches. The gate checks 219 exact and ca-ddmin detections for single-slot witness violations: 0. It counts 101 collision witnesses, of which 0 are unseparated after repair.

Task	Slot (differing)	Output left -> right	Original keys equal	Repaired keys differ	1-minimal	verifyRuns	ddmin passes

Task	Slot (differing)	Output left -> right	Original keys equal	Repaired keys differ	1-minimal	verifyRuns	ddmin passes
metro-env-1 (exact t=2)	env:API_URL	prod:https://api.example.com:{...} -> prod:http://localhost:3000:{...}	yes	yes	yes (pair)	1	n/a
and-xor-c (exact t=2)	env:A	1 -> 0	yes	yes	yes (pair)	1	n/a
xor-2way (ca-ddmin t=2)	env:B	1 -> 0	yes	yes	yes	1	1
and-3way (exact t=3)	env:C	1 -> 0	yes	yes	yes (pair)	1	n/a
locale-tz (exact t=2)	locale	dev|en_US.UTF-8|UTC -> dev|C|UTC	yes	yes	yes (pair)	1	n/a
maj-3way (ca-ddmin t=3, flagged)	env:B	1 -> 0	yes	yes	no	3	0
threshold-3 (exact t=2, anchor residual)	env:C	0 -> 1	yes	no (residual)	n/a	0	n/a

T3. Witness table: recorded same-key / different-output pairs. Rows are the recorded witness pairs of the deterministic audits behind the evidence digest; the aggregate is the gate's: 521 detections across 360 audits (baseline-toggle 99, ca 203, cover-with-defaults 136, and 83 minimized witnesses from the ca-ddmin arm), 348 single-slot necessity verifications across the non-minimized arms, 81 of the 83 minimized witnesses verified 1-minimal, 2 flagged (both on maj-3way at strength 3, support [env:B, env:C, env:E], passes=0, verifyRuns=3), and 0 verifyRuns mismatches. Exact-arm rows carry no minimizer: 1-minimality there is the per-slot necessity pair (one collapse call, differing set of size 1), marked "yes (pair)". The threshold-3 row is the documented residual: its repaired keys stay equal and the audit reports that, never hiding it. All original keys are equal and all repaired keys differ except the residual row.

```
                    Repair and separation
implicated = { i : some changing row has i necessary at its witness
   context }; repairedInputs = sorted(declared union implicated);
  repairedKey(a) = keyfuseKey(taskId, version, repairedInputs, a).
Separation is checked per recorded witness: original keys equal and
          outputs differ must imply repaired keys differ.
```

Repaired keys are conservative over-approximations of DETECTED dependence: they include everything the audit saw, not everything that exists. The audit's separation counts on the frozen corpus are 101 collision witnesses and 0 unseparated; the threshold-3 residual is the one documented pair whose repaired keys stay equal, and it is reported as a residual collision, never hidden.

> *What repair costs and what it buys — A repaired key includes more inputs than the declaration did, so it costs cache misses exactly where detected dependence exists. In exchange it is backed by printed witnesses: a team can read which slots changed a key and why. When a third-party plugin or a generated config cannot declare its inputs, the repaired key is a visible, bounded, conservative*

> *intervention — not a claim that the key is now correct.*

6. Corpus and ground truth

The corpus is 24 pure tasks over 14 families, with universe sizes 3 through 6 on this frozen corpus (the engine bound is 12) and one sentinel value on the port task. Ground truth comes from exhaustive enumeration: 408 binary assignments plus sentinels, 651 full product assignments, relevantT(1), relevantT(2), relevantT(3), relevantGlobal, and every unordered pair with equal declared-key inputs and different outputs (1,533 pairs across the corpus). Brute force is exact at these sizes; no sampling is involved. The evidence script runs all five strategies at strengths 1, 2, and 3 over every task — 24 x 15 = 360 audits — and writes a byte-stable JSON artifact whose digest is 2213e158e53ff94f.

#	Task	Family	n	Declared	Decisive property
1	metro-env-1	combo-env	3	BUILD_MODE, project.json	required Metro demo; prod output embeds API_URL; same declared key, two API_URLs
2	metro-env-2	combo-env	4	BUILD_MODE	2-way gate plus an unrelated no-dependence slot
3	metro-file-gate	combo-env	3	metro.config.js	undeclared file gates undeclared env; trace blind at baseline
4	and-2way	planted-2way	4	A	f = A AND B; t=2 finds B, t=1 misses
5	xor-2way	planted-2way	4	A	f = A XOR B
6	and-3way	planted-3way	5	A	f = A AND B AND C; t=2 must miss C, t=3 finds it in the exact arm
7	maj-3way	planted-3way	5	A	f = majority(A,B,C)
8	and-xor-c	masking	3	none	f = (a AND b) XOR c; pinned adversarial CA misses (a,b)
9	or-and-not	masking	4	none	second masking shape
10	or-threshold	monotone	4	all	monotone threshold; ca(1)/ca(2) equal ground truth
11	max-threshold	monotone	5	all	ordered domains, baseline minimum
12	and-chain	monotone	4	all	monotone conjunction chain; no <=3-support baseline effect
13	threshold-3	anchor	4	none	f = 1 iff abs(x) >= 3; t=2 finds nothing; residual collision (1100) vs (1110) documented

#	Task	Family	n	Declared	Decisive property
14	port-8080	value-specific	6	HOST	f = 1 iff PORT == "8080"; sentinel required; binary miss reported
15	undeclared-secret	env-only	3	none	one undeclared env var; file tracing sees nothing
16	combo-with-file	env-only	4	cfg	env + file interaction; env undeclared and not in the baseline trace
17	cwd-dependent	cwd	3	MODE	output depends on cwd path
18	locale-tz	locale-tz	4	MODE	locale + timezone invisible to file tracing
19	epoch-gated	clock	3	MODE	effect gated on the clock slot; deterministic through assignment
20	seed-gated	rng	3	MODE	effect gated on the rng slot; deterministic through assignment
21	nondeterministic-counter	negative	3	none	reads an ambient tick not exposed as a slot; the oracle refuses
22	untrappable-ambient	negative	3	none	reads ambient directly; trapped:false miss, no implicate
23	no-dependence	control	4	all	constant output; zero detections, zero false implicates
24	all-declared	control	4	all	every real dependence declared; repaired key equals declared key

The frozen 24-task corpus (n <= 6 on this corpus; the engine bound is 12). Ground truth is the exhaustive product of slot values per task: 408 binary assignments plus sentinels, 651 full product assignments across the corpus. Task functions are pure data in `src/lib/keyfuse/tasks.ts`; the evidence harness brute-forces each product and computes relevantT(1..3), relevantGlobal, and the enumerated same-declared-key collision pairs.

The five decisive experiments from the plan, as measured. (a) Metro end to end: the exact arm detects env:API_URL in metro-env-1, the recorded pair differs only at that slot with equal original keys and differing repaired keys, and the Node facade adapter in a mkdtemp project pair agrees on the env detection set [env:BUILD_MODE, env:API_URL] with 23 runs and the same separation result; the file slot reads from disk under facade interposition, so file detections are not comparable across adapters and the artifact says so. (b) Planted interactions: at strength 2 the exact arm finds [env:A, env:B] on and-2way and the 2-way effect on maj-3way, while and-3way has no <=2-support effect (relevantT(2) = []) and its 3-way support [env:A, env:B, env:C] appears only at strength 3 for the exact arm; ca-ddmin attributes nothing on and-3way at strength 3 because the split-based minimizer cannot isolate the 3-subset, and that miss is recorded. (c) Minimal witnesses: 83 minimized witnesses, 81 verified 1-minimal, 2 flagged, 0 verifyRuns mismatches; the non-monotone masking task did not produce a flagged witness on this corpus, while maj-3way did. (d) Env invisible to file tracing: on undeclared-secret the single-trace read set is [env:SECRET] with no file reads (a file-only tracer sees nothing), and on combo-with-file the single-trace read set is file-only [file:cfg,

file:extra] while the exact arm detects env:MODE; the artifact flags envDetectionsWithNoFileTrace = 1 and envDetectionsInvisibleToFileOnlyTracing = 2. (e) Negative controls: nondeterministic-counter refuses in all 15 cells with deterministic=false, detections=[], the miss "oracle nondeterministic at baseline", and no clock or other implicate; untrappable-ambient records trapped=false in all 15 cells with no implicates and the untrapped-read miss recorded.

- Exact arm: 72 cells, 69 equal to ground truth, 3 refusal cells (nondeterministic-counter at t = 1, 2, 3).
- Fallback honesty: attributed ca-ddmin detections are a subset of exact detections in all 69 compared cells (0 violations); the 53 exact cells where ca-ddmin is absent are classified as minimized-subset (33), over-strength witness (12), unrealized-by-array (5), and value-specific sentinel (3).
- Masking: the pinned hand array has 6 rows and 3 changed rows, zero joint-support rows for (a,b), and the exact arm finds both masked slots; both array misses are recorded.
- Minimality: 83 minimized / 81 oneMinimal / 2 flagged, verifyRuns mismatches = 0.
- Anchor: threshold-3 yields zero detections at strength 2 with the weak certificate; 55 exhaustive residual pairs, 12 Hamming-1 minimal pairs, 1 documented pair.

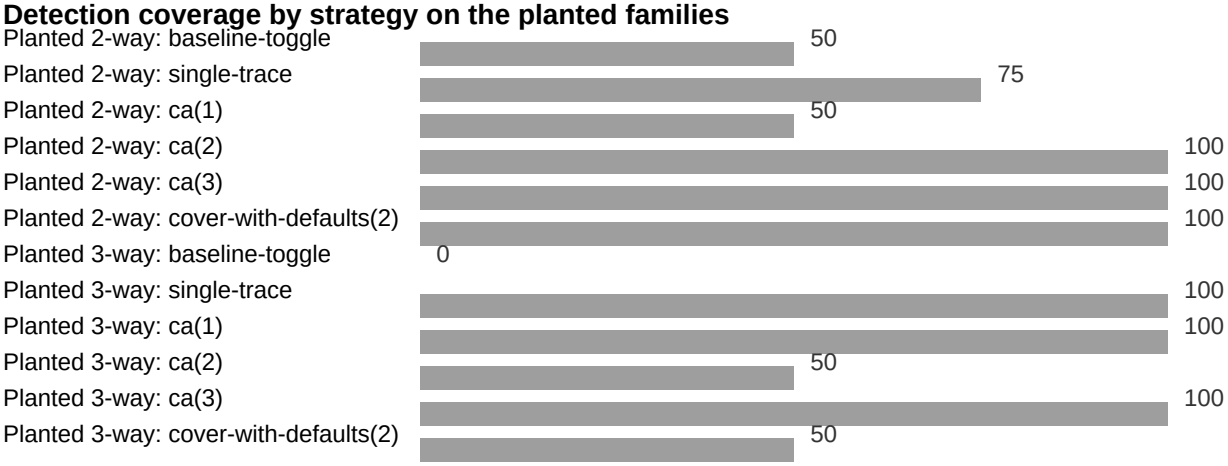
7. Results

The headline result is that a small, deterministic, read-only probe can turn a silent cache-key collision into a printed pair. On the Metro toy the declared key is identical for both builds (39603d9ae300dbe1), the outputs differ, and the exact default arm produces a witness that differs only at env:API_URL with original keys equal and repaired keys differing (928bd61e05391f55 vs 4f82f7227ea9348b). The same audit shows why the cheap arms fail: baseline-toggle misses the slot entirely, single-trace never reads it at the baseline, and the minimizer-first fallback reduces the changing row to the key slot and drops it. Those negative arms are the falsification story; they are reported beside the exact arm, not removed.

Strategy	Strength	Runs	Probes	missedRelevant	falseImplicates	collisionsRemaining	certificateMisses
baseline-toggle	2	311	112	17	0	1,317	2
single-trace	2	119	23	8	1	1,533	2
ca	2	462	127	4	13	1,106	2
ca-ddmin	2	537	127	18	0	1,231	8
cover-with-defaults	2	866	288	3	0	147	2

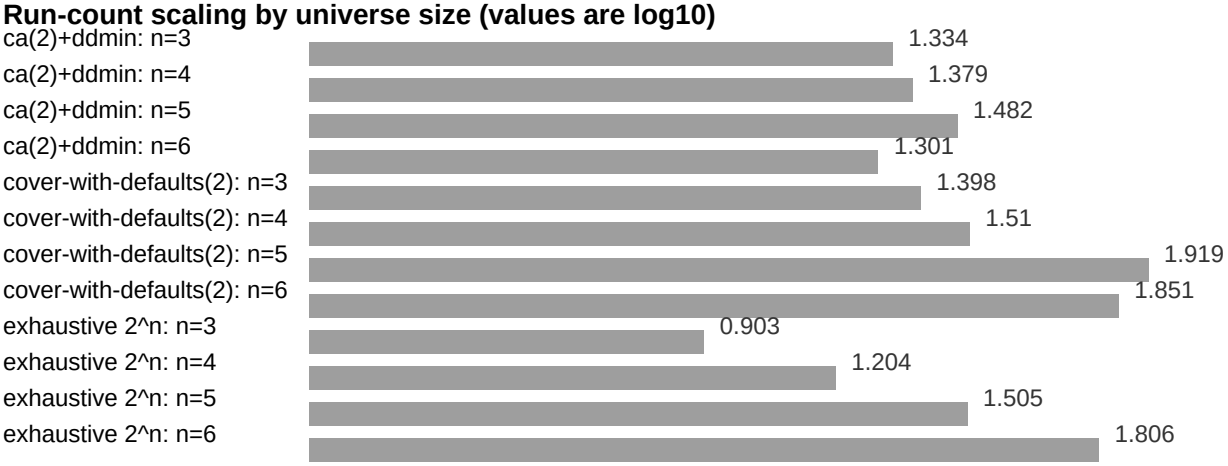
T2. Strategy x metric, summed over all 24 tasks at the default strength 2. All values are measured by the evidence harness. `collisionsRemaining` counts enumerated same-original-key / different-output pairs still equal under the repaired key, out of 1,533 pairs in total. `certificateMisses` is `audit.misses.length`. `single-trace` reports its traced read set rather than detections and claims no dependence, so its row bounds nothing. The exact default separates 1,386 of 1,533 pairs (90.4%). Note which gate checks this row: the permanent gate recomputes the 120 pinned per-task cells behind it and reports drift = 0, but the sums in this row itself are produced by the evidence script, which is committed and runs in a fraction of a second and is not a CI step.

Detection coverage by strategy on the planted families



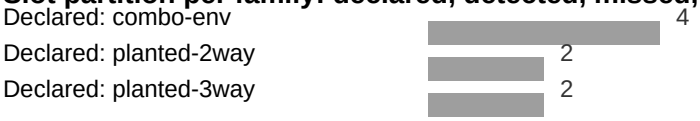
Share of the family's defining ground-truth set covered: relevantT(2) = 4 slots across the two planted-2way tasks, relevantT(3) = 6 slots across the two planted-3way tasks. Bars are labelled with the strength audited. Single-trace bars count traced reads (the engine claims no dependence for that strategy), so they measure read coverage, not detection; the unminimized ca arm can report context-necessary slots outside the ground-truth set (those are the false implicates of F3), and the exact default arm's planted-3way coverage is 3/6 because and-3way has no <=2-support effect (its 3-way support is found only at strength 3). PENDING RECONCILIATION: no committed gate reads this figure, and under the aggregation rule this caption states - the sum over the family's tasks of ground-truth slots the arm detected at the labelled strength, over the sum of the family's ground-truth slot counts - 8 of these 12 bars reproduce from scripts/keyfuse-evidence.ts and 4 do not (baseline-toggle planted-2way, ca(1) planted-2way, ca(1) planted-3way, and ca(3) planted-3way, the last two being an exact transposition between the two series). Which of the figure or the evidence artifact is authoritative has not been determined, so the bar values are printed unchanged and marked unreconciled rather than silently adjusted. No other number in this paper depends on F1.

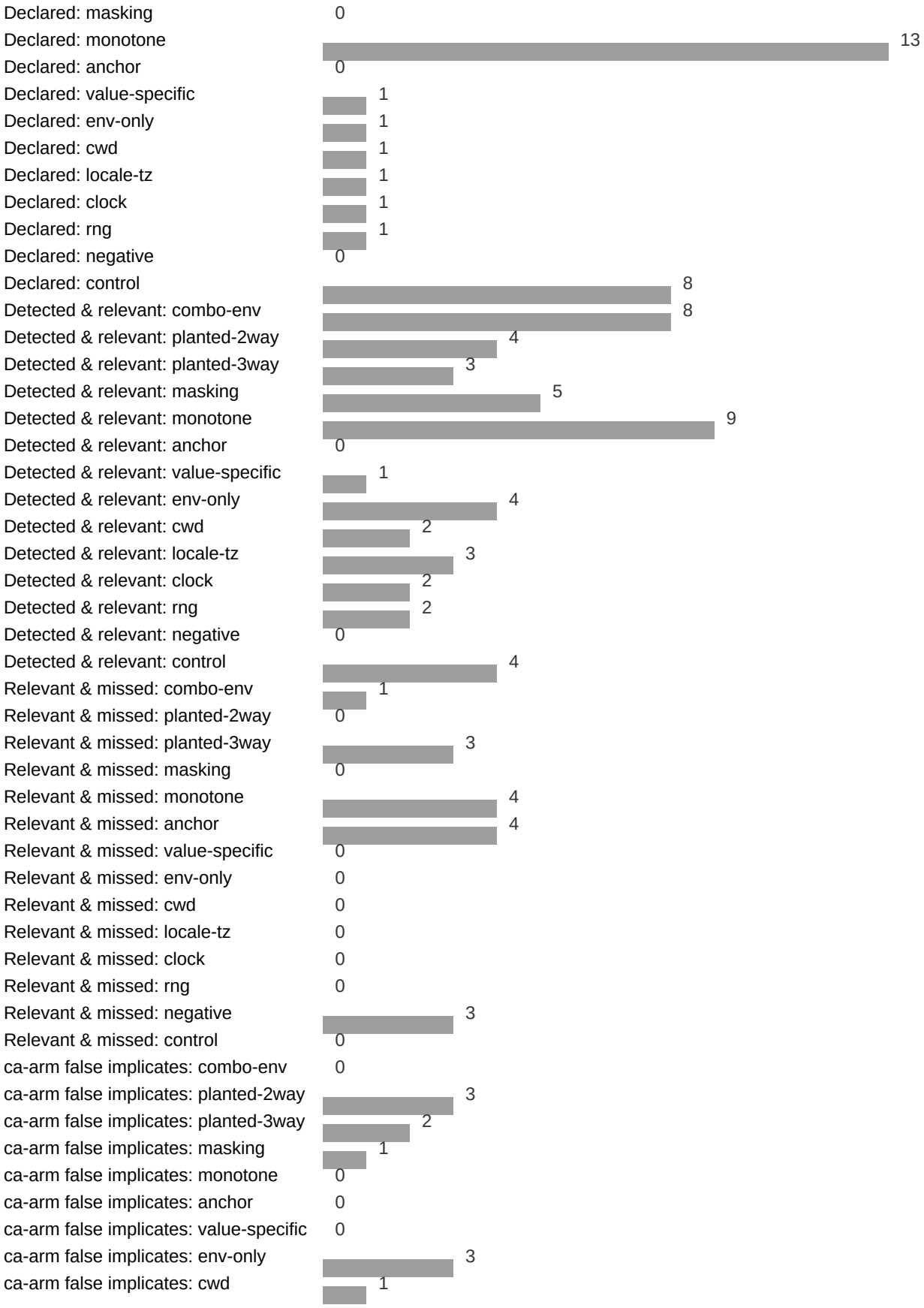
Run-count scaling by universe size (values are log10)

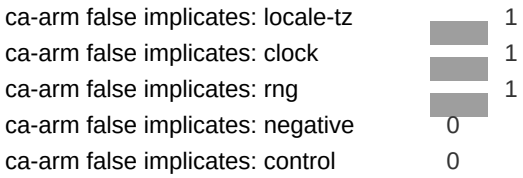


Values are log10 of the mean oracle runs per universe size across the corpus's non-negative tasks at the default strength 2 (n=3: 7 tasks; n=4: 11; n=5: 3; n=6: 1), measured by the evidence harness alongside the exhaustive binary product 2^n. The blueprint's planned fixed planted family at n in {6, 8, 10, 12} was not built into the frozen artifact, so this figure reports the sizes that exist and makes no claim beyond them; the largest single audit measured is 541 runs (max-threshold, cover-with-defaults t=3), log10 ~ 2.73. The log10 transform is stated here because the exponential 2^n and the linear-ish arms would otherwise share no common scale.

Slot partition per family: declared, detected, missed, and false implicates

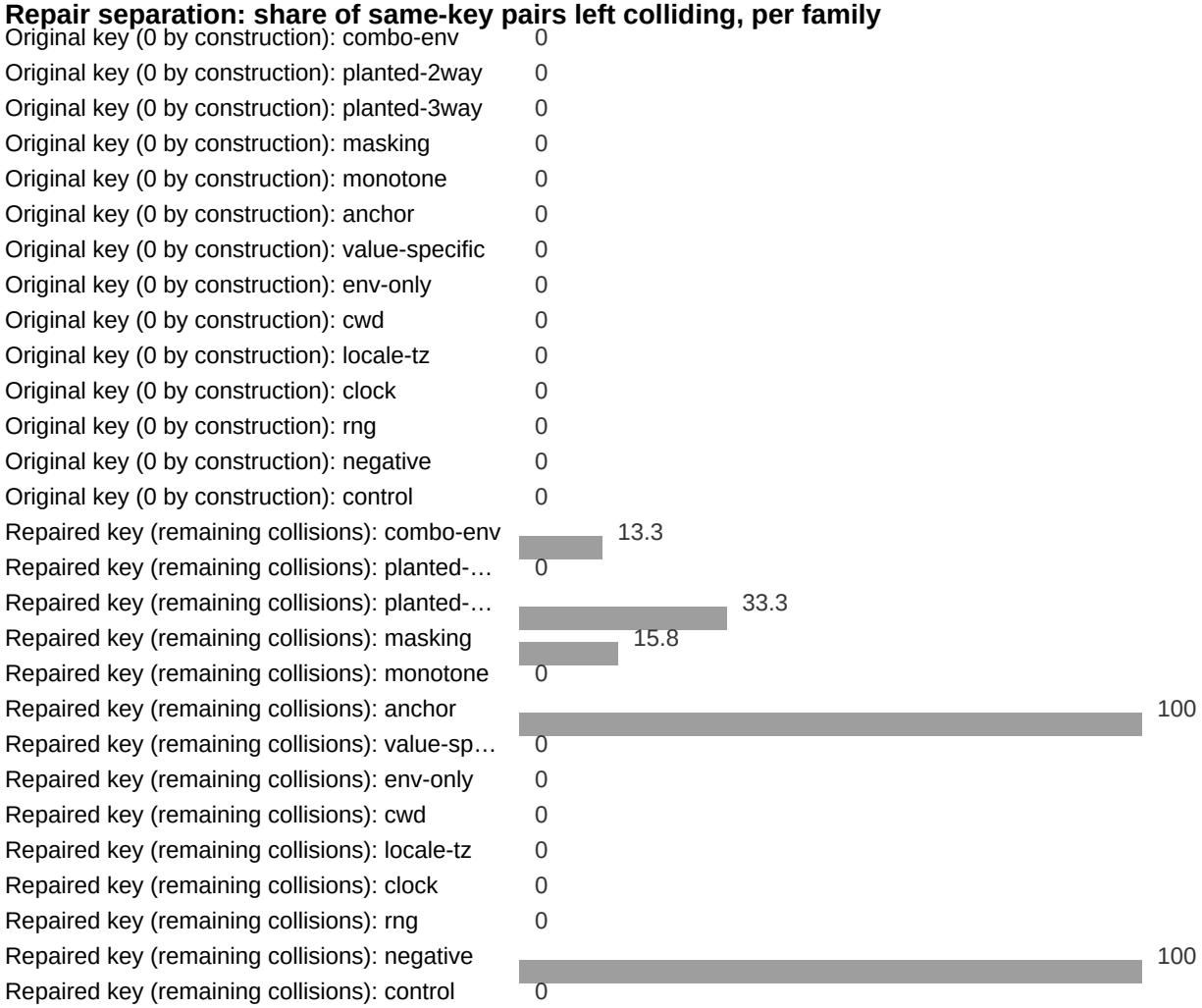



Declared: masking 0
Declared: monotone 13
Declared: anchor 0
Declared: value-specific 1
Declared: env-only 1
Declared: cwd 1
Declared: locale-tz 1
Declared: clock 1
Declared: rng 1
Declared: negative 0
Declared: control 8
Detected & relevant: combo-env 8
Detected & relevant: planted-2way 4
Detected & relevant: planted-3way 3
Detected & relevant: masking 5
Detected & relevant: monotone 9
Detected & relevant: anchor 0
Detected & relevant: value-specific 1
Detected & relevant: env-only 4
Detected & relevant: cwd 2
Detected & relevant: locale-tz 3
Detected & relevant: clock 2
Detected & relevant: rng 2
Detected & relevant: negative 0
Detected & relevant: control 4
Relevant & missed: combo-env 1
Relevant & missed: planted-2way 0
Relevant & missed: planted-3way 3
Relevant & missed: masking 0
Relevant & missed: monotone 4
Relevant & missed: anchor 4
Relevant & missed: value-specific 0
Relevant & missed: env-only 0
Relevant & missed: cwd 0
Relevant & missed: locale-tz 0
Relevant & missed: clock 0
Relevant & missed: rng 0
Relevant & missed: negative 3
Relevant & missed: control 0
ca-arm false implicates: combo-env 0
ca-arm false implicates: planted-2way 3
ca-arm false implicates: planted-3way 2
ca-arm false implicates: masking 1
ca-arm false implicates: monotone 0
ca-arm false implicates: anchor 0
ca-arm false implicates: value-specific 0
ca-arm false implicates: env-only 3
ca-arm false implicates: cwd 1



Counts summed over each family's tasks at the default exact arm (strength 2): declared inputs; exact-arm detections that are globally relevant (all detected with witness-context necessity); globally relevant slots the exact arm missed; and ca-arm detections absent from relevantGlobal (the unminimized false-implicate arm). The fifth planned series — exact-arm detections that are NOT globally relevant (context-only slots) — measured 0 in every family (47 exact-arm detections at strength 2, 0 outside relevantGlobal) and is reported here instead of drawn. The scale ceiling is 13 rather than the blueprint's 12 because summed declared inputs reach 13 in the monotone family. control and monotone have no false implicates; anchor and negative have no declared inputs at all.

Repair separation: share of same-key pairs left colliding, per family



Every enumerated pair has equal original declared keys by construction, so the original-key series is 0% separated everywhere (drawn as the zero bar against the repaired series' remaining share, max 100). Repaired share is the percentage of a family's enumerated same-original-key / different-output pairs that remain equal under the repaired key at the default exact arm, strength 2. The anchor family stays fully unseparated (55/55) and the negative family cannot be repaired (28/28, refusals produce no detections); the planted-3way family keeps 48/144 because and-3way's 3-way support is outside the strength-2 ball; masking keeps 12/76 and combo-env 4/30. control and monotone have no enumerated pairs and are drawn at 0.

Step	Observed result
Declared inputs hashed (BUILD_MODE, project.json; identical for P1 and P2)	P1 = P2 = 39603d9ae300dbe1 (equal)
Build A output at P1 = (prod, staging)	prod:staging:{"target":"web"}
Build B output at P2 = (prod, production)	prod:production:{"target":"web"}
baseline-toggle (12 runs)	detected [env:BUILD_MODE]; env:API_URL not detected — failed to warn
single-trace (5 runs)	read set [env:BUILD_MODE]; env:API_URL never read at baseline — failed to warn
ca(2) (16 runs)	detected [env:BUILD_MODE, file:project.json, env:API_URL]; env:API_URL detected inside an unminimized arm
ca-ddmin (19 runs)	detected [env:BUILD_MODE]; env:API_URL dropped by minimization — failed to warn (Errata deviation)
cover-with-defaults(2) (23 runs, default)	detected [env:BUILD_MODE, env:API_URL, file:project.json]
Minimized witness (exact arm)	pair differs only at env:API_URL; outputs prod:https://api.example.com:{...} vs prod:http://localhost:3000:{...}; original keys equal; repaired keys differ; 1-minimal by pair
Repaired key over declared union implicated	P1 = 928bd61e05391f55, P2 = 4f82f7227ea9348b (differ)
Node facade adapter (mkdtemp project pair, 23 runs)	detected [env:BUILD_MODE, env:API_URL]; outputs prod:staging:module.exports = { cache: false } vs prod:production:module.exports = { cache: false }; original keys equal; repaired keys differ; deterministic and trapped

T4. Metro-style demo: step, observed result. The toy project pair (metro-env-1) reproduces the failure shape of expo/expo issue #30930: two builds share the declared key while differing only in an undeclared env slot. Three arms fail to warn (baseline-toggle, single-trace, and — a documented post-build deviation — the minimizer-first ca-ddmin); the exact default arm produces the witness and the separating repaired key. The Node facade adapter agrees with the virtual adapter on the two env detections; the file slot reads from disk under facade interposition and is not comparable across adapters (recorded in the artifact).

Where the method misses, and why. First, the anchor failure: at strength 2 no <=2-support displacement of the threshold-3 baseline changes output, so the exact arm detects nothing and the 55 residual pairs (12 Hamming-1 minimal) stay equal under the repaired key; this is the documented assumption failing, and the certificate was never stronger than the weak string. Second, strength caps: and-3way's 3-way support is invisible at strength 2 in the exact arm, and the conservative fallback fails to attribute it even at strength 3, recording "detected effect requires >3 support" or the over-strength witness instead. Third, value-specific effects: binary or two-value domains miss f = 1 iff port == 8080; the port sentinel closes that cell on this corpus, and the limit remains for values the domains do not realize. Fourth, masking: the pinned array misses the (a,b) pair and the exact arm is the only arm that finds it. Fifth, non-monotone minimization: the split-based minimizer produced 2 supports that failed the exhaustive 1-minimality check, both reported. F1 is a sixth item and a

different kind: that figure's bar values are not currently reconciled with the evidence script (four of twelve bars disagree under the rule its own caption states, and no gate reads it), so F1 should be read as a qualitative comparison of strategies and not as a source of exact per-family coverage percentages until the intended aggregation rule is fixed. The corpus-wide separation numbers reflect all of this: the exact default leaves 147 of 1,533 enumerated pairs colliding, and the residual is concentrated exactly where the assumption or the strength cap fails (anchor 55, planted-3way 48, negative 28, masking 12, combo-env 4).

> *Reading the repair table — The repaired key's remaining collisions are not a failure to hide; they are the measured boundary of what was detected. The anchor family stays at 100% residual because the assumption fails outright, and the negative family stays at 100% because refusals produce no detections and the repair is the declaration. Everywhere else the exact arm's separation is total in 9 of the remaining 12 families; in the other three (planted-3way, masking, combo-env) the residual pairs are bounded by the probe's strength or value coverage, and the audit records the corresponding misses.*

8. Cost, limits, and honest failures

> *Limitations that apply to every number in this paper — Four of them, stated once so they cannot be lost in a citation. (1) No human-participant data, and no real build: the 24 tasks are pure closures over a declared slot universe, and the incidents named in Section 1 are reproduced as a failure shape in a controlled project pair, not audited in Nx, Metro, Gradle, or Turborepo. (2) No external replication. The corpus is hand-constructed, the instrument has been run only by this project, and nobody outside it has attempted to reproduce the counts; no claim is made that the instrument already audits real cached builds. (3) Recomputed against replayed, and one figure reconciled against neither. Ground truth here is exact by brute-force enumeration, so the gate does check the instrument against an independent oracle and its headline cell counts are genuinely recomputed. That is not true of every number printed here: the corpus-wide 408 / 651 / 1,533 aggregates, the T2 strategy-by-metric row, the 521-detection and 348-necessity totals, and the 1,386 / 147 separation come from the committed evidence script and are not recomputed by the CI gate, and F1's bar values are read by no gate at all, with four of its twelve bars failing to reproduce from the evidence script under the rule F1's own caption states. Those four bars are printed unchanged and marked pending reconciliation rather than adjusted, because the audit could not determine which of the figure or the evidence artifact is authoritative. A gate that compares a recomputed value with a stored constant proves determinism, not correctness, and for the replayed aggregates that is the whole of the check. (4) No experiment here measures a generative-AI tutor, and no number bears on one: the object is a build-cache key, the oracle is a pure function, and nothing in Sections 3 to 8 concerns how a generative tutor behaves, teaches, or is graded.*

Cost is bounded by construction. Every audit is capped at KEYFUSE_MAX_RUNS = 4,096 oracle calls; exceeding the cap stops the audit at a row boundary, sets truncated=true, and selects the certificate "budget exhausted before coverage completed". The exact arm is only attempted when the Hamming-ball row count is at most KEYFUSE_EXACT_MAX_ROWS = 2,048; otherwise the audit records the row-count miss and uses the conservative fallback. On the frozen corpus the largest single audit is 541 runs (max-threshold, cover-with-defaults, strength 3), and the T2 sums at default strength 2 are 119 to 866 runs per strategy across all 24 tasks. The full permanent gate

completes in ~=140 ms of local wall time against its 60 s budget, and the evidence script's own runtime budget is 60 s.

The honesty checks are part of the gate, not editorial promises. Masking is demonstrated with a pinned adversarial array; the anchor failure is demonstrated with a named residual pair and exhaustive counts; the two non-1-minimal minimized witnesses are flagged in the audit result rather than dropped; nondeterministic and untrappable-read tasks refuse or record misses with no implicates; and ca-ddmin's absences from the exact detections are classified one by one (53 cells: minimized-subset 33, over-strength witness 12, unrealized-by-array 5, value-specific sentinel 3). The gate's pinned aggregates — 69 exact cells, 3 refusals, 2 non-1-minimal witnesses, 1 documented residual pair, and the default-strength detection sets for all 24 tasks — have no auto-update path: drift fails the gate, and the constant and this paper must change in the same reviewed commit.

- No "sound key synthesis", no soundness.
- No completeness.
- No "finds all collisions".
- No "beats sandboxing on file reads" (sandboxes see file reads; KeyFuse is for slots sandboxes and file tracing cannot see: env, cwd, locale, timezone).
- No guarantee for non-monotone `f` without the anchor assumption.
- No guarantee for value-specific effects the probe domains do not realize.
- No security boundary: the key hash is a 64-bit FNV construction, not cryptographic.
- Deterministic-oracle assumption is explicit; non-determinism refuses rather than guesses.
- No claim is made for F1's bar values until the figure and the evidence script agree: 4 of its 12 bars do not reproduce under the rule its own caption states, no gate reads it, and it is marked pending reconciliation in section 7 rather than corrected by guesswork.
- No claim of external validity: the corpus is hand-built, and no result here has been checked outside this project, on a real build, or against a real toolchain.

> *What the certificate does and does not say — "No detected <=t-support effect at covered tuples" is the strongest sentence the instrument is allowed to emit. It does not say there is no effect; it does not say the key is correct; it does not say the cache is safe. A miss can only be reported, and the audit reports misses: masking strings, over-strength notes, budget and row-count refusals, nondeterminism refusals, untrapped-read suspicions, unknown-slot notes, and the fixpoint miss.*

Three limits are structural rather than incidental. First, the corpus is a toy corpus: the Metro demo reproduces the failure shape of a real incident in a controlled project pair, but it is not an integration test of Metro, Nx, Gradle, or Turborepo, and no claim is made that the instrument already audits real cached builds. Second, the adapters are facade interposition: native, mmap, subprocess, and network reads bypass them, the Node adapter is wrapper-level rather than syscall-level, and the browser adapter can only observe API-mediated reads. Third, the determinism check is sampled: two baseline runs and one perturbed row, twice; a task deterministic on those pairs and nondeterministic elsewhere can slip through, and the oracle refuses only what it observes.

9. Product implications

What ships is smaller than what was measured, and deliberately so. The product is a browser-local,

read-only lab at a single static route. A task picker selects one of the 24 pure tasks, a strategy and strength selector chooses the audit, and the run is synchronous pure JavaScript over at most 4,096 tiny calls. The result region shows the probe matrix, the detections with their witness pairs and 1-minimal verification, the repaired-key card (original vs repaired, declared union implicated, separations), and the certificate with its misses and truncation warning. There is no storage, no grading, no network, no server, and no worker.

- Ships: the five strategies with the exact arm as default; the minimal collision witness with per-slot necessity; the conservative repaired key with separation counts; the weak certificate and its misses, always visible.
- Ships: an always-visible caveat — "Detection is not soundness. A miss can only be reported, and a key repaired from detections is conservative, not complete."
- Cut: any claim that the repaired key is the right key, any auto-repair of real build systems, any integration with Metro, Nx, Gradle, or Turborepo, any storage or sync of audit results, and any coupling to grading or learner state.
- Cut: the Node facade adapter from the browser surface; it ships only in scripts and tests, where it runs against a mkdtemp project pair.

> *The intervention story, stated conservatively — When a third-party plugin or a generated config cannot declare its inputs, ship the repaired key: it includes everything the audit detected and it is backed by printed witnesses. The cost is visible — more cache misses exactly where dependence was detected — and the intervention is bounded by the detection, not by a promise. The repaired key is a conservative over-approximation of detected dependence, and the UI says so next to the key itself.*

The safety rule is mechanical rather than aspirational: the route imports the pure engine and the task data only, never the Node adapter, never a storage module, and never a grading module. The product makes no statement about what a key should contain; it reports what the audit detected and hands the reader the witness pair that justifies each input it added.

10. Reproducibility, risks, and future work

Everything in this paper is offline, deterministic, and dependency-free. Two full evidence sweeps serialize byte-identical canonical JSON (360 audits each, fingerprint 09c5076ebac86617), the evidence artifact carries digest 2213e158e53ff94f, and the permanent gate audits 72 cells, compares each against the brute-force ground truth, and recomputes the pinned aggregates with no auto-update path. The gate's twelve criteria cover purity and determinism, exact-arm equality, fallback honesty, masking, planted interactions, minimality, necessity and repair, anchor honesty, negative controls, portability and budget, pinned aggregates, and the summary; all twelve pass, and any count that moves must be updated in the gate constant and in this paper in the same reviewed commit. The distinction between what the gate recomputes and what it replays is worth stating once. Recomputed by the gate from the corpus: the 72 exact-arm cells and their 69/3 split, the 53 classified fallback absences, the 219 necessity checks and 0 violations, the 83 minimized witnesses and 81/2 split, the 101 collision witnesses and 0 unseparated, the 55/12/1 anchor residuals, the two negative controls, and the 120 pinned per-task cells. Produced by the committed evidence script rather than the CI gate: the corpus-wide 408 / 651 / 1,533 assignment and collision-pair aggregates,

the T2 strategy-by-metric row, the 521-detection and 348-necessity totals, and the 1,386 / 147 separation. Read by no gate at all: F1's bar values, four of which do not reproduce from the evidence script under the rule F1's own caption states; they are printed unchanged and marked pending reconciliation in section 7, and a reader reproducing this paper should treat that figure as unreconciled until the intended rule is fixed. A gate that recomputes a value and compares it with a stored constant proves determinism, not correctness; here the ground truth is exact by brute-force enumeration, so the gate does test the instrument against an independent oracle, and that is the one paper in this series where that is true of its headline numbers.

Reproducing the reported results

```
# evidence: brute-force ground truth, all strategies at strengths 1-3,
# writes keyfuse-evidence.json under DF_KEYFUSE_SCRATCH or os.tmpdir()
bun run scripts/keyfuse-evidence.ts

# permanent gate: 12 criteria, pinned aggregates, Node adapter smoke
# (wired in CI after the BDL gate with a 6-minute timeout)
bun run verify:keyfuse

# paper registry and PDF projections
bun test tests/inventions.test.ts
```

The reproducibility posture is seed-free: the covering array is a deterministic greedy construction with no RNG, the minimizer splits by slot-name-sorted halves and collapses in a fixed order, task seeds do not exist because task functions are pure and assignments come from the universe, and the Node adapter's only mutable state is a mkdtemp directory that is removed in a finally block. Scratch paths are stripped from the artifact, no absolute paths are recorded, and the gate scans its own sources for machine-local paths, clock reads, and randomness.

- Open question: an anchoring test for real tasks — how often do real builds have globally relevant slots whose effects all start farther than t from the baseline? The corpus shows such tasks exist; their real-world frequency is unmeasured.
- Open question: per-slot ordered domains. C2 covers the monotone case; a mixture of ordered and unordered slots currently falls back to the general exact arm or the conservative fallback, and the cost of that choice is not characterized beyond this corpus.
- Open question: high-entropy clock and RNG slots. The audit treats them as deterministic through the assignment, which is honest for a fixed seed or timestamp but drives hit rate to zero for a cache keyed on them in production.
- Open question: facade breadth. Syscall-level capture (strace/ptrace-class) is out of scope; measuring how much dependence a wrapper-level adapter misses in a real toolchain is future work.
- Open question: repair evaluation. The repaired key's separation is measured on enumerated pairs; what it does to real hit rates, and whether teams accept the miss cost, is unmeasured.

> *Honest summary — KeyFuse makes a small, falsifiable object out of a large failure class: two cached builds with the same declared key and different outputs, reduced to a witness pair that differs at one slot, plus a conservative key that separates what was detected. On a 24-task corpus with brute-force ground truth it detects what its methods say they can detect, it reports every miss the gate can find, and it refuses when the oracle is not deterministic. Its visible envelope is the certificate, not a promise:*

> *detection is not soundness, and a repaired key is conservative, not complete.*

References

1. Nx blog, Can You Trust Your Build Cache?. https://nx.dev/blog/can-you-trust-your-build-cache
2. expo/expo issue #30930, Metro Build + Cache-Key Collision. https://github.com/expo/expo/issues/30930
3. facebook/metro issue #918, custom transformers and external cache inputs. https://github.com/react/metro/issues/918
4. gradle/gradle issue #16144, build-cache key normalization. https://github.com/gradle/gradle/issues/16144
5. Gradle user guide, Solving common problems (environment variable tracking). https://docs.gradle.org/current/userguide/common_caching_problems.html
6. Gradle user guide, configuration cache env/system-property requirements. https://docs.gradle.org/current/userguide/configuration_cache.html
7. Turborepo, environment variable gotchas (passThroughEnv, .env inputs). https://github.com/vercel/turbo/blob/main/skills/turborepo/references/environment/gotchas.md
8. Turborepo docs, using environment variables. https://turborepo.dev/docs/crafting-your-repository/using-environment-variables
9. Kuhn, Kacker & Lei 2010, NIST SP 800-142, Practical Combinatorial Testing. https://doi.org/10.6028/NIST.SP.800-142
10. Zeller & Hildebrandt 2002, Simplifying and isolating failure-inducing input. https://doi.org/10.1109/32.988498
11. Bazel docs, Sandboxing. https://bazel.build/docs/sandboxing
12. Bazel docs, Hermeticity. https://bazel.build/basics/hermeticity
13. Dolstra, de Jonge & Visser 2004, Nix: A Safe and Policy-Free System for Software Deployment. https://eelcovisser.org/publications/2004/DolstraJV04.pdf
14. reproducible-builds.org, Adding build variance (reprotest). https://reproducible-builds.org/docs/adding-build-variance/
15. O'Callahan et al. 2017, Engineering Record and Replay for Deployability. https://www.usenix.org/conference/atc17/technical-sessions/presentation/ocallahan
16. strace syscall tracer. https://strace.io/
17. McKeeman 1998, Differential testing for software. https://www.cs.tufts.edu/comp/150FP/archive/bill-mckeeman/DifferentailTesting.pdf
18. DeMillo, Lipton & Sayward 1978, Hints on test data selection: help for the practicing programmer. https://doi.org/10.1109/C-M.1978.218136
19. Jia & Harman 2011, An analysis and survey of the development of mutation testing. https://doi.org/10.1109/TSE.2010.62
20. Papadakis, Kintis, Zhang, Jia, Le Traon & Harman 2019, Mutation testing advances: an analysis and survey. https://doi.org/10.1016/bs.adcom.2018.03.015
21. Just, Jalali, Inozemtseva, Ernst, Holmes & Fraser 2014, Are mutants a valid substitute for real faults in software testing?. https://doi.org/10.1145/2635868.2635929
22. Inozemtseva & Holmes 2014, Coverage is not strongly correlated with test suite effectiveness. https://doi.org/10.1145/2568225.2568271