Can Several Minimal Findings Reveal a Hidden Attribute?

An Exactly Solvable Pilot on Linked, Coarsened and Task-Output Releases, with a Release-Audit Toolkit

Anish Prashun
Independent researcher

Independent technical report Version 1.0 November 2026
DOI: 10.5281/zenodo.24304575

Abstract

Multi-stage workflows, including AI-assisted ones, often pass small sets of structured findings between stages instead of raw records. We ask how much they reveal about an attribute that none of them states, in a synthetic testbed where every ceiling is computed exactly. A fictional household workflow releases intake (A) and follow-up (B) findings while a three-level reserve band stays hidden. On the same 3,000 test households we compare A, B, both releases linked (C), a declared coarsening of C (D) and the nine report-checklist answers alone (E), using attackers trained on 6,000 labels, exact Bayes and information references, and a deterministic utility measure. The directions of the main effects follow from the construction; their sizes are the result. Linking raised naive-Bayes accuracy by 0.081 over A and 0.068 over B (exact ceilings 0.517, 0.527 and 0.604; baseline 0.400), and one cross-release pair ruled out a band for 5.75% of households. The coarsening changed accuracy by -0.018 [-0.031, -0.006] on the pre-specified seed, but met this confirmatory criterion in only 3 of 6 seed runs; the power analysis had used the wrong reference gap. The coarsening kept 80.5% of action items and still leaked more than either single release. The checklist answers kept every action item and still leaked (ceiling 0.565); whether they leak less than a coarsening depends on the checklist. A standard-library toolkit computes these quantities, a per-field information budget and a leakage-utility frontier from a declarative specification. This extends classic composition results; no language model is involved.

1 Introduction

Workflows that handle personal data are often split into stages, and a common minimisation practice is for each stage to pass downstream a few structured findings rather than the raw record. In an AI-assisted workflow, an intake step may emit a handful of categorical findings, a follow-up step a few more, and a later step composes a report from them. Each set can look harmless. The question is what several of them reveal together about an attribute that none of them states, and what a designer can do about it without breaking the downstream task.

The question is old: composition attacks, sequential releases and the utility cost of generalisation are established (Section 2). This report is a focused extension and illustration of those results, not a discovery. It applies the classic comparison of single, linked and generalised releases, plus one alternative that the classic composition papers do not compare, releasing only the task's outputs, to structured findings passed between workflow stages. The AI-workflow setting is motivation only: no language model produces, transforms or attacks anything here.

What the study adds is a testbed in which the answers are computed exactly: the Bayes-optimal accuracy of every view, its mutual information with the hidden attribute, and the accuracy of naive Bayes with unlimited training data come from enumerating a small, fully specified generator. That separates what follows from the construction from what is measured. By construction, the directions of the linking hypothesis (H1) and of the coarsening loss (H2a) hold at the Bayes ceiling, because adding a release cannot lower it and the coarsened view and the task outputs are functions of the linked view; so do the utility ordering (H2b), the confinement of exclusions to the linked view (S3), the absence of unsupported defaults where the strict renderer never abstains (S6), E's full utility, and "E leaks no more than C". What is measured is the size of each effect under a documented generator, how close learned attackers with finite labels come to the exact references, and how fragile a pre-specified test is.

Contributions.

1. An exactly solvable testbed in which every ceiling is computed rather than estimated, logical exclusion is reported apart from statistical inference, and utility is a deterministic task checklist split into supported and unsupported answers.
2. Effect sizes under a documented generator, from a protocol frozen before test scoring (self-attested), including a confirmatory result (H2a) that holds on the pre-specified seed but meets its criterion in only 3 of 6 seed runs.
3. An exploratory comparison showing that task outputs dominate the linked fine release but are not ordered against a coarsening in general.
4. `release_audit`, a standard-library toolkit that computes the same exact quantities, a per-field information budget and a release-design frontier for a user's own specification.

2 Related work

Composition and sequential release. Ganta et al. [1] showed that independently anonymised releases about overlapping populations can be combined: their intersection attack intersects the candidate sensitive values left by each release and breaches k-anonymity and its variants, whereas definitions in the style of differential privacy provably resist composition attacks. Wang and Fung [3] showed that joining a new release with an earlier one about the same people can enable sensitive inference, and generalised the new release so that the join becomes lossy while each release keeps its own information requirement; Xiao and Tao [4] showed similar failures under repeated publication. Our view C is the joined release. Our exclusions are not an instance of the intersection attack, which intersects the sensitive values each release leaves possible: neither of our releases rules out a band on its own, and the exclusion arises only when the two are joined, as in Wang and Fung's observation that the join "enables inferences across tables". Unlike Wang and Fung, we assume the linkage is known, so coarsening gets no credit for hiding the join.

Generalisation, information and baselines. Generalising values into coarser categories is a standard remedy [6, 7]. Brickell and Shmatikov [5] measured privacy loss as the adversary's gain in learning sensitive attributes and utility as data-mining accuracy on the same records, and concluded that "even modest privacy gains require almost complete destruction of the data-mining utility". Kifer [8] attacked partition-based sanitisation with a naive Bayes classifier, our main attacker's model family. Query-view security asks whether published views disclose a secret query [9], and

quantitative information flow bounds the leakage of composed channels, including parallel channels on one secret [10]. Attribute inference should be compared with an imputation baseline that has the same distributional knowledge [11]; we report every leak as a gain over the frequency baseline and call what we measure distributional inference. Our utility is a fixed downstream checklist, which is what makes the task-output alternative (E) definable.

Language-model systems. Language models infer personal attributes from text [13, 14]. Patil et al. [15] studied multi-agent systems in which a hidden attribute emerges only from combining individually innocuous outputs (single-source leakage is zero by construction, and the defences block or refuse); Asif and Amiri [16] formalised compositional leakage along sequential agent pipelines with mutual information. Zhou et al. [17] searched for the least revealing transformation of a single prompt, including abstraction, that passes a utility check, without studying several releases; ToolPrivacyBench [18] counts private facts that reach unauthorised tools, by presence rather than inference; AgentDAM [19] and AirGapAgent [20] evaluate or enforce data minimisation for agents. We measure graded gains over single releases and use the checklist as the utility predicate.

Position. A literature search found no study that runs exactly this comparison on the same held-out individuals with exact Bayes references and a separate deterministic utility measure. The nearest recent work we found treats sequential releases to colluding receivers information-theoretically, with mutual-information leakage, a separate utility and optimised randomised releases [2]; it does not compare coarsened or task-output releases or report Bayes-accuracy ceilings. The search was not exhaustive, and every ingredient is established.

3 Setting and threat model

Each synthetic household has a hidden reserve band: `low` (liquid reserve under one month of essential costs), `medium` (one to three months) or `high` (more than three). Release 1 (intake) has five categorical fields, release 2 (follow-up) has four, and a downstream step fills a nine-item report checklist. The band exists only in the scoring data, there is no free text, and every probability is invented.

- **View and linkage.** The attacker sees only what a downstream system would receive in a given condition (Section 4). In the linked conditions it is told that both releases belong to the same household; how hard linkage is in practice is out of scope. In the study's own data files the release-1 and release-2 views carry the same opaque household identifier: a stable key across stages is exactly the linkage assumed.
- **Knowledge.** 6,000 labelled training households from the same generator, in the same representation: a strong, upper-bound attacker for its model family (Section 7.8 adds weaker ones). It never sees the band (except in a labelled positive control), seeds, label-correlated file order or any other condition's representation; identifiers are opaque HMAC values and test rows are shuffled.
- **Isolation.** Each attacker runs as a separate `python3 -I` process in a fresh, empty directory, with an otherwise empty environment and its input on standard input only: process-level isolation, not an operating-system sandbox.

Leakage is a gain over a frequency baseline that predicts the most common band. *Statistical inference* shifts the attacker's posterior; *logical deduction* occurs when a released value has probability

zero under some band, which it then *excludes* (it *determines* the band if only one remains).

4 Generator and conditions

Table 1: Generator of the main scenario ($\lambda = 1$). Cells give $P(\text{value} \mid \text{band})$ in the listed value order. The resolution is `not_applicable` exactly when there is no event. The 0.00 for a low-band household settling a major expense in full is the one hard zero. Every probability is invented.

Field	Values, in order	`low`	`medium`	`high`
Prevalence π		0.40	0.35	0.25
`r1.income_pattern`	P(`irregular`)	0.50	0.35	0.25
`r1.essential_share`	`under_50`, `50_70`, `over_70`	0.15, 0.40, 0.45	0.30, 0.45, 0.25	0.50, 0.35, 0.15
`r1.event_code`	`none`, `vehicle_minor`, …, `home_major`	Independent of the band: $P(\text{event}) = 0.50$, type uniform, $P(\texttt{major} \mid \text{event}) = 0.40$		
`r1.payment_flag`	`none`, `late_utility`, `overdraft_fee`, `both`	0.45, 0.20, 0.20, 0.15	0.65, 0.15, 0.12, 0.08	0.85, 0.07, 0.06, 0.02
`r1.contact_channel`	`email`, `phone`, `post`	Independent of the band: 0.60, 0.30, 0.10		
`r2.resolution`, minor event	`settled_in_full`, `instalment_plan`, `outstanding`	0.60, 0.25, 0.15	0.75, 0.18, 0.07	0.85, 0.12, 0.03
`r2.resolution`, major event	same	0.00, 0.45, 0.55	0.25, 0.55, 0.20	0.80, 0.17, 0.03
`r2.payment_flag`	as `r1.payment_flag`	0.50, 0.18, 0.18, 0.14	0.70, 0.13, 0.10, 0.07	0.88, 0.06, 0.05, 0.01
`r2.goal`	`none`, `build_buffer`, `reduce_debt`, `major_purchase`	0.25, 0.35, 0.30, 0.10	0.30, 0.25, 0.25, 0.20	0.35, 0.10, 0.15, 0.40
`r2.document_status`	`complete`, `missing_id`, `missing_income_proof`	Independent of the band: 0.70, 0.15, 0.15		

Table 1 gives the generator. Income pattern, essential-cost share, both payment flags (conditionally independent given the band) and the savings goal carry moderate additive band information. The event's size (release 1) and its resolution (release 2) interact: size alone carries no band information, resolution alone some, and a major expense settled in full is strong evidence of a high reserve. One hard constraint is planted: a low-reserve household cannot settle a major expense in full, so the pair (`major`, `settled_in_full`) excludes `low`, and only the linked fine view shows the pair. Contact channel, document status and the event's occurrence and type carry no band information but are needed for the report. Raw households take 65,664 distinct value combinations with positive probability.

Scenarios. Every scenario has its own 10,000 households, split by household into 6,000 training, 1,000 development and 3,000 test households. Sensitivity scenarios mix each conditional distribution toward its marginal, $P_\lambda(f \mid b) = (1-\lambda)M(f) + \lambda P(f \mid b)$ with $M(f) = \sum_b \pi(b)P(f \mid b)$, at $\lambda = 0$ (`independence`, a control), 0.25, 0.50 and 0.75; two more keep $\lambda = 1$ and make the prevalence uniform (`prev_balanced`) or 0.70, 0.20, 0.10 (`prev_skewed`). Exclusions can occur only when $\lambda = 1$. A positive control plants the band as a field in both releases. The main scenario is regenerated from five further master seeds; with the primary run these give six seed runs, each with its own households, split and training.

Conditions. Every condition is computed for the same test households: **A**, release 1 only; **B**, release 2 only; **C**, both releases linked as one record; **D**, C after a declared coarsening; and **E**, the nine checklist answers only (Table 2), the minimal output the report needs, pre-specified as a

secondary condition. The coarsening renames each field it changes so that granularity is explicit, and drops no field: `under_50` and `50_70` shares become `up_to_70`; the event code loses its size (`medical_major` becomes `medical`); each payment flag becomes `none` or `any`; and `settled_in_full` and `instalment_plan` become `arranged`. Income pattern, contact channel, goal and document status pass through unchanged.

Table 2: The report checklist. Correct answers come from the full raw record. The last column is E's exact input for the worked example of Table 3.

Item	Values	Correct answer	Example
`budget_session`	yes, no	yes iff either payment flag is not `none`	yes
`overdraft_guidance`	yes, no	yes iff either flag is `overdraft_fee` or `both`	yes
`essential_cost_review`	yes, no	yes iff the share is `over_70`	no
`expense_referral`	`none` or a type	the event type iff an event occurred and its resolution is `instalment_plan` or `outstanding`	`medical`
`hardship_review`	yes, no	yes iff a major event is `outstanding`	no
`plan_monitoring`	yes, no	yes iff the resolution is `instalment_plan`	yes
`goal_module`	`none` or a goal	the goal	`build_buffer`
`documents_request`	`none`, `id`, `income_proof`	from the document status	`income_proof`
`contact_channel`	`email`, `phone`, `post`	the channel	`phone`

Table 3: Worked example: hand-checked household H1 of the protocol (true band `medium`). Each condition receives exactly these key-value pairs as one JSON line: A the release-1 rows, B the release-2 rows, C all nine, D the right-hand columns, and E the answers in Table 2. The last row counts the strict renderer's correct items out of nine.

Fine field (A or B, and C)	Value	Field in D	Value in D
`r1.income_pattern`	`regular`	`r1.income_pattern`	`regular`
`r1.essential_share`	`50_70`	`r1.essential_share_coarse`	`up_to_70`
`r1.event_code`	`medical_major`	`r1.event_type`	`medical`
`r1.payment_flag`	`overdraft_fee`	`r1.payment_difficulty`	`any`
`r1.contact_channel`	`phone`	`r1.contact_channel`	`phone`
`r2.resolution`	`instalment_plan`	`r2.resolution_coarse`	`arranged`
`r2.payment_flag`	`none`	`r2.payment_difficulty`	`none`
`r2.goal`	`build_buffer`	`r2.goal`	`build_buffer`
`r2.document_status`	`missing_income_proof`	`r2.document_status`	`missing_income_proof`
Strict renderer, items correct of 9: A 4, B 4, C 9, D 6, E 9			

In the worked example (Table 3), D's `any` hides whether the release-1 flag was an overdraft fee and `arranged` hides whether there is an instalment plan, so the report cannot settle overdraft guidance, the referral or plan monitoring. A second hand example shows `home_major` in release 1 and `settled_in_full` in release 2: neither release excludes anything alone, the linked pair excludes `low`, and D cannot, because coarsening removes the size.

5 Attackers, references, renderers and measures

Attackers. All attackers receive the training rows and labels of one view and the test rows of the same view, and none abstains. Arithmetic is exact, and ties break in the order low, medium, high. `baseline` predicts the most frequent training band. `nb` is categorical naive Bayes over all visible fields, with an add-1 smoothed prior and add-α smoothing ($\alpha = 1$) per field over the values seen in training plus one slot for unseen values. `nb_pair` is `nb` with one schema-linked cross-release pair joined into a single feature, (`event_code`, `resolution`) in C and (`event_type`,

`resolution_coarse`) in D; elsewhere it equals `nb`. `lookup` predicts from the add-1 smoothed band counts of the full visible tuple when that tuple occurs at least $m = 5$ times in training, and otherwise falls back to `nb`.

Exact references. The *oracle*, a reference rather than an attacker, is the exact Bayes posterior under the true generator, $P(b \mid v) \propto \pi(b)P(v \mid b)$, from enumerating every raw household; scored on the test households it is the "sample oracle". The enumeration also gives the population Bayes accuracy $\sum_v \max_b \pi(b)P(v \mid b)$, $I(\text{band}; \text{view})$ and $H(\text{band} \mid \text{view})$ in bits, the mass of view values that exclude a band, and `nb_limit`, the exact accuracy of `nb` with unlimited training data (likewise for `nb_pair`). The fields are conditionally independent given the band apart from the size-resolution pair (and a band-independent duplication of "no event"), so `nb_limit` equals the Bayes ceiling exactly in A, B and D, and `nb_pair`'s limit equals it in C, where its one joined pair is the generator's one interaction (Table 5). That learned attackers approach the ceilings is expected by design and says nothing about attackers in general.

Renderers and utility. The *strict* renderer reads only the supplied view and abstains unless the view entails the answer; for E it reads the answers directly. The *default-filling* renderer emits a default (`no`, `none`, or `email` for the channel) wherever the strict renderer abstains. An *action item* is a (household, item) pair whose correct answer is `yes` or not `none`; the contact channel always is one. Each output is *supported-correct* (entailed by the view), *unsupported-correct* (a lucky default), *unsupported-wrong*, or an *omission*, and only supported outputs earn action-item credit. This split was added after the review; the original protocol counted lucky defaults as correct.

Measures. On the test split, per scenario, view and attacker: accuracy (correct out of 3,000), macro recall, and mean log-loss in bits, $-\log_2 p(\text{true band})$ with probabilities clipped below at 10^{-12}. Intervals are 95% percentile bootstraps with 2,000 resamples of test households, resampled jointly across views for paired differences. They cover test-sample uncertainty only: each attacker is fitted once per dataset, and the seed runs show training and generator variability. There is no correction for multiple comparisons; only H1a, H1b and H2a are confirmatory.

6 Pre-specification and validation

Hypotheses. The confirmatory hypotheses use `nb` on the test split of `main`. **H1a:** accuracy C − A > 0 and **H1b:** C − B > 0, each with a 95% interval excluding 0. **H2a:** D − C < 0, with a 95% interval excluding 0. **H2b**, strict-renderer action items conveyed D $> \max$(A, B) and D $<$ C, was listed as confirmatory in v1.0 and relabelled descriptive in v1.1, because the generator and the renderer fix its outcome. Controls: in `independence` every attacker must be within 0.05 of baseline accuracy; in the positive control `nb` must reach at least 0.99 in A–D with 3,000 directly exposed test rows in each; elsewhere there must be no direct exposure.

Sample size, and an error in the power analysis. Before the freeze, we enumerated the generator and found a C − D gap of only about 3 accuracy points at the Bayes ceiling; with 600 test households the paired 95% interval would also have had a half-width of about 3 points, so the households per scenario were raised from 3,000 to 10,000 before any data were scored. That analysis contained an error of ours: it sized H2a from the Bayes-optimal gap, but H2a is tested with `nb`, which cannot use the size-by-resolution interaction that the coarsening removes (Section 7.2).

It also printed D's ceiling as 0.576, a double rounding of the exact 0.575466; the protocol now reads 0.5755, and no output used the printed value.

Freeze history. The freeze records SHA-256 hashes of the protocol, fixtures and every source file (from v1.1 also the toolkit and example specifications), and the full run refuses to score the test split unless they match. It is self-attested; the study was not pre-registered. All commits date from 13 November 2026. The first (`b542ccc`) adds the protocol with all its pre-freeze revisions, the code and the development results at once; the v1.0 freeze (`7c90ed1`) follows 5 seconds later and the v1.0 test results (`edb8928`) 3 minutes 20 seconds after the first commit; later freezes and results (`5313da6`, `ab3b9bf`, `da38969`, `986af66`) follow the review and the toolkit fix. Whether the protocol preceded the development runs, and whether the sample-size change preceded any data, therefore rests on attestation, not on the commit history. One point is in the study's favour: at freeze time the development results were visible and unflattering, with H2a not supported (-0.0060 [-0.0280, 0.0160]) and both sequences behind S7 not monotone in λ, and the hypotheses and the S7 rule were frozen unchanged. Development and test splits come from the same datasets and are scored by the same trained models, so any choice made after a development run was made with knowledge of the test outcome up to sampling noise; the operational rules for S5 and S7 fall in this class.

Post-review revisions. An independent senior review of the frozen v1.0 study found no bug that changes a reported number and three reporting problems that had to be fixed: H2a's fragility and power analysis, the credit given to lucky defaults, and the wording of S7. Revision v1.1 relabelled H2b, split utility three ways, reworded S1 and S7, and added the exploratory analyses and the toolkit, all labelled as post-review. A generated delta file compares every v1.0 output with its v1.1 counterpart cell by cell: 0 frozen numbers changed. Revision v1.1.1 fixed a post-freeze bug in the toolkit's Monte Carlo reporting; exact mode and the pilot are unchanged. The test split has been scored more than once; the pipeline is deterministic, so rescoring cannot change a result, but "scored once" is attested rather than enforced.

Validation. Every run executes 164 checks and exits successfully only if all pass, as they did on the test split: hand-checked households, split isolation, byte-identical regeneration of every dataset, removal of the band from attacker inputs, exact denominators, the controls, the generator against its specification, the oracle and the hard zero, attacker isolation, identical attacker outputs across two processes, hand-computed attacker posteriors, a strict renderer that is never wrong over every raw record, and 51 unit, toolkit and mutation tests. The mutation tests must catch a band leaked into release 2, a split shuffled per view, a guessing strict renderer, two broken naive Bayes variants and the reviewer's bypass of the attacker source check. That check audits only the shipped attacker and is not a sandbox, and the exposure scan matches names and values exactly against the band, so "no direct exposure" means that no field is named or valued as the band, not that nothing encodes it. The reproducibility check ran the pipeline twice: 77 output files were byte-identical and matched the committed manifest (Python 3.13.7). Two re-derivations of the oracle sharing no code with it (by the study lead and by the reviewer) agree with it.

7 Results

7.1 Leakage per view

Tables 4 and 5 give the main results. Every attacker that sees more than the prior beats the 0.3890 baseline in every view, and the estimates lie within sampling noise of the exact values: `nb` reaches

Table 4: Attackers on the 3,000 test households of `main` (frozen v1.0 outputs). Brackets are 95% bootstrap intervals (test-sample uncertainty only). Log-loss is the mean of $-\log_2 p$(true band) in bits; lower is better. `nb_pair` equals `nb` in A, B and E and is not repeated there. The oracle is the exact Bayes posterior, a reference rather than an attacker.

View	Attacker	Accuracy	Macro recall	Log-loss (bits)
any	baseline	0.3890 [0.3727, 0.4060]	0.3333 [0.3333, 0.3333]	1.5626 [1.5522, 1.5725]
A	`nb`	0.5193 [0.5017, 0.5363]	0.5062 [0.4883, 0.5228]	1.3716 [1.3476, 1.3949]
A	`lookup`	0.5063 [0.4880, 0.5237]	0.4914 [0.4730, 0.5077]	1.4145 [1.3913, 1.4387]
A	oracle	0.5210 [0.5020, 0.5383]	0.5060 [0.4887, 0.5214]	1.3663 [1.3418, 1.3903]
B	`nb`	0.5327 [0.5150, 0.5503]	0.5391 [0.5222, 0.5557]	1.3260 [1.3027, 1.3492]
B	`lookup`	0.5280 [0.5100, 0.5457]	0.5256 [0.5080, 0.5420]	1.3539 [1.3317, 1.3766]
B	oracle	0.5347 [0.5163, 0.5517]	0.5415 [0.5246, 0.5585]	1.3225 [1.2996, 1.3456]
C	`nb`	0.6003 [0.5823, 0.6177]	0.6043 [0.5866, 0.6215]	1.1893 [1.1622, 1.2169]
C	`nb_pair`	0.6087 [0.5913, 0.6260]	0.6122 [0.5947, 0.6291]	1.1589 [1.1319, 1.1867]
C	`lookup`	0.5987 [0.5810, 0.6157]	0.6010 [0.5838, 0.6177]	1.2085 [1.1802, 1.2363]
C	oracle	0.6093 [0.5917, 0.6267]	0.6118 [0.5946, 0.6288]	1.1503 [1.1232, 1.1780]
D	`nb`	0.5820 [0.5637, 0.5983]	0.5836 [0.5650, 0.6004]	1.2357 [1.2105, 1.2630]
D	`nb_pair`	0.5807 [0.5630, 0.5977]	0.5818 [0.5643, 0.5993]	1.2346 [1.2096, 1.2617]
D	`lookup`	0.5677 [0.5500, 0.5847]	0.5663 [0.5490, 0.5840]	1.2860 [1.2602, 1.3122]
D	oracle	0.5763 [0.5583, 0.5930]	0.5789 [0.5616, 0.5954]	1.2293 [1.2033, 1.2566]
E	`nb`	0.5663 [0.5480, 0.5840]	0.5735 [0.5557, 0.5905]	1.2933 [1.2571, 1.3273]
E	`lookup`	0.5623 [0.5440, 0.5790]	0.5572 [0.5386, 0.5738]	1.3077 [1.2791, 1.3372]
E	oracle	0.5753 [0.5580, 0.5923]	0.5685 [0.5514, 0.5850]	1.2497 [1.2240, 1.2760]

Table 5: Exact population values for `main`, from enumeration of the generator. MI and H are in bits; Excl. is the probability mass of view values that exclude a band. The last two columns give the strict renderer's share of action items conveyed: exact expectation, and observed on the test split (count of 10,998 action items).

View	Bayes	`nb_limit`	`nb_pair_limit`	MI	H(band | view)	Excl.	Exact share	Observed
baseline	0.4000	0.4000	0.4000	0.0000	1.5589	0		
A	0.5172	0.5172	0.5172	0.1878	1.3711	0	0.5205	0.5187 (5,705)
B	0.5271	0.5271	0.5271	0.2179	1.3410	0	0.4582	0.4575 (5,032)
C	0.6042	0.5953	0.6042	0.3889	1.1699	0.0575	1	1 (10,998)
D	0.5755	0.5755	0.5755	0.3148	1.2441	0	0.8054	0.8022 (8,823)
E	0.5650	0.5582	0.5582	0.2896	1.2693	0	1	1 (10,998)

0.6003 in C against its unlimited-data value of 0.5953 and the Bayes ceiling of 0.6042. Half a point of accuracy is noise here: in D, `nb` (0.5820) beats the sample oracle (0.5763) on accuracy, though not on log-loss (1.2357 against 1.2293). `lookup` is below `nb` in every view and is not a flexible or upper-bound attacker; no learned attacker in the suite can find an interaction it was not given. Train and test households overlap heavily in feature space by design: for a share of 0.9017 of test households the full view-A tuple occurs at least 5 times in training, against 0.2253 in view C.

7.2 Confirmatory results

All three confirmatory tests are supported on the pre-specified seed (Table 6). Linking the two releases raised the naive-Bayes attacker's accuracy by 8.1 points over release 1 and 6.8 points over release 2, against exact ceiling gaps of 8.7 and 7.7 points, although neither release contains the band. The direction is implied by the construction; the size is the result. H2b holds, descriptively: the strict renderer conveyed 5,705 (A), 5,032 (B), 10,998 (C) and 8,823 (D) of 10,998 action items,

Table 6: Confirmatory tests (`nb`, `main`, test split, frozen v1.0) beside the exact population gaps of the Bayes oracle and of `nb` with unlimited data. Discordant: households correct in the first view only / the second view only.

Test	Estimate [95% CI]	Discordant	Bayes gap	`nb_limit` gap	Outcome
H1a: C − A	+0.0810 [0.0623, 0.0993]	549 / 306	+0.0870	+0.0782	supported
H1b: C − B	+0.0677 [0.0497, 0.0850]	470 / 267	+0.0771	+0.0683	supported
H2a: D − C	−0.0183 [−0.0313, −0.0057]	181 / 236	−0.0287	−0.0199	supported (frozen seed)

against exact expected shares of 0.5205, 0.4582, 1 and 0.8054.

Table 7: D − C on all six seed runs of `main` (run 0 is the frozen primary run; runs 1–5 are post-review exploratory analyses of the pre-specified seed replicates). Accuracy differences, with a paired 95% bootstrap interval for `nb`; `nb`'s log-loss difference is in bits. The intervals for that log-loss, for `nb_pair` and for the sample oracle exclude 0 in every run; their bounds closest to 0 are +0.0293, −0.0033 and −0.0040.

Run	`nb` accuracy [95% CI]	Criterion met	`nb` log-loss	`nb_pair` accuracy	Sample oracle accuracy
0	−0.0183 [−0.0313, −0.0057]	yes	+0.0465	−0.0280	−0.0330
1	−0.0130 [−0.0260, 0.0003]	no	+0.0436	−0.0173	−0.0177
2	−0.0047 [−0.0190, 0.0087]	no	+0.0421	−0.0283	−0.0273
3	−0.0160 [−0.0297, −0.0030]	yes	+0.0432	−0.0253	−0.0250
4	−0.0133 [−0.0273, 0.0010]	no	+0.0529	−0.0253	−0.0227
5	−0.0200 [−0.0330, −0.0073]	yes	+0.0480	−0.0273	−0.0237

H2a across seeds. H2a is the confirmatory result for the frozen seed, and it is not robust: the criterion is met in 3 of 6 seed runs (Table 7). The across-seed mean of `nb`'s D − C accuracy difference is −0.0142 (SD 0.0054; t interval on 5 degrees of freedom [−0.0199, −0.0085]). This mean is post hoc: it combines test-sample, training and generator variability, which the bootstrap does not, and it is not a confirmatory test. The exact reference gaps are −0.0287 for the Bayes oracle and −0.0199 for `nb` with unlimited training data; the design-time power analysis used the first when it should have used the second. With a normal approximation and a paired standard error of 0.0068 (the median over the six runs), power was about 0.56 at the across-seed mean and 0.84 at `nb`'s unlimited-data gap, against 0.99 had the attacker been Bayes-optimal. The pre-specified seed happened to land on the passing side. Secondary views are stable: `nb`'s log-loss is worse in D than in C in 6 of 6 runs, by 0.0421 to 0.0529 bits (pre-specified as secondary under S9, not a replacement test), and `nb_pair` and the sample oracle, which can use the interaction the coarsening removes, show D below C with intervals excluding 0 in 6 of 6 runs. The fragility is specific to an attacker that cannot see that interaction. We do not replace the primary seed, pool the seeds into a new test, or call H2a robust or replicated.

7.3 Secondary results

S1 is not confirmed on accuracy, although it was on the development split (Table 8). With Tables 5 and 6, S4 completes the coarsening's account: the declared coarsening removes 2.9 points of accuracy at the ceiling and 19.5% of the action items, and the coarsened linked releases still leak more than either fine release alone.

7.4 Logical exclusions

The planted zero produced exclusions in 193 of 3,000 test households of view C (0.0643, against an exact mass of 0.0575) and in no household of any other view; in the five replicate runs the counts

Table 8: Secondary predictions (`main`, test split, `nb` unless stated). "By construction" marks outcomes fixed by the generator or the definitions.

Code	Prediction	Result	Reading
S1	`nb_pair` − `nb` on C > 0	+0.0083 [−0.0007, 0.0177]; log-loss −0.0303 [−0.0400, −0.0212] bits	Not confirmed on accuracy; log-loss is secondary (S9), not a replacement.
S2	E − baseline > 0	+0.1773 [0.1560, 0.1993]	As predicted: checklist answers alone leak.
S3	exclusions > 0 in C only	C 193 households; A, B, D, E 0	By construction (one planted zero).
S4	D − A, D − B, no direction	+0.0627 [0.0417, 0.0837]; +0.0493 [0.0317, 0.0673]	Still above either fine release, consistent with Ganta et al.
S5	`lookup` coverage in C low	0.2253	As predicted (frozen rule: below 0.5).
S6	default-fill unsupported > 0 in A, B, D; 0 in C, E	wrong defaults A 5,293, B 4,147, D 2,175; all unsupported 13,900, 11,349, 4,445; C, E 0	By construction; v1.0 omitted lucky defaults.
S7	gains shrink as λ falls	C − A at $\lambda = 0.25, 0.50, 0.75, 1$: −0.0010, +0.0383, +0.0550, +0.0810	Frozen rule met; weak evidence (Section 7.7).
S8	prevalence: accuracy and macro recall	Section 7.7	Reported.
S9	information measures	MI: A 0.1878, B 0.2179, C 0.3889, D 0.3148, E 0.2896 bits	Reported.

were 175, 172, 178, 162 and 165. Every exclusion rules out `low`, and none determines the band. The first test household with an exclusion (worked case 3, true band `high`) shows `vehicle_major` in release 1 and `settled_in_full` in release 2; its posteriors (low/medium/high) are:

Posterior	A	B	C	D	E
oracle	0.098/0.323/0.579	0.147/0.330/0.522	0.000/0.106/0.894	0.065/0.314/0.621	0.097/0.319/0.585
`nb`	0.107/0.354/0.539	0.171/0.316/0.514	0.030/0.214/0.756	0.083/0.336/0.581	0.059/0.260/0.681
`nb_pair`			0.001/0.083/0.917	0.084/0.324/0.592	

Neither release alone rules anything out, since a low-band household can report a major event and can settle a minor one. Linked, the pair excludes `low` exactly. `nb`, which multiplies one factor per field, cannot represent the exclusion and leaves 0.030 on `low`; `nb_pair`, which sees the pair jointly, leaves 0.001. In D the size is gone, and so is the exclusion. This is a composition effect planted by construction, but not the intersection attack of Ganta et al. [1]: each release alone leaves all three bands possible, and the exclusion comes only from the joined pair, as in the join inferences of Wang and Fung [3].

7.5 Utility

The default-filling renderer answers without evidence on 51.0% of items in A, 42.0% in B and 16.4% in D, and much of that is right only by luck (Table 9). Crediting lucky defaults had inflated B's action items conveyed from 0.4575 to 0.6229, purely by guessing the most common contact channel. Two facts qualify D's 80.5% utility. Three action-item classes are pass-throughs of one field each (contact channel, goal, documents); they make up 2.0075 of 3.6793 action items per household, or 54.6%, and coarsening never touches them. On the four items that the coarsening does affect (overdraft guidance, referral, hardship review, plan monitoring), D conveys 0.0871 of 0.8030 action items per household, or 10.8%.

Table 9: Utility split three ways (post-review correction, v1.1). Shares of all item outputs are exact expectations for `main`. The last column gives the share of action items conveyed on the test split, under the v1.0 credit rule and with credit for supported outputs only. C and E are complete under both renderers.

View	Renderer	Supported correct	Unsupported correct	Unsupported wrong	Omission	Action items conveyed v1.0 → v1.1
A	strict	0.4900	0	0	0.5100	0.5187 → 0.5187
A	default	0.4900	0.3140	0.1960	0	0.5187 → 0.5187
B	strict	0.5803	0	0	0.4197	0.4575 → 0.4575
B	default	0.5803	0.2649	0.1548	0	0.6229 → 0.4575
D	strict	0.8359	0	0	0.1641	0.8022 → 0.8022
D	default	0.8359	0.0845	0.0795	0	0.8022 → 0.8022
C, E	either	1	0	0	0	1 → 1

7.6 The trade-off

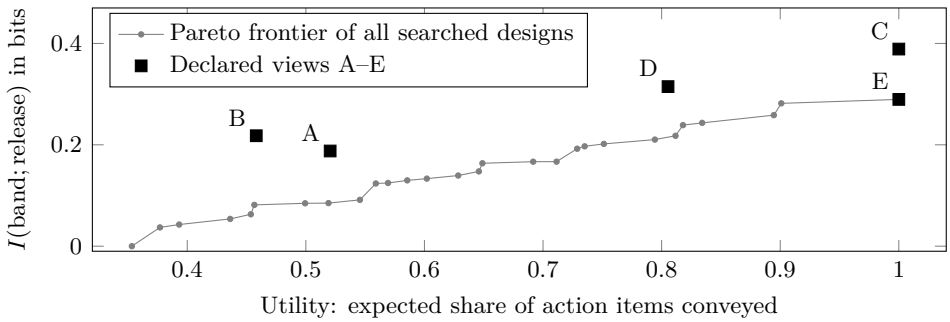


Figure 1: Leakage against utility, both exact for `main`. Grey: the designs that no other design beats on both axes, from the toolkit's exhaustive search over keeping, coarsening or dropping each field, plus task-output designs (Section 8). Squares: the five declared views. D lies above and to the left of the frontier, so it is dominated; E (the checklist answers) is the frontier's full-utility end.

Figure 1 places the five views against the toolkit's exact frontier. Linking (C) gives full utility at the highest leakage. The declared coarsening D is dominated: the checklist answers E leak less (0.2896 against 0.3148 bits) and convey everything, and a field-level design that coarsens the share and the event, keeps the resolution, and drops the income pattern and the second payment flag reaches 0.8946 utility at 0.2585 bits. A and B leak less than any linked view but convey about half of the action items.

7.7 Sensitivity to dependence and prevalence

The exact gains shrink as λ falls, by construction (Table 10). The frozen S7 rule (`nb`'s point estimates of C − A and C − B non-decreasing over $\lambda = 0.25$ to 1) was met, but this is weak evidence: adjacent steps are within about two standard errors, the rule failed on the development split, and at $\lambda = 0.25$ `nb` realises no detectable gain (-0.0010 [-0.0153, 0.0127]) although the population gain is 0.0128. Effects of about one point are below this design's resolution.

Under skewed prevalence, H2a's direction fails on accuracy for `nb`: D − C is $+0.0010$ [-0.0060, 0.0080], while log-loss still shows the loss ($+0.0237$ [0.0144, 0.0331] bits) and the exact gap is -0.0174. With a 0.7000 baseline, argmax accuracy is a poor leakage measure. For S8, in `prev_skewed nb`'s accuracy in C, D and E is 0.7380, 0.7390 and 0.7247 but its macro recall is 0.4806, 0.4713 and 0.5054; in `prev_balanced` the two agree (0.5743, 0.5503, 0.5460 against 0.5755, 0.5513, 0.5474).

Table 10: Sensitivity scenarios. Exact columns are population values; the `nb` columns are test-split estimates with 95% bootstrap intervals, and the log-loss difference is in bits. The baseline accuracy is 0.4000, except 0.3333 in `prev_balanced` and 0.7000 in `prev_skewed`.

	Exact		`nb`, test split		
Scenario	C − A	D − C	C − A	D − C	D − C log-loss
`independence`	0	0	−0.0007 [−0.0080, 0.0077]	0.0000 [−0.0090, 0.0090]	−0.0033
`dep_025`	+0.0128	−0.0048	−0.0010 [−0.0153, 0.0127]	−0.0013 [−0.0147, 0.0127]	+0.0015
`dep_050`	+0.0358	−0.0137	+0.0383 [0.0190, 0.0560]	−0.0227 [−0.0367, −0.0087]	+0.0125
`dep_075`	+0.0605	−0.0214	+0.0550 [0.0367, 0.0723]	−0.0223 [−0.0363, −0.0083]	+0.0339
`main`	+0.0870	−0.0287	+0.0810 [0.0623, 0.0993]	−0.0183 [−0.0313, −0.0057]	+0.0465
`prev_balanced`	+0.0946	−0.0292	+0.0773 [0.0587, 0.0970]	−0.0240 [−0.0383, −0.0097]	+0.0456
`prev_skewed`	+0.0354	−0.0174	+0.0200 [0.0107, 0.0297]	+0.0010 [−0.0060, 0.0080]	+0.0237

7.8 Controls

Table 11: `nb` accuracy on `main` by number of training labels (nested subsets of the training rows; exploratory, post-review). The 6,000-label row is the primary run; the last two rows are exact.

Labels	A	B	C	D	E	C − A	D − C
100	0.4440	0.5070	0.5050	0.5067	0.5153	+0.0610	+0.0017
300	0.4667	0.5200	0.5397	0.5353	0.5443	+0.0730	−0.0043
1,000	0.5140	0.5197	0.5783	0.5563	0.5563	+0.0643	−0.0220
3,000	0.5187	0.5323	0.5963	0.5833	0.5633	+0.0777	−0.0130
6,000	0.5193	0.5327	0.6003	0.5820	0.5663	+0.0810	−0.0183
`nb_limit`	0.5172	0.5271	0.5953	0.5755	0.5582	+0.0782	−0.0199
Bayes ceiling	0.5172	0.5271	0.6042	0.5755	0.5650	+0.0870	−0.0287

Pre-specified controls. All passed. In `independence` the largest gap from the baseline was 0.0253 (`lookup` in A), within the 0.05 tolerance, which is wider than the effects of interest. In the positive control `nb` reached 1.0000 in A–D with 3,000 exposed test rows in each, and no other scenario had a direct exposure.

Exploratory controls (post-review). With training labels permuted, no attacker beat the baseline in any view (the largest accuracy above the 0.3890 baseline was 0.0000). Attackers seeing only identifier characters and row position reached 0.3863 [0.3687, 0.4037]. In the learning curve (Table 11), linking already helps an attacker with 100 labels (C − A +0.0610) while the coarsening's effect is invisible there (D − C +0.0017), and with few labels the checklist answers are about as useful to `nb` as the linked release (0.5443 against 0.5397 at 300 labels).

7.9 Task outputs against coarsening (exploratory)

The comparison of E with D was not pre-specified; it was raised after the test results were seen. Three statements are defensible.

- **E always dominates the linked fine release,** by construction: same utility, and less leakage here (0.5650 against 0.6042).
- **E still leaks:** its ceiling is 0.1650 above the baseline, more than either single release (0.5172 for A, 0.5271 for B).

Table 12: Exact values for variants of D and E in `main` (exploratory, post-review).

View	Bayes accuracy	MI (bits)	`nb_limit`
C (linked fine releases)	0.6042	0.3889	0.5953
D	0.5755	0.3148	0.5755
D without `income_pattern`	0.5667	0.2936	0.5667
E	0.5650	0.2896	0.5582
E plus one item that needs `income_pattern`	0.5762	0.3110	0.5600
E plus one item "major expense settled in full"	0.5793	0.3252	0.5666
E plus both	0.5871	0.3456	0.5783
E without `goal_module`	0.5380	0.2218	0.5305

- **Against a coarsening, E is not ordered in general.** E's ceiling is below D's by 0.0105 (MI 0.2896 against 0.3148 bits), but one extra checklist item reverses this (Table 12): an item needing the income pattern lifts E's ceiling above D's (0.5762 against 0.5755), though not its mutual information, and an item flagging a major expense settled in full lifts both (0.5793 and 0.3252 bits). D keeps `income_pattern`, which the checklist never uses, while coarsening the fields the checklist needs: it is a task-unaware coarsening and an easy opponent.

The test split cannot resolve the E − D gap: the sample oracle's E − D accuracy is −0.0010 [−0.0167, 0.0143] on the primary run, and its interval excludes 0 in 1 of 6 seed runs (`nb`'s in 2 of 6). `nb` overstates the gap, because E's items share fields and naive Bayes is misspecified on E (unlimited-data accuracy 0.5582 against an exact 0.5650). Under skewed prevalence the exact gap shrinks to −0.0024. With the same schema, checklist and coarsening, E had lower Bayes accuracy than D in 88.5% to 92.5% of 400 random generators per setting (four settings, from mild perturbation to fully random tables) and lower mutual information in 90.0% to 96.8%: fairly robust to the probabilities, not to the checklist or the coarsening.

8 A reusable release audit

The pilot hard-codes one schema, generator, set of views and checklist, so we built `release_audit`, a standard-library toolkit (`python3 -m release_audit`) that asks the same questions of a user's own pipeline. It is not part of the pilot's inference.

Specification and evaluation. A user writes one JSON file: the hidden target and its prior; fields assigned to stages, with exact conditional probability tables (a zero is a hard constraint); declared coarsenings; the task as checklist items given by ordered rules over raw fields; and the views to compare, including task outputs. When the raw space has at most 500,000 positive-probability records (a configurable limit), Bayes accuracy, mutual information, `nb_limit`, exclusion masses and utility are exact. Above it the toolkit samples (by default 200,000 records), reports Bayes accuracy as a bracket between a held-out plug-in classifier (usually pessimistic) and resubstitution (optimistic), headlines the Miller–Madow corrected mutual information, and reports no exclusions, since a sampled zero is not a hard constraint (the v1.1.1 correction). Utility is by entailment: an item is conveyed if every positive-probability raw record yielding the released value gives it the same value. On the household specification this equals the pilot's strict renderer on all 328,320 record-view pairs, and the oracle's Bayes accuracies, mutual information and `nb_limit` for A–E are reproduced.

Information budget and design search. For each field the toolkit reports its information alone, the cost of dropping it from the full linked view and the cost of each declared coarsening

Table 13: The toolkit's per-field information budget for the household specification (`main`, exact, bits except the Bayes column). The reference is the full linked view: 0.3889 bits and Bayes accuracy 0.6042.

Field	Stage	MI alone	MI lost if dropped	Bayes lost if dropped	MI lost by coarsening
`r2.goal`	2	0.0914	0.0587	0.0265	
`r1.essential_share`	1	0.0865	0.0554	0.0254	0.0205
`r1.payment_flag`	1	0.0850	0.0544	0.0229	0.0025
`r2.payment_flag`	2	0.0846	0.0545	0.0224	0.0041
`r2.resolution`	2	0.0629	0.0697	0.0270	0.0412
`r1.income_pattern`	1	0.0318	0.0192	0.0083	
`r1.event_code`	1	0.0000	0.0281	0.0088	0.0281
`r2.document_status`	2	0.0000	0.0000	0.0000	
`r1.contact_channel`	1	0.0000	0.0000	0.0000	

(Table 13), and it scans cross-stage pairs for synergy, $I(\text{target}; a, b) - I(\text{target}; a) - I(\text{target}; b)$, and for exclusions that appear only when stages are linked. On the household specification it finds the size-by-resolution interaction without being told where to look: `r1.event_code` carries 0.0000 bits alone but costs 0.0281 bits when dropped, and the pair (`r1.event_code`, `r2.resolution`) carries 0.1029 bits, a synergy of 0.0400, the largest of any pair, with the exclusion of `low` appearing only when the two are linked. The design search enumerates releases (keep, coarsen or drop each field, plus task outputs alone and with one field) and reports the frontier: for the household specification, 3,903 designs with 28 on the frontier (Figure 1); D is dominated.

Real-data mode. From labelled records the toolkit estimates the same tables with cross-validated naive Bayes, a pairwise logistic regression that finds interactions unaided, learning curves and bias-corrected mutual information, without exclusions. On 400 records sampled from the grocery specification, plug-in resubstitution accuracy for the linked view is 0.9100 against an exact 0.8002.

A second example. The grocery example is a different fictional pipeline: order, picking and delivery stages, a binary hidden attribute (young children in the household, prior 0.3000) and one invented cross-stage hard constraint (households without young children never ask for "no bell" on an afternoon slot). Linking the stages raises the exact ceiling from a 0.7000 baseline to 0.8002. Unaided, the budget finds the synergy between delivery window and doorstep instruction (0.0378 bits) and the exclusion it creates only when order and delivery are linked (mass 0.0330). The household ordering of E and D does not carry over: here the task outputs leak more than the declared coarsening (Bayes accuracy 0.7692 against 0.7190; 0.1843 against 0.1225 bits) while conveying every action item against 0.4667, and they keep the exclusion, because two items pass the window and the doorstep instruction through.

Release-design checklist. The documentation condenses the pilot into seven lessons: a stable key across stages is the linkage; a verbatim pass-through leaks at full strength (dropping `goal_module` from E lowers its information from 0.2896 to 0.2218 bits); cross-stage hard constraints give exclusions; default-filling renderers make unsupported claims; task outputs still leak; coarsen fields the task does not need first; check designs against the exact ceiling. A first specification of 8 to 12 fields is estimated at 3 to 6 hours; the toolkit has not been used outside this study.

9 Limitations

Synthetic, no language model, linkage assumed. Every probability is invented and no generator was validated against real households, so no result is a rate or effect size for real households or pipelines. No language model produces the findings, attacks them or renders the report, there is no free text, and nothing here says how language-model attackers or language-model-generated findings behave. Linkage is assumed, not tested. **By construction.** The directions of H1 and H2a, H2b, S3, S6, "E keeps full utility" and "E leaks no more than C" follow from the design, and the attackers match the generator's structure, so their closeness to the ceilings says nothing about attackers in general. **One coarsening, one checklist.** D is a single hand-picked, task-unaware coarsening, and the E-versus-D ordering reverses with one checklist item. Nothing here says that coarsening works or fails in general, or that minimal structured findings or task outputs are safe or unsafe as a class. Coarsening is not differential privacy [12], which provably resists composition attacks [1]. **Fragility and resolution.** H2a meets its criterion in 3 of 6 seed runs, and the power analysis used the wrong reference gap. Intervals cover test-sample uncertainty only, the independence tolerance (0.05) is wider than H2a or the E − D gap, and a true gain of about one point (`dep_025`) went undetected. **Isolation, exposure, pre-specification.** Attacker isolation is process-level, not a sandbox, and the exposure scan finds fields named or valued as the band, not encodings of it. The freeze is self-attested, the first three commits fall within minutes, and the whole v1.0 study, from protocol to test results, was carried out in one evening, which bounds how much deliberation "pre-specified" implies; development results previewed the test outcome up to noise. **Toolkit and literature.** Toolkit results are exact for the specification, not the world; synergy is scanned pairwise; the design search turns greedy beyond 5,000 field-option combinations; utility assumes the task is exactly the declared checklist. The prior-work search was not exhaustive.

10 Conclusion

In a small generator where every ceiling is computed exactly, linking two releases of structured findings raised a naive-Bayes attacker's accuracy by 6.8 to 8.1 points over a single release, and one cross-release pair ruled out a band outright, although neither release contained the band. A declared coarsening removed 2.9 points at the ceiling and 19.5% of the action items and still leaked more than either single release; its confirmatory test passed on the pre-specified seed but in only 3 of 6 seed runs. The task's outputs kept full utility and still leaked; against a coarsening their ordering depended on the checklist. These illustrate established results, with sizes attached.

Code and data availability

The git repository holds the protocol with its revision log, `FREEZE.json`, code, fixtures, summary results, the frozen v1.0 outputs, the toolkit with documentation, and this paper with `paper/NUMBERS.md` and `paper/check_numbers.py`, which re-derive every number in the paper from the project's files. It is public at `https://github.com/Anishpras/linked-findings-inference`. `python3 run_pilot.py` regenerates every pilot output (the README gives about 95 s on a 12-core laptop), including the per-case files and datasets, which are too large for git and are regenerated byte for byte against SHA-256 hashes in `results/output_manifest.json`; `-check-reproducible` runs the pipeline twice and compares every byte. The code uses only the Python standard library; all data are synthetic.

Acknowledgements and AI-assistance statement

The code, analysis and manuscript were prepared with substantial assistance from Claude (Anthropic), an AI assistant, working under the author's direction. Separate AI agents carried out the literature search, the implementation and the independent senior review. The author is responsible for the content. The work used no paid API, cloud compute or external funding.

References

[1] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith. Composition attacks and auxiliary information in data privacy. In *Proc. ACM SIGKDD (KDD)*, pp. 265–273, 2008. arXiv:0803.0032. https://doi.org/10.1145/1401890.1401926

[2] S. Taylor, P. K. Vippathalla, and J. Coon. Adaptive privacy of sequential data releases under collusion. arXiv:2601.21859 [cs.IT], Jan. 2026 (v2 Jul. 2026). https://arxiv.org/abs/2601.21859

[3] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *Proc. ACM SIGKDD (KDD)*, pp. 414–423, 2006. https://doi.org/10.1145/1150402.1150449

[4] X. Xiao and Y. Tao. M-invariance: Towards privacy preserving re-publication of dynamic datasets. In *Proc. ACM SIGMOD*, pp. 689–700, 2007. https://doi.org/10.1145/1247480.1247556

[5] J. Brickell and V. Shmatikov. The cost of privacy: Destruction of data-mining utility in anonymized data publishing. In *Proc. ACM SIGKDD (KDD)*, pp. 70–78, 2008. https://doi.org/10.1145/1401890.1401904

[6] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. Knowledge and Data Engineering*, 13(6):1010–1027, 2001. https://doi.org/10.1109/69.971193

[7] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):571–588, 2002. https://doi.org/10.1142/S021848850200165X

[8] D. Kifer. Attacks on privacy and deFinetti's theorem. In *Proc. ACM SIGMOD*, pp. 127–138, 2009. https://doi.org/10.1145/1559845.1559861

[9] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *Proc. ACM SIGMOD*, pp. 575–586, 2004. https://doi.org/10.1145/1007568.1007633

[10] Y. Kawamoto, K. Chatzikokolakis, and C. Palamidessi. On the compositionality of quantitative information flow. *Logical Methods in Computer Science*, 13(3), 2017. arXiv:1611.00455. https://doi.org/10.23638/LMCS-13(3:11)2017

[11] B. Jayaraman and D. Evans. Are attribute inference attacks just imputation? In *Proc. ACM CCS*, pp. 1569–1582, 2022. https://doi.org/10.1145/3548606.3560663

[12] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography (TCC)*, LNCS, pp. 265–284, 2006. https://doi.org/10.1007/11681878_14

[13] R. Staab, M. Vero, M. Balunović, and M. Vechev. Beyond memorization: Violating privacy via inference with large language models. In *ICLR*, 2024. https://arxiv.org/abs/2310.07298

[14] H. Yukhymenko, R. Staab, M. Vero, and M. Vechev. A synthetic dataset for personal attribute inference. In *NeurIPS Datasets and Benchmarks Track*, 2024. https://arxiv.org/abs/2406.07217

[15] V. Patil, E. Stengel-Eskin, and M. Bansal. The sum leaks more than its parts: Compositional privacy risks and mitigations in multi-agent collaboration. arXiv:2509.14284, 2025. https://arxiv.org/abs/2509.14284

[16] S. Asif and M. M. Amiri. Information-theoretic privacy control for sequential multi-agent LLM systems. arXiv:2603.05520, 2026. https://arxiv.org/abs/2603.05520

[17] J. Zhou, N. Mireshghallah, and T. Li. Operationalizing data minimization for privacy-preserving LLM prompting. arXiv:2510.03662, 2025. https://arxiv.org/abs/2510.03662

[18] S. Hu, L. Liu, Z. Meng, and Z. Zhao. ToolPrivacyBench: Benchmarking purpose-bound privacy in tool-using LLM agents. arXiv:2606.28061, 2026. https://arxiv.org/abs/2606.28061

[19] A. Zharmagambetov, C. Guo, I. Evtimov, M. Pavlova, R. Salakhutdinov, and K. Chaudhuri. AgentDAM: Privacy leakage evaluation for autonomous web agents. In *NeurIPS Datasets and Benchmarks Track*, 2025. https://arxiv.org/abs/2503.09780

[20] E. Bagdasarian, R. Yi, S. Ghalebikesabi, P. Kairouz, M. Gruteser, S. Oh, B. Balle, and D. Ramage. AirGapAgent: Protecting privacy-conscious conversational agents. In *Proc. ACM CCS*, 2024. https://arxiv.org/abs/2405.05175