HASHMARK: TYPE-AGNOSTIC WATERMARKING FOR TABLES WITH FREE-FORM TEXT

Anonymous authors
Paper under double-blind review

ABSTRACT

Tables are released, copied, and edited far from their owners. A copy found later may have had its rows, values or text changed. We introduce HashMark, a watermark for tables containing both numbers and free-form text. It makes small numerical edits and selects text paraphrases under one secret key. Verification requires only the suspect table, the key, and a public descriptor of how it was marked; neither the original table nor its generator is needed. Column-level statistical tests account for repeated values and dependence between text cells, providing evidence that a suspect table contains watermarked data.

On mixed tables, the numbers and the text each keep the verdict when the other is attacked, and no clean control or wrong key is flagged in any mixed-table test. With 80% of the rows or cells of a table replaced, HashMark still detects every table, while the generation-time watermark MUSE detects 26–34%. A geometric model predicts how many text cells can be marked to within 0.32 percentage points across four corpora. Numerical marking keeps the data as close to the original as an unwatermarked release, to three decimal places, moves downstream accuracy by at most 0.38 points, and marks $10{,}000$ rows in 0.2 s.

1 INTRODUCTION

Proving that a released table came from its owner is a long-standing goal of database watermarking. This is primarily because tables travel far from their source: they are copied, merged, and edited by parties the owner never sees. Modern releases make the problem harder. A single table often mixes measurements, categories, and free-form text, and a later copy may have been sorted, subsampled, rounded, or partially rewritten. Furthermore, the owner may no longer hold the original table, or the generator that produced it: many tables released today are synthetic, sampled from a machine-learning model (a *generator*) trained on real data. However, existing tabular schemes mark only numbers (He et al., 2024; Ngo et al., 2026; Zheng et al., 2024) or need the generator at detection time (Fang et al., 2025; 2026; Zhu et al., 2025), and none marks the text inside a table. This raises a question: *can one watermark both the numbers and the free-form text of a table under a single key, and verify a suspect copy without the original table or the generator?* In this work we answer this question in the affirmative.

1.1 OUR APPROACH AND CONTRIBUTIONS

We introduce HashMark, a watermark that marks both the numbers and the free-form text of a table under one secret key, and verifies a suspect copy using only that copy, the key, and a public description of how the table was marked. Our approach rests on a single idea. A keyed hash function $\mathcal{H}$, which outputs one bit, splits the possible values of each column into two halves, which we call *valid* and *invalid*; "valid" means only that an item satisfies the secret rule, not that it is correct. Without the key, the split is unpredictable, so an unrelated table has about half of its items valid. Before release, we move items into the valid half; a verifier holding the key counts them and finds far more valid items than chance allows.

We treat each column type as follows.

- **Text.** A public *sentence encoder*, a fixed neural network that gives sentences with similar meaning nearby vectors, maps each cell to a vector, its *embedding*; we subtract the column's average em-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

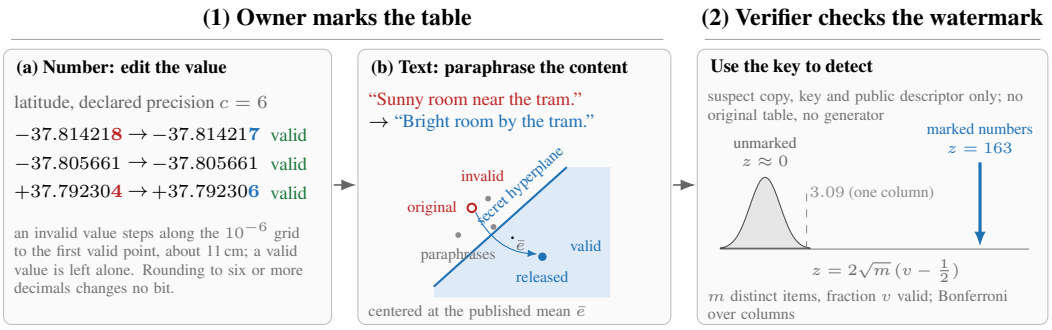


Figure 1: HashMark on one Airbnb listing. (1) Before release, the owner steps (a) an invalid latitude up to its sixth decimal to the first valid value and (b) swaps the summary for a paraphrase that lands on the valid side of a secret hyperplane; values already valid are left alone. (2) Holding only the suspect copy, the key and the public descriptor, a verifier counts valid items per column (m): an unmarked table scores near zero, and the marked Airbnb coordinates score $z = 163$.

bedding $\bar{e}$, which we publish (we call this *centering*), and a secret *hyperplane*, a flat cut through the space like a line dividing a plane, derived from the key, splits the space in two. A cell is valid when it falls on the positive side; to make it valid, we generate a few paraphrases with a public paraphrasing model and publish one that lands there.

- **Numbers.** The owner fixes a precision c, the number of decimals it will publish. A value is valid when the hash of the key, the column name, and the value written to c decimals is zero; otherwise we step it by 10^{-c} until it is, typically one or two steps.
- **Categories.** Values such as marital status are never changed, since replacing `single` with `divorced` falsifies the record rather than nudging it. We treat every category as valid, so these columns carry no evidence and never decide ownership (Appendix B.3).

HashMark thus accepts every column type and marks those where a small change leaves the record true. The marked columns, their precisions, the encoder and the average embedding form a public *descriptor*; only the key is secret. On an Airbnb listing table, for instance, HashMark moves latitude and longitude by about eleven centimeters and replaces some summaries with paraphrases (Figure 1). Sorting, subsampling or reordering the table changes none of these bits.

To verify, let m be the number of distinct items in a marked column, T the number that are valid, and $v = T/m$. We report

$$z \;=\; 2\sqrt{m}\left(v - \tfrac{1}{2}\right),$$

the excess of valid items over the chance level of one half. If the key is unrelated to the table, each item is valid with probability one half, so z is approximately *standard normal*: a bell curve centered at 0 with spread 1, rarely beyond ± 3. We call this chance distribution of z the *null*; a watermarked column scores far above it.

Contributions.

- We watermark every kind of column in a table, numbers, categories and free-form text, under one secret key, and we are the first to mark free-form text inside a table, which prior tabular watermarks leave unmarked or untested (Table 1). Verification needs only the suspect copy, the key, and a public descriptor.
- We analyze the construction: we bound how far marking moves a value, give the exact null distribution for numerical columns and the exact null variance when text cells share one direction, predict from paraphrase geometry how many text cells can be marked, and show how evidence shrinks under deletion.
- Further, we empirically evaluate on eight tabular datasets, three mixed tables and four text corpora against five baselines, under deletion, replacement, rounding, noise and paraphrasing, with ablations of each design choice. Under our calibrated test, the numbers and the text each keep the

Table 1: HashMark versus representative prior tabular watermarks. *Blind* detection needs no original data; generator access is listed separately, matching Table 2. For text, HashMark publishes the column's mean embedding $\bar{e}$ (Section 1.1). *Edit*: changes values of a finished table. *Sampling*: draws several candidates (rows from a generator, or paraphrases) and releases one that carries the mark; HashMark edits numbers and samples paraphrases for text. *Handling of free-form text is unspecified and untested.

	Ngo et al. (2026)	Zheng et al. (2024)	Zhu et al. (2025)	MUSE	HashMark
Data types	Numerical	Any*	Numerical	Any*	**Any**
Blind detection	✓	✗ (needs data)	✓	✓	✓
No generator needed	✓	✓	✗ (inversion)	✗	✓
Embedding	Edit	Edit	Sampling	Sampling	**Edit + Sampling**

verdict when the other is attacked, and no clean control or wrong key is flagged in any mixed-table test; we verify ownership without the original table or the generator.

- We provide a public implementation of our approach, along with the code for all experiments, to facilitate reproducibility and further research in this area. The code is available at `https://anonymous.4open.science/r/hashmark-artifact-3BE8`.

1.2 Related Work

Table 1 summarizes how HashMark differs from the closest prior watermarks.

Classical database watermarks. Agrawal & Kiernan (2002), Sion et al. (2003) and Shehab et al. (2008) embed bits in least-significant digits or subset statistics, assuming a stable primary key and schema to re-locate marked tuples; a synthetic release has neither, and HashMark recomputes its predicate on the released values themselves. Ownership can also be tested through a model trained on the data, by radioactive marking (Sablayrolles et al., 2020) or dataset inference (Maini et al., 2021); HashMark instead verifies the released table itself.

Binning and red–green methods. WGTD (He et al., 2024), Ngo et al. (2026), and TabularMark (Zheng et al., 2024) adapt red–green binning (Kirchenbauer et al., 2023), moving values into keyed "green" intervals. TabularMark is non-blind because it needs the original data; Ngo et al. (2026) is numerical-only. Both place values within valid intervals, whereas HashMark places them on the grid targeted by truncation, giving exact survival at the declared precision (Appendix H.1). We evaluate TabularMark by locating its key rows through a content hash (Appendix J.1). TableMark (Xia et al., 2026) traces recipients with multiple bits; TAB-DRW (Zhao et al., 2026) mixes columns per row and loses the mark under column deletion.

Generative watermarks. TabWak (Zhu et al., 2025) embeds in diffusion latents and inverts the generator for detection. RINTAW (Fang et al., 2025) and MUSE (Fang et al., 2026) select among k generated rows at $O(k)$ cost; our Edit Mode, which edits the numbers of a finished table (Algorithm 1a), needs no generator and costs $O(1)$ per cell. MUSE's bits depend on row structure (seed columns chosen by ranking each row's values, or column position), and its column attack refills columns in place, so dropping or permuting columns, which leaves HashMark's name-keyed bits intact, is untested (Appendix L.2).

Semantic watermarks for text. SemStamp (Hou et al., 2024a;b) selects text by locality-sensitive hashing, on the side of random hyperplanes an embedding falls, and tests the valid fraction against one half. We add centering (Mu & Viswanath, 2018). PMark (Huo et al., 2026a) and concurrent SAMark (Huo et al., 2026b) use random pivots and generation-time thresholding; invariant-representation methods take another route (Liu et al., 2024). None address tables, and tables change what a text watermark can do. Language-model watermarks (Kirchenbauer et al., 2023) key each token on the tokens before it, so the mark lives inside one long sequence: it needs many tokens to accumulate, and an edit early on disturbs everything after it. A table cell is a short, self-contained text whose meaning does not depend on the cells around it. HashMark therefore keys each cell's bit on that cell alone and chooses its paraphrase without regard to any other cell. Evidence accumulates across cells rather than within one text: z grows with the number of cells, not their length, so a column of one-sentence summaries carries a strong mark; rewriting one cell cannot disturb another;

and sorting, subsampling or reordering rows leaves every bit intact. HashMark is also not a pseudorandom code (Christ & Gunn, 2024), which is built into token sampling and has no analog for independently drawn table rows (Remark 3; Appendix L.2).

1.3 PRELIMINARIES

Task and access. An owner marks a table before release, then tests a suspect table for provenance evidence. A *clean* table is one that has not been watermarked; the term does not imply that its records are error-free. *Blind* verification means that the original table is not needed. Verification uses the suspect table, the secret key S, and the public descriptor: the marked columns and their precisions and, for text, the encoder and the published mean embedding $\bar{e}$.

Threat model. Everything except the key is public: the attacker knows the algorithm, the encoder, the paraphraser and the descriptor, including every declared precision c. It may edit values, rewrite text, delete or add rows and columns, and publish the table inside a larger one. It lacks the key S, and without it every item looks valid or invalid with probability one half, so it cannot tell which items carry the mark. Removing the mark therefore means changing enough items, blindly, to push the valid fraction back to one half, which damages the data it wants to reuse. We do not consider an attacker who learns S, nor one that queries the owner's detector adaptively.

Keyed hashing. The secret key makes the numerical marking rule unpredictable to someone who does not know it. Formally, we assume a keyed hash $\mathcal{H}(S, \cdot)$, instantiated with HMAC-SHA-256, is a pseudorandom function (PRF; Appendix C.1). In the ideal model, a fixed set of distinct inputs independent of a random key receives independent fair bits. Column identifiers separate the inputs of different numerical columns. For text, the key is expanded into a direction in the embedding space, which we model as uniformly random. Unlike hash bits on distinct inputs, the bits it gives to similar sentences are correlated, because similar vectors tend to fall on the same side of any hyperplane. We write m for the number of distinct evidence items and v for their valid fraction.

2 HashMark

Section 1.1 described HashMark informally; here we give its algorithms and prove their properties, and Section 3 tests them. A column's score is

$$z = \frac{T - m/2}{\sqrt{m/4}} = 2\sqrt{m}\left(v - \frac{1}{2}\right), \qquad v = \frac{T}{m}, \tag{1}$$

where T counts the valid items among the m items of the column and v is the valid fraction.

2.1 HashMark: EMBEDDING

Algorithm 1 makes the embedding precise. For numbers (part a, which we call *Edit Mode*), the owner rounds the column to its declared precision c and moves each invalid value that occurs once by the smallest multiple of 10^{-c} that makes it valid. About half the values are already valid and stay put, and since every released value lies on the 10^{-c} grid, rounding to c or more decimals changes no bit. For text (part b), the owner releases the valid paraphrase furthest from the hyperplane, the one a small rewrite is least likely to flip. The average embedding $\bar{e}$ is computed on the original column and published; recomputing it on the tested column would make the signed distances sum to zero and erase the mark.

Quantized variant. The grid survives rounding but not noise: even a tiny random change usually moves a value to another grid point, and so to a fresh random bit. The quantized variant HashMark-Q therefore hashes which *bin*, a range of values, the value falls in, so a perturbation flips the bit only if it crosses into another bin. Bins are evenly spaced on a logarithmic scale, so each covers a fixed relative range. With K bins fitted on a reference table, let $\ell_{\min}$ and $\ell_{\max}$ be the logarithms of the smallest positive and the largest value, and $\delta = (\ell_{\max} - \ell_{\min})/K$. A positive value x falls in bin

$$b(x) = \min\Big\{\max\Big\{\Big\lfloor \tfrac{\ln x - \ell_{\min}}{\delta} \Big\rceil, 0\Big\}, K - 1\Big\},$$

where $\lfloor \cdot \rceil$ rounds to the nearest integer, and is valid when $\mathcal{H}(S, j \| b(x)) = 0$. An invalid value moves to the nearest valid bin, keeping its relative position inside the bin (Proposition 5); nonpositive values

Algorithm 1 HashMark embedding. Both parts move a cell into the keyed valid region; only the move differs. S is the secret key and $\mathcal{H}$ the keyed hash (HMAC-SHA-256); Enc is the public sentence encoder, which maps text to its embedding; Π is the public paraphraser, which returns a random rewrite of a sentence. The secret direction r_j is a random unit vector generated from the full HMAC-SHA-256 output on (S, j) rather than one bit.

(a) Edit Mode, numerical column

Input: column j with values $x_1, \dots, x_m$; key S; precision c; $\langle x \rangle$ writes x to c decimals
Output: watermarked column $x'_1, \dots, x'_m$
$x_i \leftarrow \mathrm{round}(x_i, c)$ for all i *// publish at c*
for $i = 1$ **to** m **do**
  if $\mathcal{H}(S, j \| \langle x_i \rangle) = 0$ **or** x_i occurs more than once in the column **then**
    $x'_i \leftarrow x_i$ *// left exact*
  else
    $k \leftarrow 1$
    while $\mathcal{H}(S,\ j \| \langle x_i + k \cdot 10^{-c} \rangle) \neq 0$ **do**
      $k \leftarrow k + 1$
    $x'_i \leftarrow x_i + k \cdot 10^{-c}$
return $x'_1, \dots, x'_m$

(b) Free-form text column

Input: column j with $x_1, \dots, x_m$; paraphraser Π; encoder Enc; key S; pool size q
Output: watermarked column and reference mean $\bar{e}$
$r_j \leftarrow$ direction seeded by $\mathcal{H}(S, j)$ *// secret*
$\bar{e} \leftarrow \frac{1}{m} \sum_i \mathrm{Enc}(x_i)$ *// published*
for $i = 1$ **to** m **do**
  $\mathrm{bit}(u) \equiv \frac{1}{2}\big(1 - \mathrm{sign}\langle \mathrm{Enc}(u) - \bar{e},\ r_j \rangle\big)$
  if $\mathrm{bit}(x_i) = 0$ **then**
    $x'_i \leftarrow x_i$ *// already valid*
  else
    $\mathcal{P} \leftarrow \{\Pi(x_i)\}_{\ell=1}^{q}$ *// q paraphrases*
    $\mathcal{V} \leftarrow \{u \in \mathcal{P} : \mathrm{bit}(u) = 0\}$
    if $\mathcal{V} = \emptyset$ **then**
      $x'_i \leftarrow x_i$ *// exhausted*
    else
      $x'_i \leftarrow$ the element of $\mathcal{V}$ farthest from the hyperplane
return $x'_1, \dots, x'_m$ and $\bar{e}$

Algorithm 2 HashMark detection. $\mathcal{H}$, Enc and r_j are as in Algorithm 1; $\langle x \rangle$ writes x to c_j decimals. Inputs: suspect table, key S, descriptor (marked columns, precisions c_j, encoder, reference mean $\bar{e}$), level α, number of fresh keys B; C is the number of marked columns.

(a) Numerical column j

$U \leftarrow$ distinct values of column j written to c_j decimals
$m \leftarrow |U|$
$T \leftarrow \#\{x \in U : \mathcal{H}(S, j \| \langle x \rangle) = 0\}$
$p_j \leftarrow \Pr[\mathrm{Bin}(m, \frac{1}{2}) \geq T]$ *// binomial tail*

(b) Text column j

$U \leftarrow$ distinct cells of column j; $r_j \leftarrow$ direction seeded by $\mathcal{H}(S, j)$
$T \leftarrow \#\{x \in U : \langle \mathrm{Enc}(x) - \bar{e}, r_j \rangle \geq 0\}$; $z_0 \leftarrow z$ of Eq. (1)
$z_1, \dots, z_B \leftarrow$ same score under B fresh random keys
$p_j \leftarrow \big(1 + \#\{k : z_k \geq z_0\}\big)/(B + 1)$ *// empirical*

Decision: WATERMARKED if $\min_j p_j < \alpha/C$, else NO EVIDENCE.

are not scored. Wider bins tolerate more noise but move values further, which makes them easier to detect (Appendices H.2 and J.1).

2.2 HashMark: DETECTION

Algorithm 2 gives each marked column a p-value, the probability that a table carrying no watermark (the *null*) scores at least as high, and combines them. For numbers the null is exact: distinct values under a random key get independent fair bits, so the valid count is binomial. For text it is not, because cells tested against one hyperplane have dependent bits; we compare the column with its own scores under B fresh keys, counting the real key as one more draw so that the p-value never falls below $1/(B+1)$. The quantized variant counts cells, since cells in one bin share a bit, and uses thresholds from clean tables. Testing C columns gives C chances of a lucky pass, so the *Bonferroni correction* tests each at level α/C, keeping the table-wide false-alarm rate below α (measured in Appendix D.1); for a single standard-normal score, level 0.001 means $z > 3.09$.

A *shared direction* means that every text cell in a column is tested against the same secret hyperplane. Cells with similar meaning then tend to fall on the same side together, so their bits are not independent coin flips. The *null variance* is how much the valid count T varies from one random key to the next when the column carries no watermark; with independent bits it is $m/4$, and shared directions change it. The next proposition gives it exactly.

Proposition 1 (Null variance under shared directions)**.** *Let $u_1, \ldots, u_m$ be nonzero vectors fixed independently of the secret (embeddings minus the published reference), let cell i use direction $r_{g(i)}$ for a group label $g(i)$, and let the directions of distinct groups be independent and uniform on the sphere. The count $T = \#\{i : \langle u_i, r_{g(i)} \rangle \geq 0\}$ has $\mathbb{E}[T] = m/2$ and*

$$\mathrm{Var}(T) \;=\; \frac{m}{4} \;+\; \frac{1}{\pi} \sum_{g} \sum_{\substack{i<j \\ g(i)=g(j)=g}} \arcsin\!\left(\frac{\langle u_i, u_j \rangle}{\|u_i\| \, \|u_j\|} \right).$$

(Proof deferred to Appendix C.4.) The pairwise term comes from a classical fact (Goemans & Williamson, 1995; Charikar, 2002): two vectors at angle θ fall on the same side of a random hyperplane with probability $1 - \theta/\pi$; Our default uses one direction per column, so every pair of cells contributes; groups arise only in the per-row ablation (Appendix I.1.2).

The arcsines can widen or narrow the null; without centering, they widen it, because sentence embeddings all point roughly the same way. The identity holds for any fixed column, including a release, but fixes only the variance, not the shape of the tail, which is why text is decided against fresh keys. Stouffer summaries, which add several column scores and divide by the square root of their number, describe combined evidence but do not decide (Appendices B.4 and F.1.1).

2.3 Properties of HashMark

We now characterize how much evidence embedding can install (capacity) and how much of it an attack leaves (survival).

2.3.1 Capacity

Capacity is how much evidence the owner can install. For numbers, the search needs only a few grid steps, and the next theorem bounds both the largest change to any value (L_∞) and how far each column's distribution moves (the Wasserstein distances W_1 and W_∞: how far values must move, on average or at most, to turn one distribution into the other), relative to the data already rounded to the grid.

Theorem 1 (Numerical Fidelity)**.** *In the ideal random-function (random-oracle) model, let $\mathbf{X}$ be a fixed dataset of N numerical values on the declared precision grid and $\mathbf{X}_w$ its Edit Mode release. Assume each search visits distinct canonical values and can continue until acceptance; repeated inputs are left unchanged. The expected L_∞ distortion is bounded by:*

$$\mathbb{E}[||\mathbf{X} - \mathbf{X}_w||_\infty] \leq (\log_2 N + 2) \cdot 10^{-c}.$$

Moreover, for every $\eta \in (0, 1)$, with probability at least $1 - \eta$,

$$||\mathbf{X} - \mathbf{X}_w||_\infty \leq \lceil \log_2(N/\eta) \rceil \cdot 10^{-c},$$

and the distribution of values in every column (its empirical marginal F) satisfies $\mathbb{E}[W_1(F_{\mathbf{X}}, F_{\mathbf{X}_w})] \leq 10^{-c}$ and $W_\infty(F_{\mathbf{X}}, F_{\mathbf{X}_w}) \leq ||\mathbf{X} - \mathbf{X}_w||_\infty$.

(Proof deferred to Appendix C.4.)

For text, capacity depends on how different a cell's paraphrases are: ten near-copies offer fewer chances than ten diverse rewrites. A cell can be marked when at least one candidate lies on the valid side.

Proposition 2 (Capacity from pool geometry)**.** *Let a cell's candidates, the original and q paraphrases ($q + 1$ in total in this proposition and the geometry tables), have nonzero centered embeddings $u_0, \ldots, u_q$, let U be the matrix whose rows are these embeddings, and let the direction r be a Gaussian vector (which points in a uniformly random direction), independent of the pool. Define* exhaustion *as the event that every candidate lands on the invalid side. Its probability is the orthant probability that all $q + 1$ coordinates of a correlated Gaussian vector are negative:*

$$p_i \;=\; \Pr\big[\, Ur < 0 \,\big], \qquad Ur \sim \mathcal{N}\big(0,\, UU^\top\big),$$

which depends on the pool only through the angles between its candidates: $p_i = \frac{1}{2}$ for identical candidates and $p_i = 2^{-(q+1)}$ for orthogonal ones. Assume selection leaves m distinct evidence items, so deduplication does not change the denominator. Then the expected valid fraction is $\mathbb{E}[v] = 1 - \frac{1}{m} \sum_i p_i$, and Eq. (1) gives its expected detection statistic.

(Proof deferred to Appendix C.4.)

In words, the orthant probability is the chance that a random direction puts every candidate on the invalid side at once; the more alike the candidates, the more often this happens. Paraphrases of one cell are so alike that exhaustion stays far above the independent-candidate value $2^{-(q+1)}$ (Appendix E.1).

2.3.2 SURVIVAL

Removing items shrinks the evidence. Under uniform sampling, the valid fraction stays the same in expectation while the score falls with the square root of the number of items left; this describes the mean, not every subset.

Proposition 3 (Uniform subsetting of evidence)**.** *Fix a release with m distinct evidence items, $T = mv$ of them valid, and a descriptor. Retain a uniformly sampled subset of m' items without replacement, leaving their predicates unchanged. Then $T' \sim \mathrm{Hypergeom}(m, T, m')$ and*

$$\mathbb{E}[z'] = 2\sqrt{m'}(v - \tfrac{1}{2}).$$

The exact detection probability at a threshold z_{th} is the corresponding hypergeometric upper tail.

(Proof in Appendix C.4.) The hypergeometric law describes drawing without replacement. The expected score scales as $\sqrt{m'/m}$, so keeping more than $z_{\mathrm{th}}^2/[4(v-1/2)^2]$ items keeps the *mean* above threshold when $v > 1/2$; ten valid items already have null probability $2^{-10} < 0.001$. Section 3 measures row deletion and content-dependent edits directly.

Appendix H.3 tests the quantized survival model under rounding and truncation, and Appendix E.2 tests the random-hyperplane angle model after selection.

3 EXPERIMENTAL FINDINGS

We test HashMark on real tables and ask three questions: how much text can be marked, what survives when the table is edited, and how HashMark compares with existing watermarks. We find that most text cells can be marked, the mark survives heavy editing, and HashMark detects as well as the strongest baselines while maintaining unwatermarked fidelity and requiring no generator. A table is declared watermarked when some marked column passes its test at a table-wide false-alarm level of 0.001 (Algorithm 2). Numerical tables are standard UCI benchmarks; the mixed tables (Airbnb, California, Wine) each have a free-text column (Appendix A.1).

Capacity. Almost any number can be made valid, so numerical capacity is limited only by repeated values, which we leave untouched: 93% of the 968 distinct values in a 500-row `Shoppers` table end up valid, and 99% on `Wilt`. Text is harder, because paraphrases of one sentence are close to one another. Proposition 2 predicts the fraction of cells with no valid paraphrase to within 0.0032 at every pool size on four corpora (Figure 3), and still holds after changing the encoder or paraphraser (Appendix E.2); treating paraphrases as independent would predict almost none.

Comparison with tabular baselines. We compare with two watermarks that act during generation (MUSE, TabWak) and two that edit a finished table (TabularMark, WGTD). We measure fidelity with a classifier trained to tell watermarked rows from clean ones (C2ST, reported as the benchmark's normalized score, 1 = indistinguishable) and with the MLE (machine-learning efficacy) gap, the accuracy lost by a prediction model trained on the watermarked table. HashMark moves values by 10^{-6}, below what either metric resolves, so it matches the unwatermarked table to three decimals while detecting every watermarked table (AUC 1.000: a watermarked table always outscores a clean one), without the original table or a generator (Table 2). On `Shoppers`, TabularMark has a higher C2ST, and WGTD, TabularMark and MUSE a smaller MLE gap. Marking $10{,}000$ rows of numbers takes 0.20 s and verifying them 0.07 s on a laptop CPU; text takes about 0.6 s per cell on a laptop GPU, mostly paraphrasing (Appendix A.2).

Mixed tables and attacks. Table 3 marks three mixed tables end to end; prediction models trained on the watermarked versions lose less than a fifth of a point of accuracy. Airbnb and California carry numerical and text evidence, and Wine only text. The two kinds of evidence support each other.

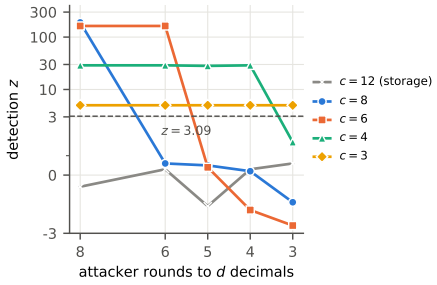


Figure 2: Detection after an attacker rounds the Airbnb coordinates to d decimals, for each declared precision c. A release survives rounding to c or more decimals exactly and loses the mark one decimal below; storing at full precision ($c = 12$) survives no rounding.

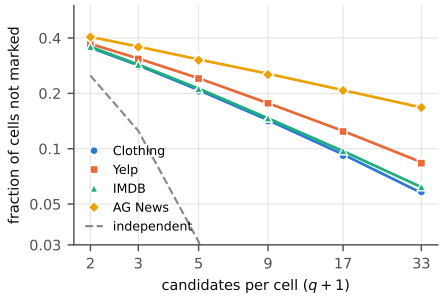


Figure 3: Fraction of text cells with no valid candidate, against candidates per cell, on four corpora ($m = 500$ cells each). Markers are measured over 100 keyed directions per cell; lines are predictions from the candidates' geometric similarity (Proposition 2) on the same pools; the dashed line treats candidates as independent. Values are tabulated in Appendix E.1.

Table 2: Fidelity and detection against tabular watermarking baselines at 500 rows. C2ST: normalized score (1 = indistinguishable); MLE gap: downstream accuracy delta (lower is better); AUC: `Adult` / `Shoppers`. Darker / lighter : best/second-best watermark per fidelity column. *Blind*: no original table needed. TabWak detects with its own model; TabularMark locates its key rows through a content hash (Appendix J.1).

			`Adult`		`Shoppers`		
Method	Blind	Needs gen.	C2ST ↑	MLE gap ↓	C2ST ↑	MLE gap ↓	AUC
w/o watermark	N/A	N/A	0.996	0.017	0.974	0.017	N/A
TabWak	✓	✓	0.769	0.048	0.548	0.132	0.990 / 0.990
TabularMark	✗	✗	0.987	0.021	0.975	0.013	1.000 / 1.000
WGTD	✓	✗	0.978	0.019	0.887	0.008	1.000 / 1.000
MUSE (joint score)	✓	✓	0.883	0.017	0.950	0.015	1.000 / 1.000
HashMark (Edit Mode)	✓	✗	0.996	0.017	0.974	0.017	1.000 / 1.000

Rounding Airbnb's coordinates to two decimals erases the numerical mark, but the text alone still rejects ($p = 5.0 \cdot 10^{-5}$); replacing every summary with another of the owner's paraphrases erases the text mark, but the coordinates still reject. Only both attacks together remove the verdict. The text mark also survives a fresh paraphrase by a different model and one that keeps paraphrases on the wrong side of a direction estimated from the release, and no clean or wrong-key control is ever accepted (Table 16).

Against structural attacks, we report the true positive rate (TPR), the share of watermarked tables detected at a 1% false-positive rate, the operating point of the MUSE benchmark we compare against. With 80% of the rows or cells replaced, HashMark still detects every `Shoppers` table, while MUSE detects 26% and 34%, and 26–53% on `Magic` and `News` (Table 34). MUSE is stronger on `Beijing`, which has a single markable column (Table 36), and replacing or altering 80% of the columns of `News` lowers HashMark to about 0.4 (Table 33). Deletion and dilution follow Proposition 3 (Appendix G.1). Random noise erases exact values by design; the quantized variant with 8 bins still detects 99% of `Shoppers` tables when each value is multiplied by $1 + 0.4\varepsilon$ (ε standard normal), and 70% and 92% on `Magic` and `News`, but bins that wide are visible to a classifier (Appendix J.1).

Attack cost. Removing a mark should cost the attacker. For text, Figure 4 rewrites 1,000 reviews even more aggressively and plots the score against how far the text drifts, measured by the cosine similarity of the embeddings (1 for unchanged text). The score falls only as the text drifts, and after three rounds at the owner's own paraphrasing temperature (the setting that controls how random the

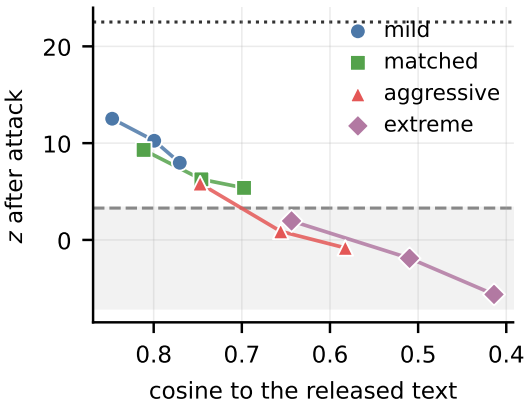


Figure 4: Raw global-hyperplane z after rewriting vs. cosine to the released text, $m = 1000$ Clothing reviews. Lines: paraphraser temperature T (its sampling randomness: mild 0.8, matched 1.2, aggressive 1.5, extreme 2.0); points: rounds 1–3. Dotted: no attack (22.5); dashed: $z = 3.29$, above the calibrated threshold for this column (null spread 0.80).

Table 3: Mixed-table scores (descriptive) and downstream prediction changes over 20 paired seeds. z/SD: text z over its fresh-key null SD, not a p-value. *Wrong*: text z under an unrelated key, within 2 null SDs. *Utility*: accuracy change in points (R^2 for California, a regression task). *Det.*: calibrated decision (Table 16), which never accepts a wrong key.

Dataset	num. z	text z	z/SD	Wrong	Utility	Det.
Airbnb	163.2	83.1	11.5	8.1	-0.18 pt	✓
California	141.9	118.6	10.0	24.3	$+3\times10^{-4}$ R^2	✓
Wine	N/A	34.2	12.5	2.4	-0.15 pt	✓

rewrites are) it is still above 3.29, stricter than this column's calibrated threshold (Appendices I.1.1 and I.4). For numbers, the declared precision decides what survives (Figure 2): rounding to c or more decimals leaves every bit intact, and one decimal less removed the mark. Declaring $c = 6$ puts that point at about 11 cm of latitude; storing values at full precision survives no rounding at all (Appendix H.1).

Design ablations. We also test a naive composition of separate watermarks, WGTD-style binning for numbers and SemStamp-style selection for text, which separates marked from clean releases by only 2.8 null standard deviations against 67.8 for HashMark (Appendix F.1.2); Appendix L collects further diagnostics.

Data utility and text quality. Accuracy on numerical tasks changes by at most 0.38 points (Appendix K.2); a task that depends on the text pays more, 0.86 points when every Wine description is rewritten. Judged by independent language models, the default release rewrites a third of the cells and keeps mutual entailment (each text supporting the other) at 0.75 over the column; on the rewritten cells, choosing a valid paraphrase rather than a random one costs 7 points of mutual entailment and raises judged contradictions from 8% to 18% (Appendix K.1).

4 Conclusion

A table's numbers and its free-form text can carry one watermark, and each protects the other. HashMark marks both under one secret key and verifies a suspect copy without the original table or its generator. Counting distinct values, testing text against fresh keys, and declaring the published precision separate evidence from chance and fix in advance what the mark survives. Because each cell is marked on its own, the evidence grows with the number of cells rather than their length, so even columns of one-sentence summaries carry a strong mark, and sorting, subsampling or replacing most of a table leaves the rest verifiable. Marking costs 0.2 s per 10,000 rows of numbers and about 0.6 s per text cell, so an owner can mark a table as part of an ordinary release. The same approach applies to any column for which a public model can propose meaning-preserving alternatives; next, carry more than one bit so a leaked copy identifies its recipient, and mark columns such as addresses or codes that have few valid alternatives. Some limits remain: exact values do not survive noise, the noise-tolerant variant is visible to a classifier, the shared text direction can be estimated from a release, and marking text costs accuracy when a task depends on that text.

Ethics Statement

No human subjects were involved; all datasets are public, and automated models judge text quality. Watermark detections support audits together with provenance records. Because false-positive rates vary across fixed keys, calibrate a deployed key on held-out clean data before using a detection to accuse anyone of misuse. Publishing a semantic reference mean warrants a privacy assessment for sensitive columns, and releases follow source licenses and established controls for personal text.

Reproducibility Statement

Code for HashMark, the baseline harnesses, and seeded scripts for the experiments, all run on public datasets, are available at `https://anonymous.4open.science/r/hashmark-artifact-3BE8`.

Use of Large Language Models

Large language models were used as general-purpose assistive tools in preparing this work: for drafting and editing prose. We take responsibility for the final content of this work and have manually reviewed all the output. This use is distinct from the role language models play *inside* the method, where a paraphraser generates the candidate pools of Section 2.1 and a sentence encoder supplies the embeddings; those uses are experimental apparatus, are pinned to specific model identifiers in the repository, and are described in Appendix A.2.

References

Scott Aaronson. 'reform' ai alignment with scott aaronson. AXRP - The AI X-risk Research Podcast, 2023. URL `https://axrp.net/episode/2023/04/11/episode-20-reform-ai-alignment-scott-aaronson.html`.

Rakesh Agrawal and Jerry Kiernan. Watermarking relational databases. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pp. 155–166. VLDB Endowment, 2002.

Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 380–388, 2002.

Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In Leonid Reyzin and Douglas Stebila (eds.), *Advances in Cryptology – CRYPTO 2024*, pp. 325–347, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-68391-6.

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *COLT*, 2024.

Dhruvil Dave. Github commit messages dataset. Kaggle Dataset, 2023. URL `https://www.kaggle.com/datasets/dhruvildave/github-commit-messages-dataset`.

Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly-detectable watermarking for language models. *IACR Communications in Cryptology*, 1(4), 2025. ISSN 3006-5496. doi: 10.62056/ahmpdkp10.

Liancheng Fang, Aiwei Liu, Henry Peng Zou, Hengrui Zhang, and Philip S. Yu. RINTAW: A robust invisible watermark for tabular generative models. In *The 1st Workshop on GenAI Watermarking*, 2025. URL `https://openreview.net/forum?id=3K4oAgZTcO`.

Liancheng Fang, Aiwei Liu, Henry Peng Zou, Yankai Chen, Hengrui Zhang, Zhongfen Deng, and Philip S. Yu. MUSE: Model-agnostic tabular watermarking via multi-sample selection. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=R1QuNKyVOw`.

Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2nd edition, 2019. URL https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/.

Hengzhi He, Peiyu Yu, Junpeng Ren, Ying Nian Wu, and Guang Cheng. Watermarking generative tabular data, 2024. URL https://arxiv.org/abs/2405.14018.

Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. SemStamp: A semantic watermark with paraphrastic robustness for text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 4067–4082, 2024a.

Abe Bohan Hou, Jingyu Zhang, Yichen Wang, Daniel Khashabi, and Tianxing He. k-SemStamp: A clustering-based semantic watermark for detection of machine-generated text. In *Findings of the Association for Computational Linguistics (ACL)*, 2024b.

Jiahao Huo, Shuliang Liu, Bin Wang, Junyan Zhang, Yibo Yan, Aiwei Liu, Xuming Hu, and Mingxun Zhou. PMark: Towards robust and distortion-free semantic-level watermarking with channel constraints. In *International Conference on Learning Representations (ICLR)*, 2026a. arXiv:2509.21057.

Jiahao Huo, Wenjie Qu, Yibo Yan, Kening Zheng, Jiaheng Zhang, Xuming Hu, Philip S. Yu, and Mingxun Zhou. SAMark: A self-anchored text watermarking with paragraph-level paraphrase robustness. *arXiv preprint arXiv:2605.25796*, 2026b.

Brian Johnson. Wilt. UCI Machine Learning Repository, 2013. URL https://archive.ics.uci.edu/ml/datasets/Wilt. DOI: https://doi.org/10.24432/C5KS4M.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics and Probability Letters*, 33(3):291–297, 1997. ISSN 0167-7152. doi: https://doi.org/10.1016/S0167-7152(96)00140-X. URL https://www.sciencedirect.com/science/article/pii/S016771529600140X.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6p8lpe4MNf.

Pratyush Maini, Mohammad Yaghini, and Nicolas Papernot. Dataset inference: Ownership resolution in machine learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=hvdKKV2yt7T.

Jiaqi Mu and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations (ICLR)*, 2018.

Dung Daniel Ngo, Archan Ray, Akshay Seshadri, Daniel Scott, Saheed Obitayo, Niraj Kumar, Vamsi K. Potluru, Marco Pistoia, and Manuela Veloso. Adaptive and robust watermark for generative tabular data. In Emilija Perković and Daniel Malinsky (eds.), *Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence*, volume 337 of *Proceedings of Machine Learning Research*, pp. 4796–4831. PMLR, 17–21 Aug 2026. URL https://proceedings.mlr.press/v337/ngo26a.html.

PromptCloud. Amazon product dataset 2020. Kaggle Dataset, 2020. URL https://www.kaggle.com/datasets/promptcloud/amazon-product-dataset-2020/data.

Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Radioactive data: tracing through training. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020. URL http://proceedings.mlr.press/v119/sablayrolles20a.html.

C. Okan Sakar, S. Olcay Polat, Mete Katircioglu, and Yomi Kastro. Real-time prediction of online shoppers'purchasing intention using multilayer perceptron and lstm recurrent neural networks. *Neural Computing and Applications*, 31(10):6893–6908, 2019. doi: 10.1007/s00521-018-3523-0. URL https://doi.org/10.1007/s00521-018-3523-0.

Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. Watermarking relational databases using optimization-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 20(1):116–129, 2008. doi: 10.1109/TKDE.2007.190668.

Xingjian Shi, Jonas Mueller, Nick Erickson, Mu Li, and Alexander J. Smola. Benchmarking multimodal automl for tabular data with text fields. In *NeurIPS Datasets and Benchmarks Track*, 2021.

R. Sion, M. Atallah, and S. Prabhakar. Rights protection for relational data. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*, pp. 98–109, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 158113634X. doi: 10.1145/872757.872772. URL https://doi.org/10.1145/872757.872772.

Yuyang Xia, Yaoqiang Xu, Chen Qian, Yang Li, Guoliang Li, and Jianhua Feng. TableMark: A multi-bit watermark for synthetic tabular data. *arXiv preprint arXiv:2603.13722*, 2026.

Hengrui Zhang, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4Ay23yeuz0.

Yizhou Zhao, Xiang Li, Peter Song, Qi Long, and Weijie J Su. Robust spectral watermark for synthetic tabular data. *Statistical Learning and Data Science Section B*, 2026. ISSN 3051-3901. URL https://openreview.net/forum?id=1g39uYtxPx.

Yihao Zheng, Haocheng Xia, Junyuan Pang, Jinfei Liu, Kui Ren, Lingyang Chu, Yang Cao, and Li Xiong. Tabularmark: Watermarking tabular datasets for machine learning. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, pp. 3570–3584, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400706363. doi: 10.1145/3658644.3690373. URL https://doi.org/10.1145/3658644.3690373.

Chaoyi Zhu, Jiayi Tang, Jeroen M. Galjaard, Pin-Yu Chen, Robert Birke, Cornelis Bos, and Lydia Y. Chen. Tabwak: A watermark for tabular diffusion models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=71pur4y8gs.

A Experimental Protocols and Reading Guide

Our experiments answer four questions. Does the ownership decision raise false alarms at the rate we claim? How much evidence can we install in each kind of column? Which edits remove it, and what do they cost the attacker? What does marking cost in fidelity and downstream accuracy? Table 4 maps each claim to the appendix that supports it. We call the keyed rule that decides validity the *predicate*: the hash for numbers, and for text the side of the secret hyperplane on which a centered embedding falls (the *semantic predicate*, a form of locality-sensitive hashing, LSH). For text we release either the valid paraphrase farthest from the hyperplane (*margin* selection, our default) or, in ablations, the first valid one (*first-valid*). We first give the protocols and the decision rule, then the evidence, grouped under calibration, capacity, robustness and utility.

Table 4: Index of claims, evidence roles, supporting appendix sections, and representative tables. *Primary* results directly support a body claim; *supporting* results test its scope; *diagnostic* results explain a mechanism without defining the deployed detector; *ablation* results evaluate a non-default design; *legacy* results use an earlier protocol; and *extension* results lie outside the central mixed-table evaluation.

Claim or question	**Role**	**Appendix**	**Representative evidence**
Distinct-input, fixed-key, and family-wise calibration	primary	Apps. D.1, D.1.1, F.1.1	Tabs. 8, 10, 15
Text null depends on direction sharing	primary / ablation	App. C.2, Sec. I.1.2	Tab. 26
Pool geometry predicts text capacity	primary	Sec. 2.3.1; Apps. E.1, E.2	Tab. 11
Deletion and dilution follow sample size	primary	App. G.1	Tab. 18
Exact and quantized predicates under rounding	primary	Apps. H.1, H.3	Tab. 22
Quantized fidelity and noise tolerance	supporting	Apps. H.2, J.1	Tabs. 41, 30
Fresh paraphrasing and direction estimation (global default)	supporting	Secs. I.1.1, I.1.1, I.1.1	Tabs. 27, 25, 24
Row seeding and the attacker-cost frontier	ablation	Sec. I.1.2, App. I.3	Tabs. 26, 28
Mixed-table detection and component failures	primary	App. F.1	Tabs. 13, 16
Text quality and downstream utility	primary	Apps. K.1, K.2, F.1	Tabs. 38, 39
Structural attacks against blind baselines	supporting	App. J.1	Tabs. 32, 31
Detector diagnostics	diagnostic	App. L.1	Tab. 41

Score conventions. A watermarked column has more valid items than chance, so only large positive scores count as evidence. When the score of an unmarked column is standard normal, $z > 3.09$ happens with probability 0.001 and $z > 3.29$ with probability about 0.0005. One diagnostic, for text scored without subtracting the mean embedding, counts large scores of either sign, $|z| > 3.29$, which also has probability 0.001. On mixed tables we sometimes add column scores into one summary (a Stouffer score, the sum divided by the square root of the number of columns); the summary only describes the evidence, and the decision is always made column by column with the Bonferroni correction.

A.1 Dataset Details

Wilt. Wilt (Johnson, 2013) is the public UCI remote-sensing dataset for identifying diseased trees from Quickbird spectral and texture features. It contains $4{,}839$ records, five predictors, and a binary target. We use it for synthetic-table detection and utility experiments. The dataset is licensed under CC BY 4.0.

California Housing Prices. This 1990 U.S. Census dataset (Kelley Pace & Barry, 1997; Géron, 2019) contains 20,640 districts with demographic and geographic predictors. In the utility experiments the downstream task is five-class classification of `ocean_proximity`, which is why accuracy is reported. We use it in numerical fidelity, utility, and robustness diagnostics. This dataset is distinct from the AutoML California listings table below.

Shoppers. The UCI Online Shoppers Purchasing Intention dataset (Sakar et al., 2019) contains 12,330 sessions, 17 predictors, and a binary purchase target. It is the principal numerical calibration, synthetic-generation, robustness, and baseline dataset. The dataset is licensed under CC BY 4.0.

Adult. The UCI Adult census extract used in Table 40 contains 45,222 records after rows with missing values are removed, with six continuous and eight categorical predictors and a binary income target. It is used only in the baseline comparison. The UCI release is licensed under CC BY 4.0.

Melbourne Airbnb. The multimodal benchmark table (Shi et al., 2021) contains 18,316 training listings with coordinates, structured listing attributes, and a free-text summary. We use latitude and longitude as numerical evidence, the summary as text evidence, and ten-class price prediction for downstream utility. Tables 3 and 13 report the resulting mixed-table experiment.

AutoML California listings. This multimodal benchmark table (Shi et al., 2021) contains 37,951 training property sales with tax and lot measurements, structured attributes, and a free-text summary. It supports the mixed-table regression experiment in Tables 3 and 13. It is not the California Housing Prices census dataset above.

Wine Reviews. The wine table contains 84,123 reviews after preprocessing, with price, country, province, points, variety, and free-text description fields. It is the text-only end-to-end case because its whole-dollar price is left exact (marking it on the unit grid is visible; Appendix B.3); grape-variety prediction measures downstream utility. The family-wise experiment uses a seeded 4,000-row subset.

Women's E-Commerce Clothing Reviews. The public Kaggle table contains 23,486 rows with numerical, categorical, and review-text fields. We retain nonempty reviews with downstream labels and use seeded subsets of 300 to 4,000 cells for semantic calibration, capacity, attack, quality, and utility experiments. The downstream task predicts `Recommended IND` from structured features and review text.

Yelp and IMDB Reviews. We use public Yelp and IMDB review corpora to test whether candidate-pool geometry transfers across informal review domains. Each capacity or allocation arm uses a seeded sample of 500 texts filtered to 150–400 characters; these experiments do not train downstream classifiers on either corpus.

AG News and DBpedia. AG News supplies formal news text for capacity, encoder-transfer, direction-policy, and attack ablations. DBpedia supplies encyclopedic text only for the candidate-pool cost study. Each reported corpus arm uses a seeded sample of 500 texts filtered to 150–400 characters. Their labels are not used as downstream targets in these experiments.

Amazon ASINs. We used the Amazon Product Dataset 2020 (PromptCloud, 2020). For our experiments, we parsed the dataset only to extract the unique identifiers for Amazon products, generating 30,000 actual ASINs. This dataset is licensed under CC0.

Gitcommit Hashes. We used the Gitcommit Messages dataset (Dave, 2023). It contains 4.3 million records, from which we only extracted the hashes for the gitcommit messages. The dataset is licensed under the Open Data Commons Attribution License (ODC-By) v1.0.

A.2 Text Protocols

This section gives the settings shared by our text experiments. By default we use one secret hyperplane per text column, applied to embeddings after subtracting the column's mean embedding (Section 2.1); hyperplanes seeded per row are an ablation. The table below lists the four protocols we use. Each protocol measures something different, so we compare scores only within a protocol.

Protocol family	Candidate pool and direction	Interpretation
Default	original plus q paraphrases; one centered column direction	survives row reordering; direction estimable from a release
Row-seeded ablation	original plus q paraphrases; row-seeded hyperplane	requires intact context columns and text–row pairing
Owner-pool attack	the owner's own unreleased paraphrases	gives the attacker more than the release
Earlier pool diagnostic	q paraphrases only; one global hyperplane	measures pool coverage, not the valid fraction

A.2.1 Common Generation and Attack Settings

Baselines. We compare two ways of deciding validity for text on the same paraphrases: hashing the string itself, and our embedding-based rule. Appendix I.1.1 adds a baseline adapted from SemStamp (Hou et al., 2024a), which uses a hyperplane without subtracting the mean; it isolates the effect of that subtraction rather than reproducing all of SemStamp.

How paraphrases are generated. Candidates come from a public T5-based paraphraser (`humarin/chatgpt_paraphraser_on_T5_base`) prompted with "`paraphrase: {cell}`". For each cell we decode $q = 8$ candidates in a single batched call with nucleus sampling (top-$p = 0.95$, temperature 1.2, `max_new_tokens`$= 64$) under a fixed generation seed, so the pool is reproducible. Embeddings are computed with `all-MiniLM-L6-v2` (384-dimensional, L_2-normalized) and centered on the column mean $\bar{e}$. A cell is valid when its centered embedding falls on the positive side of the secret hyperplane. Among valid candidates we release the one farthest from the hyperplane; as an ablation we also report releasing the first valid candidate. Our basic attack paraphrases each released cell once more with the same model and settings but fresh randomness, so every variant faces the same rewrite.

Detection and cost. The watermarked column scores $z = 45.09$ over $m = 4000$ cells; an unmarked column scores about 0. Paraphrases of one sentence sit close together in embedding space, so they tend to land on the same side of the hyperplane. As a result, 14.4% of cells have no valid paraphrase among 8, far more than the $2^{-8} = 0.4\%$ we would see if each paraphrase were an independent coin flip. For the same reason a valid paraphrase takes more tries to find than the 2 expected for coin flips: 2.12 candidates per cell when we stop at the first valid one, and 4.58 when we look for the farthest. Cells with no valid paraphrase stay invalid, so only $v = 0.856$ of the column ends up valid, and z must be computed from that fraction Eq. (3). Downstream accuracy stays within 0.74 points of the unwatermarked baseline.

Marking time. Generating paraphrases dominates the cost of marking text. On an Apple M5 laptop GPU, generating 4 candidates per cell with the T5-base paraphraser (sampling temperature 1.2, batches of 16 cells), embedding them with MiniLM and choosing the valid one takes 121.6 s for 200 distinct Airbnb summaries, or 0.61 s per cell. Cost grows linearly with the number of cells and the number of candidates, and cells can be marked in parallel, since each is marked on its own.

A generation-budget artifact in the released text, and its cost. Most text experiments cap each paraphrase at 64 tokens, which cuts off long inputs; the text-quality study (Appendix K.1) regenerates with a 512-token cap so that truncation does not count against quality. The verifier subtracts the published mean of the original column. Subtracting the mean of the tested column instead would force the projections to sum to zero and erase the mark.

A.2.2 Row-Conditioned Verification (Ablation)

In this ablation the verifier derives a separate hyperplane for each row from the key and a few stable fields of that row, named in the descriptor. When every row has different fields, the bits are independent and the score is standard normal; when rows repeat, we group them and use fresh keys as for the global hyperplane. Per-row hyperplanes defeat the attacker's estimate of a single shared direction, but the verifier then needs those fields and needs each text cell to stay in its row. Our row-seeded experiments keep both intact; shuffling the text or deleting a named field breaks verification.

A.2.3 GLOBAL-HYPERPLANE SETUP

We use the Women's E-Commerce Clothing Reviews dataset (23,486 rows), retaining rows with non-empty `Review Text` and labels and sampling $m = 4000$ with a fixed seed. Classifier features are `Age`, `Rating`, and `Positive Feedback Count`; one-hot encodings of `Division Name`, `Department Name`, and `Class Name`, including missing-value indicators; and the sentence embedding of the text. The label is `Recommended IND` (82.0% positive), and only the text column is watermarked. Downstream XGBoost and 100-tree Random Forest classifiers are averaged over five random $75/25$ splits.

By default one secret unit direction r is derived from the key and used for the whole column. Directions from different keys are nearly perpendicular (average $|\langle r_1, r_2\rangle| \approx 0.06$ in $d = 384$ dimensions), and a random input is valid half the time (0.4989 over 10^4 random unit vectors). Our script `text_watermark.py` saves each stage (paraphrasing, selection, attack, scoring) to disk, so scores can be recomputed without generating the paraphrases again.

B CONSTRUCTION, DETECTION, AND SCOPE

B.1 CONSTRUCTION DETAILS

Algorithm 1 in Section 2.1 gives the embedding for numbers and text.

Per-cell distortion. Theorem 1 bounds the largest move over all N values, which grows like $\log N$. A single value moves much less. Each step lands on a fresh random bit, so the number of steps needed is like the number of coin flips until the first head, one on average; hence the expected move of a value is at most 10^{-c}, and repeated values, which we leave alone, do not move at all.

Scope. Why the surface string cannot serve as the text predicate, and the boundary with language-model watermarking, are discussed in Appendix B.2; coverage by column type is tabulated in Appendix B.3.

Remark 1 (Static releases)**.** *If a release is never reordered or subset, a sparse variant suffices: a location key selects $n' \ll n$ of the table's n cells, a value key assigns each a target bit, and each selected cell is edited until its hash matches. Because it keys on absolute coordinates, it detects edits to those cells rather than surviving row deletion.*

B.2 SCOPE OF THE PREDICATE

Remark 2 (Why not hash the string itself?)**.** *The simplest choice is to hash the text itself with $\mathcal{H}(S, \cdot)$ and release the first paraphrase that hashes to 0. It needs no encoder and detection stays blind. We tried it and do not use it. The mark then lives in the exact wording, and any paraphrase changes the wording, so one more paraphrase destroys it: the score falls from $z = 62.96$ to $z = 0.51$, which is chance. There is also no text analog of binning numbers. Our embedding-based rule looks at meaning instead of wording, so it keeps evidence under many rewrites; the cost is that similar sentences get similar bits, which is why text needs its own calibration.*

Remark 3 (Why this is not an LLM watermark)**.** *Our detector counts keyed bits and tests the count, as watermarks for language-model output do, so one may ask whether* HashMark *also watermarks such output. It does not, for three reasons. First, our bit depends only on the item, $\mathcal{H}(S, x)$. Applied to single words, a word would be valid everywhere or nowhere, so the model could use only half its vocabulary; that is why language-model watermarks key each word's bit on the words before it (Kirchenbauer et al., 2023; Aaronson, 2023). The same dependence is what* HashMark *avoids in a table: each cell is short and self-contained, so we key it on itself alone, and editing one cell never affects another. Second, releasing only valid items changes the distribution of what is released, whereas the strongest language-model watermarks leave the output distribution unchanged (Christ et al., 2024). Third, language-model watermarks skip words that have no real alternative; our counterpart for tables, counting each distinct value once (Section B.4.1), solves a different problem, repeated values, and does not carry over.*

B.3 Coverage by Column Type

The marking is chosen per column (Table 5); none of it needs a generative model.

Table 5: How HashMark marks each column type.

Column type	Marking
Continuous quantity	exact hash on the declared 10^{-c} grid (Section 2.1)
Integer quantity (interval or ratio scale)	exact hash on the unit grid (below)
Categorical, including integer-coded	never changed; every value counts as valid, so never decides
Free-form text	paraphrase selection under the semantic predicate (Section 2.1)

Integer columns are decided by metadata, not by dtype. The integer rows of Table 5 carry the same storage type and are treated differently, because an integer column can hold either a quantity that has been discretized or a category that has been coded as a number, and the released table does not say which. What matters is whether the integers measure an amount. A price in whole dollars, a review count or a duration in seconds is an amount: neighboring values are close, and moving one by a unit is a small change. A marital-status field coded 1 for single and 2 for married is a label: the numbers name categories, and moving a value by one does not nudge an amount, it states something false about a person. Ordered codes such as a five-point satisfaction scale sit in between, ordered but without a meaningful unit, and we do not mark them.

This is not only about fidelity. On labels there is no notion of distance, so "move to the nearest valid value" means nothing, and our survival analysis (Proposition 5), which is stated in terms of a bin width, does not apply.

We therefore mark an integer column only when four things hold: it measures an amount, it has enough distinct values to reach the threshold, one unit is small compared with its spread, and no other column is computed from it. `price` satisfies all four and we still decline it, on the occupancy cost measured below; a dependents count fails three of them.

Where the scale permits it, we mark an integer column on its unit grid. Each cell tosses a fair coin for its direction and then steps one unit at a time that way until its keyed bit is valid. The coin needs no key, since detection only checks the released value, and the random direction leaves the column's mean unchanged in expectation. Every released value is valid, so every distinct value counts as evidence: on the $4{,}000$-row `wine` table, `price` reaches $z = 9.8$ at a mean displacement of 0.90 dollars, and an integer cast or rounding to the dollar leaves every bit intact. The cost is visibility. Invalid values vanish from the column (144 distinct values become 96), and a classifier trained to tell marked from unmarked prices does so with AUC 0.74 (0.5 is chance). On the full `wine` training table in the mixed-table pipeline, marking `price` this way adds $z = 15.1$ (228 distinct values, all valid) and changes grape-variety accuracy by -0.16 points, against -0.15 with the description alone, so the cost is visibility rather than utility. We therefore leave `price` exact in the `wine` experiments and let the description carry the table.

Remark 4. *Modifying only values that occur once in the column leaves the repeated values, and with them the column's histogram, untouched, and the same two-sample test falls to chance (*0.50*). Repeated invalid values then stay invalid, however, so detection weakens (*$z = 3.2$ *on the same column) and less evidence survives deletion.*

B.4 Detection: Multiple Columns, Spoofing, and Keys

From column scores to a table decision. Testing several columns creates several chances for an accidental detection. We set a stricter threshold for each column so that the probability of any false alarm across the table is controlled under the stated assumptions.

Which columns enter the test. The owner fixes which columns are marked before release and lists them in the public descriptor, and the verifier tests exactly those. The choice may depend on the data, for instance on how many distinct values a column has, but not on the key. Choosing columns after seeing their keyed scores would need a further correction.

Only marked columns are tested, and this matters. We treat every value of a categorical column as valid, so such a column always has $T_i = m_i$ and scores $z_i = \sqrt{m_i}$, on marked and unmarked tables alike. Testing it would flag every table. It carries no evidence and is left out, and C below counts only the marked columns.

To keep the chance of any false alarm across the C columns below a level α (we use 0.001), we apply the Bonferroni correction: column i counts as evidence only if its p-value is below α/C, and the table is declared watermarked if any column does.

Tail computation. For a numerical column with T valid items among m distinct values, the p-value is the chance that m fair coins give at least T heads, $\Pr[\mathrm{Bin}(m, 1/2) \geq T]$. The experiment in Appendix D.1 uses its normal approximation $1 - \Phi(z)$, which is accurate at these sizes. Text columns get their p-value from fresh keys instead. The per-column scores in the mixed-table tables are descriptive; Appendix F.1.1 makes the full decision.

Aggregates and their interpretation. The decision always applies Bonferroni to the per-column p-values. Pooling the distinct values of all numerical columns into one count would also give a valid binomial test. On mixed tables we additionally report $z_{\mathrm{comb}} = (z_{\mathrm{num}} + z_{\mathrm{text}})/\sqrt{2}$, a Stouffer summary. Reading it against a standard normal would require the two parts to be independent and standard normal, which the text part is not, so we treat it as a description and report both parts.

Provenance and attack scope. The key and the descriptor fix what is being tested. They do not stop someone from copying marked rows into another table, or from crafting text that happens to be valid. Appendix B.5 states which attacks our analysis covers and which it does not.

B.4.1 Null Calibration

It is tempting to count every cell rather than every distinct value, but that is only safe when values rarely repeat. The hash is deterministic, so every copy of a value carries the same bit: repeats are one coin flip copied, not new flips. Real columns repeat a lot; a duration is often 0 whenever the event did not happen, so one value can fill much of the column and swing the count far from the binomial: over $10{,}000$ clean `Shoppers` tables the cell-count null has standard deviation 22.6 and flags about half of them at the nominal threshold (Table 8). We handle this in three ways. (i) For exact marking we count each distinct value once, which makes the count exactly binomial at the cost of fewer items. (ii) The quantized variant counts cells on purpose, since all cells in one bin share a bit, and takes its threshold from clean tables. (iii) The baseline comparisons score marked tables against paired unmarked ones and report AUC (the chance that a marked table outscores an unmarked one) and the detection rate at a fixed false-alarm rate; these need no formula for the null. All robustness numbers in Appendix J.1 use route (iii). Table 8 validates the exact-predicate choice on $10{,}000$ clean tables: distinct-value counting with column-separated hashing yields a null of -0.01 ± 0.994, whereas uncalibrated cell counting does not.

B.5 Attack Models, Key Management, and Provenance Scope

A detection is evidence that the table was reused. It does not certify that every record is authentic, and it does not rule out every way of faking evidence. This section separates the edits we analyze from stronger attacks, and lists what a deployment needs.

B.5.1 What evidence loss measures

Our survival results give the expected score after random deletion or random replacement. They do not cover an attacker who chooses which values to edit based on their content, or who can query our detector repeatedly. Cheap rounding below the declared precision also removes exact-value evidence across the whole table without hurting utility.

Remark 5 (Quantized numerical evidence)**.** *The quantized variant hashes which logarithmic bin a value falls in, and moves an invalid value to a valid bin while keeping its position inside the bin. A value that stays in its bin keeps its bit; one that crosses into another bin gets a fresh bit. Proposition 5 analyzes a simplified noise model, and Appendix H.3 measures rounding directly. Because invalid bins empty out, a strong classifier can tell quantized releases from unmarked ones.*

Dilution and multiple sources. If m distinct marked items with valid fraction v are mixed with n unmarked ones, the expected score is $2m(v - 1/2)/\sqrt{m+n}$. A positive score shows that marked data is present; it does not say how much, unless one also knows how strong the mark was and how much of it survived. Copying genuine marked rows into another table carries the evidence along, and nothing in the descriptor prevents that.

B.5.2 Key Management and Metadata

Secret and public information. The owner keeps one master secret and derives a separate key for each release with a key-derivation function. Only the key holder can verify. The public descriptor records the release, the marked columns and their precisions; for text it also records the encoder and the published mean embedding (and, for the row-seeded ablation, which row fields seed each hyperplane). The mean embedding may reveal something about the source text and is not a privacy mechanism. Our code takes the derived key as input; how keys are issued and stored is a deployment question, separate from detection.

Binding a claimed release. A deployment can derive $S = \mathsf{KDF}(K, \mathsf{desc})$ from the master secret K and the descriptor, with separately labeled keys for numbers, text directions and candidate selection, serializing the descriptor fields unambiguously. The verifier uses the owner's own copy of the descriptor, never one supplied by a suspect. Renaming or dropping columns can make them unverifiable, and a verifier without a copy of the descriptor cannot verify at all.

Our ablation of hyperplane designs shows a trade-off. The default global hyperplane does not depend on the rest of the row, but an attacker can estimate it from the release (Appendix I.1.1). Per-row hyperplanes defeat that estimate in our tests, but each text cell must stay with its row fields: shuffling the text or deleting a named field breaks verification, and since the fields are public an attacker can also delete chosen groups of rows. Proposition 3 covers none of these targeted attacks.

Limits of the cryptographic assumption. Without the key, any new numerical value is valid with probability $1/2$, so producing one valid value is easy. What is hard is producing many more valid distinct values than chance, without the key and without a detector to query. Our assumption on the hash (Assumption 1) does not by itself rule out copying marked items, learning from repeated detector queries, or crafting valid text. For text, the global hyperplane leaks some information about its direction through the release; per-row hyperplanes leak less when row fields are varied and intact, at the cost described above. Neither design is secure against every attack.

C Theory and Proofs

C.1 Computational Assumption

Table 6 collects the notation used throughout.

Table 6: Notation and definitions used throughout the paper. Where a symbol is reused in a quoted baseline's own notation, the text names the source.

S	secret key	m	distinct evidence items scored
$\mathcal{H}$	keyed hash family	T	count of valid items among m
B	fresh keys for the text null	v	valid fraction, T/m
j	column identifier	z	detection statistic, Eq. (1)
$\langle x \rangle$	canonical form of value x	C	marked columns in a table
c	precision exponent, edits of 10^{-c}	α	significance level
δ	quantization bin width	w	linear width of a bin
h	rounding grid, 10^{-d}	γ	bin-crossing probability
η	failure probability in Thm. 1	ε	tolerance half-width in Thm. 2
q	paraphrase pool size	p_i	exhaustion probability of cell i
κ_i	row fields seeding row i (ablation)	r_j	secret direction of text column j
$\bar{e}$	published reference mean	u_i	centered embedding of item i
θ	angle between embeddings	K	quantization bins

In words: without the key, the hash bits look like independent fair coin flips, even to someone who can ask for the bit of any input they choose. HMAC-SHA-256 is widely assumed to have this property. We state it formally below.

Assumption 1 (PRF Security of the Keyed Hash Function)**.** *Let $\mathcal{H} : \mathcal{S} \times \mathcal{X} \to \{0,1\}$ be a keyed hash family satisfying the pseudorandom function security definition. For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ given oracle access to a function $g(\cdot)$, the advantage of distinguishing $\mathcal{H}(S,\cdot)$ with a secret, uniformly random key $S \xleftarrow{\$} \mathcal{S}$ from a truly random function $f : \mathcal{X} \to \{0,1\}$ is negligible. Formally:*

$$\mathbf{Adv}^{PRF}_{\mathcal{H}}(\mathcal{A}) = \left| \Pr_{S \xleftarrow{\$} \mathcal{S}} \left[\mathcal{A}^{\mathcal{H}(S,\cdot)} = 1 \right] - \Pr_{f \xleftarrow{\$} Func(\mathcal{X},\{0,1\})} \left[\mathcal{A}^{f(\cdot)} = 1 \right] \right| \leq \epsilon \tag{2}$$

Here $\mathcal{A}^{g(\cdot)}$ denotes the adversary $\mathcal{A}$ run with oracle access to the function g and outputting a single bit, $Func(\mathcal{X},\{0,1\})$ is the set of all functions from $\mathcal{X}$ to $\{0,1\}$, and $\mathbf{Adv}^{PRF}_{\mathcal{H}}(\mathcal{A})$ is $\mathcal{A}$'s advantage in distinguishing the two worlds; ϵ is a negligible function of the security parameter (specifically related to $|\mathcal{S}|$).

The assumption is about the hash on numerical values, averaged over a random key. It says nothing about text, where the bit comes from a hyperplane rather than a hash, and it does not guarantee the false-alarm rate of any one fixed key (Appendix D.1.1).

C.2 Semantic Calibration and the Angle Diagnostic

Two nearby vectors usually fall on the same side of a random hyperplane. The identity below makes this exact for vectors chosen without looking at the hyperplane. Released text is chosen by looking at the hyperplane, so we are careful about when the identity applies.

Proposition 4 (Random-hyperplane collision for fixed vectors)**.** *Let u, u' be nonzero vectors fixed independently of a uniform random direction r, with angle $\theta \in [0,\pi]$. Then*

$$\Pr_r[\operatorname{sign}\langle u, r\rangle = \operatorname{sign}\langle u', r\rangle] = 1 - \theta/\pi.$$

Proof. The projection of an isotropic direction onto the plane spanned by the two vectors has a uniform angle. The fraction of directions that separate the vectors is θ/π (Charikar, 2002); the complementary event gives the identity. □

Use after selection. HashMark picks the released paraphrase using r, so a released cell and its later rewrite are not a pair chosen independently of r, and the identity need not hold exactly. If we approximate by assuming every cell, valid or not, keeps its side with probability p, we get

$$\mathbb{E}[z'] \approx 2\sqrt{m}\,[vp + (1-v)(1-p) - \tfrac{1}{2}], \qquad p = 1 - \bar{\theta}/\pi. \tag{3}$$

This is an approximation for reading results; Remark 6 gives the exact version with separate flip rates for valid and invalid cells. We check the approximation against several attackers, and Appendix E.2 reports where it fails badly: when the attacker reselects from the owner's own paraphrases.

We restate the null variance of Section 2.2 and prove it here.

Proposition 1 (Null variance under shared directions)**.** *Let $u_1, \ldots, u_m$ be nonzero vectors fixed independently of the secret (embeddings minus the published reference), let cell i use direction $r_{g(i)}$ for a group label $g(i)$, and let the directions of distinct groups be independent and uniform on the sphere. The count $T = \#\{i : \langle u_i, r_{g(i)}\rangle \geq 0\}$ has $\mathbb{E}[T] = m/2$ and*

$$\operatorname{Var}(T) \;=\; \frac{m}{4} \;+\; \frac{1}{\pi} \sum_{g} \sum_{\substack{i<j \\ g(i)=g(j)=g}} \arcsin\!\left(\frac{\langle u_i, u_j\rangle}{\|u_i\|\,\|u_j\|} \right).$$

Proof. Each bit is positive with probability $1/2$. Bits of cells in different groups are independent. For two cells with $g(i) = g(j)$, $\langle u_i, r\rangle$ and $\langle u_j, r\rangle$ are jointly symmetric with correlation $\rho_{ij} = \cos\angle(u_i, u_j)$, so both are positive with probability $\frac{1}{4} + \arcsin(\rho_{ij})/(2\pi)$ and the covariance of their indicators is $\arcsin(\rho_{ij})/(2\pi)$. Summing over pairs gives the stated variance. □

The proposition explains why text scores cannot be read against a standard normal. With one global hyperplane the sum runs over all $\binom{m}{2}$ pairs of cells. If the arcsines are positive on average, as they are when we do not subtract the mean embedding, $\mathrm{Var}(T)$ grows like m^2, so the spread of z grows like $\sqrt{m}$: the more cells, the worse a fixed threshold becomes. We measure exactly this. On clothing reviews without mean subtraction the spread of z over random keys is 10.55 at $m = 500$, 15.65 at $m = 1000$ and 31.24 at $m = 4000$, within a few percent of $\sqrt{m}$ scaling. Subtracting the mean spreads the embeddings out, the average arcsine becomes slightly negative on this corpus, and the spread falls below 1; this is a property of the corpus, not a general law. With a separate hyperplane per row (our ablation), the sum is empty and $\mathrm{Var}(T) = m/4$ exactly; hyperplanes shared by groups of rows sit in between. The identity closely predicts the simulated null spread: on 500 clothing reviews it gives null standard deviations of 0.725, 2.344, 1.631 and 1.001 for a global hyperplane, class keys, department–class–rating keys and full row seeds, against 0.72, 2.36, 1.64 and 1.00 over 5000 simulated secrets, and 10.99 for the uncentered global hyperplane, compared with 10.55 in the separate 200-direction ablation (Appendices I.1.1 and I.1.2). Because the proposition gives the spread but not the shape of the null, we decide text by comparing the score with its values under fresh keys, which captures the whole tail.

C.3 Quantized and Rewrite Accounting

This section collects three results. The first estimates how often rounding pushes a number into another bin, under a simple random-error model. The second counts how a rewrite gains and loses valid text cells. The third explains why noise-tolerant marking is visible: if values may move by up to ε and still keep their bit, then whole intervals of the number line must be invalid, and a release that avoids them has a histogram with visible gaps. These results describe expected behavior under their assumptions, not guarantees against every attack.

Proposition 5 (Quantized survival under rounding)**.** *Consider a bin of linear width w and a released value with uniform offset within that bin. Assume an independent rounding-error surrogate $E \sim \mathrm{Unif}[-h/2, h/2]$, or truncation-error surrogate $E \sim \mathrm{Unif}[-h, 0]$, where $h = 10^{-d}$. If $|E| \le w$ almost surely, the bin-crossing probability is*

$$\gamma = \frac{\mathbb{E}|E|}{w} = \begin{cases} h/(4w) & \textit{rounding surrogate}, \\ h/(2w) & \textit{truncation surrogate}. \end{cases}$$

For a release with cell-level valid fraction v, let $\bar{\gamma}_{\mathrm{valid}}$ be the average crossing probability over initially valid cells. Then $\mathbb{E}[v'] \ge v(1 - \bar{\gamma}_{\mathrm{valid}})$ for the cell-count detector.

Remark 6 (Post-selection rewrite accounting)**.** *For a fixed release of valid fraction v, let a be the mean probability that an initially valid cell becomes invalid and b the mean probability that an initially invalid cell becomes valid under a specified randomized attacker. Then*

$$\mathbb{E}[z'] = 2\sqrt{m}\,[v(1-a) + (1-v)b - \tfrac{1}{2}].$$

In words: valid cells that stay valid, plus invalid cells that become valid, give the new count. The formula predicts the expected score only.

Setting $a = b = \bar{\theta}/\pi$ recovers the angle-model approximation; the conditional flip rates it requires are measured in Appendix E.2.

Theorem 2 (Histogram separation for value-only predicates)**.** *Let I be an interval of length L, and let a blind predicate have valid set V that is balanced on I, $|V \cap I| = L/2$, and whose connected components not meeting the endpoints of I each have length at least 2ε, as for quantized hashing with valid bins of width 2ε, where a value at a bin center tolerates perturbations of size ε. Let ρ be a clean distribution with density at least d on I, and ρ' a release that places no mass outside V. Then the histograms ρ_h, ρ'_h with bins of width h satisfy*

$$\mathrm{TV}(\rho_h, \rho'_h) \;\ge\; d\left[\frac{L}{2}\left(1 - \frac{h}{\varepsilon}\right) - 6h\right].$$

(Proof deferred to Appendix C.4.)

C.4 Deferred Proofs

The proofs below justify the bounds used in the construction. Numerical distortion is measured relative to the declared grid; text capacity depends on fixed candidate geometry; and the deletion result assumes uniformly sampled evidence items.

Theorem 1 (Numerical Fidelity)**.** *In the ideal random-function (random-oracle) model, let* $\mathbf{X}$ *be a fixed dataset of* N *numerical values on the declared precision grid and* $\mathbf{X}_w$ *its Edit Mode release. Assume each search visits distinct canonical values and can continue until acceptance; repeated inputs are left unchanged. The expected* L_∞ *distortion is bounded by:*

$$\mathbb{E}[||\mathbf{X} - \mathbf{X}_w||_\infty] \le (\log_2 N + 2) \cdot 10^{-c}.$$

Moreover, for every $\eta \in (0, 1)$*, with probability at least* $1 - \eta$*,*

$$||\mathbf{X} - \mathbf{X}_w||_\infty \le \lceil \log_2(N/\eta) \rceil \cdot 10^{-c},$$

and the distribution of values in every column (its empirical marginal F*) satisfies* $\mathbb{E}[W_1(F_{\mathbf{X}}, F_{\mathbf{X}_w})] \le 10^{-c}$ *and* $W_\infty(F_{\mathbf{X}}, F_{\mathbf{X}_w}) \le ||\mathbf{X} - \mathbf{X}_w||_\infty$.

Proof. For each element x_i in $\mathbf{X}$, let x_i' be the corresponding element in $\mathbf{X}_w$. As defined above:

$$x_i' = x_i + k_i \cdot 10^{-c},$$

For singleton inputs, $k_i = \min\{k \ge 0 \mid \mathcal{H}(S, x_i + k \cdot 10^{-c}) = 0\}$; for unchanged repeated inputs, $k_i = 0$. In other words, $|x_i - x_i'| = k_i \cdot 10^{-c}$.

Work in the ideal random-function model of the theorem. For a singleton search, distinct queried values have independent fair bits. Therefore, for a given $x_i' = x_i + k_i \cdot 10^{-c}$, the hash function should have mapped to 1 for every choice from 0 to $k_i - 1$ and succeed in time k_i. In other words, $\Pr[K_i = k] = \left(\frac{1}{2}\right)^{k+1}$, i.e., it follows a geometric distribution.

Now, $||\mathbf{X} - \mathbf{X}_w||_\infty = \max_i |x_i - x_i'| = \max_i k_i \cdot 10^{-c}$. Each K_i satisfies $\Pr[K_i \ge k] \le 2^{-k}$, so for any integer $t \ge 0$ a union bound over the N entries gives $\Pr[\max_i K_i \ge k] \le \min\{1,\, N2^{-k}\}$ and hence

$$\mathbb{E}[\max_i K_i] \;=\; \sum_{k \ge 1} \Pr[\max_i K_i \ge k] \;\le\; t \;+\; N \sum_{k > t} 2^{-k} \;=\; t + N2^{-t}.$$

Taking $t = \lceil \log_2 N \rceil$ makes $N2^{-t} \le 1$ and $t \le \log_2 N + 1$, so $\mathbb{E}[\max_i K_i] \le \log_2 N + 2$. This gives us that:

$$\mathbb{E}[||\mathbf{X} - \mathbf{X}_w||_\infty] \le (\log_2 N + 2) \cdot 10^{-c}.$$

For the high-probability bound, all entries satisfy $\Pr[k_i \ge t] \le 2^{-t}$. A union bound over the N cells gives $\Pr[\max_i k_i \ge t] \le N\, 2^{-t}$, which is at most η for $t = \lceil \log_2(N/\eta) \rceil$. Pairing each value with its own watermarked copy is a coupling of the two empirical marginals, so $W_1 \le \frac{1}{N} \sum_i |x_i - x_i'| = \frac{10^{-c}}{N} \sum_i k_i$, whose expectation is at most 10^{-c}, and $W_\infty \le \max_i |x_i - x_i'|$.

□

Accounting identity in Remark 6. Linearity of expectation gives $mv(1 - a)$ initially valid survivors and $m(1 - v)b$ newly valid items. Their sum, substituted into Eq. (1), yields the stated identity without requiring independent attack outcomes across rows. □

Proposition 2 (Capacity from pool geometry)**.** *Let a cell's candidates, the original and* q *paraphrases (*$q + 1$ *in total in this proposition and the geometry tables), have nonzero centered embeddings* $u_0, \ldots, u_q$*, let* U *be the matrix whose rows are these embeddings, and let the direction* r *be a Gaussian vector (which points in a uniformly random direction), independent of the pool. Define* exhaustion *as the event that every candidate lands on the invalid side. Its probability is the orthant probability that all* $q + 1$ *coordinates of a correlated Gaussian vector are negative:*

$$p_i \;=\; \Pr\big[\, Ur < 0 \,\big], \qquad Ur \sim \mathcal{N}\big(0,\, UU^\top\big),$$

which depends on the pool only through the angles between its candidates: $p_i = \frac{1}{2}$ for identical candidates and $p_i = 2^{-(q+1)}$ for orthogonal ones. Assume selection leaves m distinct evidence items, so deduplication does not change the denominator. Then the expected valid fraction is $\mathbb{E}[v] = 1 - \frac{1}{m}\sum_i p_i$, and Eq. (1) gives its expected detection statistic.

Proof. Write $U \in \mathbb{R}^{(q+1)\times d}$ for the matrix whose rows are the centered candidate embeddings $u_0, \ldots, u_q$, and let r be the cell's direction. A candidate u is valid when $\langle u, r\rangle \geq 0$, and the owner releases a valid candidate whenever the pool contains one, so the cell stays unmarked exactly on the event that no candidate is valid,

$$\{\text{cell unmarked}\} \;=\; \bigcap_{j=0}^{q} \{\langle u_j, r\rangle < 0\} \;=\; \{Ur < 0\},$$

the inequality holding coordinatewise. For $r \sim \mathcal{N}(0, I_d)$ independent of the pool, Ur is a centered Gaussian vector with covariance $\mathbb{E}[Urr^\top U^\top] = UU^\top$, so $p_i = \Pr[Ug < 0]$ with $Ug \sim \mathcal{N}(0, UU^\top)$: the orthant probability of the candidate Gram matrix. Since r is isotropic, the same holds for any r uniform on the sphere, the sign event being invariant to the length of r.

The event $\{Ur < 0\}$ is unchanged when each row u_j is scaled by a positive constant, so p_i depends on $UU^\top$ only through its correlation matrix, whose entries are the cosines $\langle u_j, u_k\rangle/(\|u_j\|\|u_k\|)$; that is, only through the angles between candidates. Two reference configurations illustrate it. If all candidates coincide in direction, the coordinates of Ur are almost surely of one sign, the intersection is a single half-space, and $p_i = \frac{1}{2}$: a pool of identical paraphrases is worth one candidate. If the candidates are mutually orthogonal, $UU^\top$ is diagonal, the coordinates are independent, each negative with probability $\frac{1}{2}$, and $p_i = 2^{-(q+1)}$, the independence prediction. These are not universal extrema: two antipodal candidates have zero exhaustion. The measured paraphrase pools are strongly positively correlated, which explains why the independent-candidate approximation overstates their capacity.

A cell contributes a valid item unless it stays unmarked, so a column of m distinct such evidence items has expected valid fraction $v = 1 - \frac{1}{m}\sum_i p_i$, and substituting this v into Eq. (1) gives the expected statistic. □

Remark 7 (Scope of the independence hypothesis)**.** *The hypothesis that the direction is independent of the pool holds at embedding time by construction: the paraphraser proposes candidates without access to S, so the pool is fixed before any hyperplane is applied to it. It does not describe an adversary who resamples a pool* after *observing which candidate was released, whose alternatives are conditioned on the same hyperplane; Appendix E.2 measures that conditioning, and Remark 6 therefore retains separate conditional flip probabilities.*

Proposition 3 (Uniform subsetting of evidence)**.** *Fix a release with m distinct evidence items, $T = mv$ of them valid, and a descriptor. Retain a uniformly sampled subset of m' items without replacement, leaving their predicates unchanged. Then $T' \sim \mathrm{Hypergeom}(m, T, m')$ and*

$$\mathbb{E}[z'] = 2\sqrt{m'}(v - \tfrac{1}{2}).$$

The exact detection probability at a threshold z_{th} is the corresponding hypergeometric upper tail.

Proof. A uniform subset of m' items from a population with T valid and $m - T$ invalid items has the displayed hypergeometric count. Its mean is $m'T/m = m'v$ and, for $m > 1$, its variance is $m'v(1-v)(m-m')/(m-1)$. Substituting its mean into Eq. (1) gives the expectation. With $t_* = \lfloor m'/2 + z_{\mathrm{th}}\sqrt{m'}/2\rfloor + 1$, the event $z' > z_{\mathrm{th}}$ is exactly $T' \geq t_*$, so $\Pr[z' > z_{\mathrm{th}}] = \sum_{t\geq t_*} \binom{T}{t}\binom{m-T}{m'-t}\big/\binom{m}{m'}$. No claim about arbitrary content-adaptive deletion follows from this sampling calculation. □

Proposition 5 (Quantized survival under rounding)**.** *Consider a bin of linear width w and a released value with uniform offset within that bin. Assume an independent rounding-error surrogate $E \sim \mathrm{Unif}[-h/2, h/2]$, or truncation-error surrogate $E \sim \mathrm{Unif}[-h, 0]$, where $h = 10^{-d}$. If $|E| \leq w$ almost surely, the bin-crossing probability is*

$$\gamma = \frac{\mathbb{E}|E|}{w} = \begin{cases} h/(4w) & \textit{rounding surrogate}, \\ h/(2w) & \textit{truncation surrogate}. \end{cases}$$

For a release with cell-level valid fraction v, let $\bar{\gamma}_{\text{valid}}$ be the average crossing probability over initially valid cells. Then $\mathbb{E}[v'] \geq v(1 - \bar{\gamma}_{\text{valid}})$ for the cell-count detector.

Proof. A value at offset u from the lower edge of a bin of width w, with u uniform on $[0, w)$, leaves the bin under a shift e exactly when $u + e \notin [0, w)$. For $|e| \leq w$ this has probability $|e|/w$ over u conditional on the independent shift, and averaging over the error gives $\mathbb{E}|e|/w$, with $\mathbb{E}|e| = h/4$ for rounding and $h/2$ for truncation; the probability is capped at 1. The condition on h guarantees $|e| \leq w$, so the probability is exact. A value that stays in its bin keeps its bin index and therefore its bit, so every valid value survives as valid with probability $1 - \gamma(x)$; values that leave may land valid or invalid, which can only add to the count. Averaging the retained initially valid cells gives $\mathbb{E}[v'] \geq v(1 - \bar{\gamma}_{\text{valid}})$, the stated bound for the cell-count detector. □

Theorem 2 (Histogram separation for value-only predicates)**.** *Let I be an interval of length L, and let a blind predicate have valid set V that is balanced on I, $|V \cap I| = L/2$, and whose connected components not meeting the endpoints of I each have length at least 2ε, as for quantized hashing with valid bins of width 2ε, where a value at a bin center tolerates perturbations of size ε. Let ρ be a clean distribution with density at least d on I, and ρ' a release that places no mass outside V. Then the histograms ρ_h, ρ'_h with bins of width h satisfy*

$$\mathrm{TV}(\rho_h, \rho'_h) \;\geq\; d\left[\frac{L}{2}\left(1 - \frac{h}{\varepsilon}\right) - 6h\right].$$

Proof. At most two components of $V \cap I$ meet the endpoints of I; every other component has length at least 2ε, and together they have length at most $L/2$, so there are at most $N \leq L/(4\varepsilon) + 2$ components in total. The complement $V^c \cap I$ is a union of at most $N + 1$ intervals $g_1, \ldots, g_{N+1}$ of total length $L/2$. On a grid of bins of width h, an interval of length $|g_j|$ contains fully contained bins covering length at least $|g_j| - 2h$. The total length of bins lying entirely inside V^c is therefore at least

$$\sum_j \big(|g_j| - 2h\big) \;\geq\; \frac{L}{2} - 2h(N+1) \;\geq\; \frac{L}{2}\left(1 - \frac{h}{\varepsilon}\right) - 6h.$$

Let B be the set of those bins. The release places no mass on V^c, so $\rho'_h(B) = 0$, while $\rho_h(B) \geq d\left[\frac{L}{2}(1 - h/\varepsilon) - 6h\right]$ because ρ has density at least d on I. Since both histograms have total mass one, $\sum_{b \notin B} |\rho_h(b) - \rho'_h(b)| \geq \left|\sum_{b \notin B}(\rho_h(b) - \rho'_h(b))\right| = \rho_h(B)$, and hence

$$\mathrm{TV}(\rho_h, \rho'_h) = \tfrac{1}{2}\sum_b |\rho_h(b) - \rho'_h(b)| \;\geq\; \tfrac{1}{2}\big(\rho_h(B) + \rho_h(B)\big) = \rho_h(B),$$

which is the stated bound. □

D Calibration and the Ownership Decision

D.1 Null Calibration over 10,000 Tables

This section asks how often an unwatermarked table triggers the detector. We first average over many randomly chosen keys, then check whether the one key used by an owner behaves differently.

Summary. We first score tables that were never marked; the spread of these scores is the *null* against which a suspect table is judged. We find that counting each distinct value once (instead of each valid cell) gives a well-calibrated null over 10,000 clean tables, while counting cells flags half of them as watermarked (Table 7). For text, every cell of a column shares one hyperplane, so the bits are not independent. Without subtracting the mean embedding, the chance-score standard deviation on 500 reviews is 10.55 rather than the standard-normal reference of 1, with 155 of 200 unmarked columns scoring beyond ± 3.29 (which a standard normal exceeds one time in a thousand), and keeps growing with the column, to 31.24 at 4,000 reviews (Appendix I.1.1). Centering removes most of this, but not all: on the 17,757 Airbnb summaries the null standard deviation is 3.0 on the clean column and 7.2 on the release. Proposition 1 predicts these spreads from each column's own embeddings (on Wine, 4.37 predicted and 4.38 measured). Since the variance does not fix the shape of the tail, we decide text columns by comparing the released score with scores under many fresh keys (Appendix F.1.1).

A test can perform well when averaged over many keys yet produce more false alarms for a particular owner's key. We therefore also test performance with each key held fixed. Holding each of 50 keys fixed over 2,000 clean tables, the pooled spread is 0.997, of which 0.919 comes from tables and 0.387 from the key itself, so false-positive rates range from 0 to 1.20% across keys. An owner should therefore calibrate the deployed key on held-out clean data (Appendix D.1.1).

Table 7: Scores on clean tables. Numbers: 500-row `Shoppers` tables, with the hash ignoring or including the column name. Text: 500 reviews.

Counting	Null SD	$z > 3.09$
Cells	22.6	$\approx 50\%$
Distinct, value only	1.146	35/10,000
Distinct, per column	0.994	13/10,000
Text, uncentered	10.55	–
Text, centered	0.71	–
Text, row-seeded (ablation)	1.00	–

We draw 10,000 unwatermarked 500-row tables from a pool of synthetic `Shoppers` rows generated by TabSyn (Zhang et al., 2024), a diffusion-based table generator, each with a fresh key and row sample, and score each exactly as the detector does. Hashing a value alone gives the same bit to a value shared across columns, correlating their counts; hashing the value together with its column removes this.

Table 8: Detector scores on 10,000 clean tables, with the false-positive rate at $z > 3.09$ (nominally 0.1%), an exact 95% upper confidence bound on that rate, and the score that 0.1% of clean tables actually exceed.

Hash input	Null SD	FPR at $z > 3.09$	95% upper	Empirical threshold
Value only	1.146	$35/10{,}000 = 0.35\%$	0.46%	3.62
Column ‖ value (HashMark)	**0.994**	$13/10{,}000 = 0.13\%$	0.21%	**3.13**

When the column name is hashed with the value, the score exceeded by 0.1% of clean tables is 3.13, close to the 3.09 a standard normal predicts, and at $z > 3.29$ the observed one-sided exceedance is $5/10{,}000 = 0.05\%$. Cell counting on the same tables gives standard deviation 22.6 and flags 50.1% of them.

The family-wise decision, measured. We evaluate the numerical table-level decision on 500 paired trials (Table 9). Each trial draws a clean 500-row Shoppers table from the synthesizer pool, embeds a copy under a fresh key, and rejects when $\min_j p_j < \alpha/C$, with C taken from that trial's descriptor. The p-values use the normal approximation. Which columns are marked depends on the sampled table; our result file stores only the last trial's count ($C = 6$).

Table 9: Numerical family-wise rejection rates under Bonferroni correction over eligible columns, using upper-tail normal probabilities; 500 paired trials with a fresh key each, `Shoppers`, 500 rows. The pooled single-statistic decision over the same columns is shown for comparison. Every watermarked release is detected at every level.

	Bonferroni (primary)		pooled statistic	
α	TPR	family-wise FPR	TPR	FPR
0.001	1.000	0.000	1.000	0.000
0.01	1.000	0.006	1.000	0.010
0.05	1.000	0.054	1.000	0.052

The observed clean rejection counts are 0/500, 3/500, and 27/500. Zero rejections in 500 trials bounds the false-alarm rate at level 0.001 by 0.60% with 95% confidence; Table 8 resolves the tail more finely with 10,000 tables. All 500 marked releases are detected at each level. The experi-

ment uses numerical Shoppers columns; mixed-table component scores are reported separately in Appendix F.1.

D.1.1 Calibration Conditional on the Deployed Key

Table 8 averages over keys: each of the $10{,}000$ tables is scored under a fresh secret, which is what the PRF assumption delivers and what Section 1.3 claims. A deployed watermark, however, uses one key for its whole life, and conditional on that key the valid region has some fixed mass p_k over the column's value distribution. Only the average of p_k over keys is $\frac{1}{2}$, so the score carries a per-key offset

$$\mathbb{E}[z \mid \text{key}] \;=\; 2\sqrt{m}\,\big(p_k - \tfrac{1}{2}\big),$$

which does *not* shrink as more clean tables are scored: it is a property of the key, not sampling noise. At the $m = 1440$ distinct items of a 500-row `Shoppers` table, a mass imbalance of 0.01 already moves the mean by 0.76.

We measure it directly. We fix 50 keys and score $2{,}000$ fresh clean tables under each, reusing the same $2{,}000$ table samples across all keys so that differences are attributable to the key alone.

Table 10: Fixed-key calibration for 50 keys evaluated on $2{,}000$ clean 500-row `Shoppers` tables with shared table samples. The table reports pooled and key-conditional behavior at the nominal 0.1% rate.

Quantity	Value	Nominal
Pooled null SD (all 100,000 scores)	0.997	1.000
Pooled FPR at $z > 3.09$	0.137%	0.1%
Within-key null SD (mean over keys)	0.919	1.000
SD of the 50 per-key means	0.387	0
Range of per-key means	$[-1.32, +0.99]$	0
Per-key FPR at $z > 3.09$, median	0.08%	0.1%
Per-key FPR, maximum	$\mathbf{1.20}\%$	0.1%
Keys with FPR above 0.5%	$3/50$	N/A
Keys with no false positive at all	$19/50$	N/A
Per-key 99.9th percentile, range	$[1.51, 3.79]$	3.09

The decomposition in Table 10 is exact to the second decimal: $0.919^2 + 0.387^2 = 0.994$ against a pooled variance of $0.997^2 = 0.994$. So the score of a clean table under a random key is the sum of two parts: a per-key offset with spread 0.387, and table-to-table variation with spread 0.919, a little under 1. Averaged over keys the two add back up to spread 1, as in Table 8, which hides the per-key offset. The offset comes from a tiny imbalance: the share of valid values under one key differs from $\frac{1}{2}$ by 0.003 in the median and 0.017 at most. That is harmless for security but large enough to matter for a test over $1{,}440$ values.

The practical consequence is a wide spread in false-positive rate across keys at a threshold derived from the pooled null: 19 of 50 keys produce no false positive in $2{,}000$ clean tables, while the worst produces 24, and the threshold each key would actually need for a 0.1% rate ranges from 1.51 to 3.79. A threshold set by averaging over keys is thus stricter than needed for most keys and too lax for a few, and averaging over keys cannot tell which is which.

The owner, however, can measure it before release. The owner holds both the key and clean data, so the calibration above is a procedure they can run before release: score held-out clean tables under the deployed key and take an empirical upper quantile. With $2{,}000$ tables only about two scores lie beyond the 99.9th percentile, so the owner should report this estimate together with its confidence interval. The $2{,}000$ detector evaluations take four seconds in our implementation. We recommend this whenever a key will be used for a long time: the formula for the null describes a random key, not the key in use. The same holds for text, whose null we also estimate by averaging over keys.

E CAPACITY AND INSTALLATION COST

E.1 POOL GEOMETRY PREDICTS CAPACITY

A pool is the original text plus its candidate paraphrases. A pool is *exhausted* when none satisfies the marking rule. The question here is whether the candidates' similarity predicts how often this happens.

For every cell we compute the prediction of Proposition 2, the chance that a random hyperplane puts all $q + 1$ candidates on the invalid side, by drawing 4000 random directions. We then measure how often the pool is actually exhausted under 100 keyed hyperplanes per cell; a cell's chance of exhaustion does not depend on whether other cells share its hyperplane, so these rates hold for the default global hyperplane. The column *independent* is the prediction $2^{-(q+1)}$ that treats candidates as independent coin flips. *Effective rank* measures how spread out the candidates are: it is about 1 when all paraphrases point the same way and grows toward $q + 1$ as they spread out.

Table 11: Predicted and measured candidate-pool exhaustion and detection scores for four corpora at 500 cells and clothing reviews at 2000 cells. Predictions use pool geometry; measurements use keyed hyperplanes.

		Exhaustion			z		Effective
Corpus	q	predicted	measured	independent	predicted	measured	rank
Clothing reviews	1	0.354	0.353	0.250	6.51	6.58	1.47
	4	0.208	0.205	0.031	13.04	13.18	2.04
	8	0.143	0.141	0.002	15.96	16.04	2.31
	16	0.093	0.092	0.000	18.20	18.26	2.50
	32	0.058	0.058	0.000	19.79	19.76	2.61
Clothing reviews, 2000 cells	2	0.295	0.296	0.125	18.31	18.29	1.65
	8	0.148	0.149	0.002	31.49	31.42	2.24
	32	0.058	0.059	0.000	39.56	39.47	2.60
Yelp reviews	1	0.372	0.369	0.250	5.74	5.85	1.37
	4	0.242	0.241	0.031	11.56	11.59	1.77
	8	0.177	0.177	0.002	14.45	14.46	1.97
	16	0.125	0.124	0.000	16.78	16.81	2.11
	32	0.084	0.083	0.000	18.58	18.63	2.19
IMDB reviews	1	0.359	0.359	0.250	6.33	6.29	1.45
	4	0.212	0.212	0.031	12.86	12.88	2.04
	8	0.146	0.147	0.002	15.81	15.80	2.35
	16	0.097	0.097	0.000	18.02	18.00	2.56
	32	0.062	0.062	0.000	19.60	19.58	2.70
AG News	1	0.406	0.405	0.250	4.22	4.25	1.21
	4	0.306	0.303	0.031	8.69	8.83	1.38
	8	0.256	0.253	0.002	10.93	11.06	1.44
	16	0.208	0.207	0.000	13.05	13.09	1.49
	32	0.168	0.168	0.000	14.87	14.87	1.51

Predicted and measured exhaustion agree to within 0.0032 at every pool size on all four corpora, and at $q = 32$ per-cell predictions correlate with per-cell exhaustion at 0.82 to 0.91, falling to 0.48 at the smallest pools. At four times the scale the law is unchanged: on 2000 freshly generated review pools predicted exhaustion is within 0.0016 of measured at every size, the effective rank is again 2.6, and z doubles to 39.5 as $\sqrt{m}$ requires. The independence prediction is wrong by an order of magnitude from $q = 4$ on. Even 33 candidates have effective rank only 1.5 to 2.7, lowest for news. More paraphrases therefore buy capacity slowly, and unevenly across cells, so the per-cell prediction is the right guide to where extra paraphrases help.

E.2 Transfer, and Where the Angle Model Stops

We test two different predictions: how much evidence can be installed, and how much survives a rewrite; Table 12 reports both. The first transfers across the tested settings; the second fails when the attacker can choose from alternatives the owner generated but did not release.

The capacity law is a statement about pool geometry, so it should transfer to corpora and encoders that played no part in developing it. We predict exhaustion from geometry alone and measure it over 10 fresh secrets, on a second encoder (`all-mpnet-base-v2`) and two corpora.

Table 12: Predicted and measured exhaustion across encoders and paraphrasers, and detection after owner-pool reselection. Left: exhaustion and implied capacity z with no fitted parameters. Right: angle-model predictions and measurements after every released cell is replaced by another candidate from its owner pool.

		capacity				after a pool rewrite		
Corpus	Encoder	exh. pred.	exh. meas.	z pred.	z meas.	selection	z pred.	z meas.
Clothing reviews	MiniLM	0.058	0.062	19.79	19.61	first-valid	8.11	0.06
						margin	7.89	-0.04
AG News	mpnet	0.178	0.174	14.40	14.60	first-valid	9.32	-0.25
						margin	9.17	0.05
IMDB reviews	mpnet	0.069	0.065	19.27	19.44	first-valid	8.35	-0.42
						margin	8.09	-0.37
Clothing reviews	MiniLM, BART	0.1296	0.1302	16.57	16.54	first-valid	10.11	-0.16
						margin	9.71	-0.42

Capacity transfers: on the mpnet rows, which use an encoder the law never saw, predicted and measured exhaustion agree to within 0.0045, and the implied z is within 0.2 of measurement. Changing the *paraphraser* changes capacity, and the law tracks that too. Replacing the T5 paraphraser with BART on the same 500 cells produces candidates that stay closer to their source (mean cosine 0.891 against 0.741) and are correspondingly less diverse (1.67 effective rank of 33 against 2.61), so exhaustion rises from 0.058 to 0.130 at the same budget; geometry predicts that shift to within 0.001. A more conservative paraphraser therefore trades capacity for fidelity, at a rate the geometry predicts once the pools exist. The right-hand columns are a negative result. The angle model reads the mean angle between a released cell and its rewrite (0.18π to 0.29π here) and predicts $z \approx 8$–10; measurement gives $z \approx 0$. The gap is not caused by margin selection, since first-valid selection fails identically. It is caused by *where the rewrite comes from*: the other candidates in a cell's pool were available to the owner and not chosen, so they are conditioned on the same hyperplane, and swapping to one is not the random move the model assumes. The conditioning is direct: on clothing reviews the valid fraction across all candidates is 0.497, as the null requires, but among the candidates a cell did *not* release it is 0.306, with mean signed margin -0.017 against $+0.043$ for the released one, so an adversary drawing from the leftovers lands invalid about seven times in ten. Remark 6 accounts for this by using flip rates measured after selection. We therefore base every robustness claim on measured attacks (Appendix I.2), not on the model.

F End-to-End Mixed Tables

F.1 Hybrid Tables: Numbers, Categories, and Text

Here we mark whole mixed tables, numbers, and text together under one key. We report per-column scores that describe the evidence separately from the calibrated decision (Appendix F.1.1), and we measure prediction accuracy separately from detection.

We mark mixed tables from the multimodal AutoML benchmark (Shi et al., 2021) end-to-end with one key. Continuous columns are marked exactly on their declared grid; categorical, integer, and boolean columns are left unchanged (Appendix B.3 gives the rule for integer columns, which turns on the measurement scale rather than the storage type), and the free-text column is marked by choosing among 8 paraphrases per cell under the global hyperplane (Table 14 compares row-seeded

hyperplanes derived from the listing's structured fields). We report detection per column and combine across columns, against the clean training table and a wrong key. Downstream, an XGBoost model on the tabular features plus sentence embeddings of the text is trained on the clean or the watermarked training table and scored on the untouched test split, paired over 20 seeds; free text enters only through its embedding, and columns from which the target is derived are removed.

Table 13: Hybrid watermark on mixed tables, Melbourne Airbnb (18,316 training listings, ten-class price) and AutoML California listings (37,951 training sales, regression). m is the number of distinct values (numbers) or cells (text) scored. Accuracy change is watermarked minus clean, in points, with a 95% interval over paired seeds. Both mixed tables are marked at the declared precision $c = 6$, with the storage precision $c = 12$ shown beneath each for contrast. Rows labeled $c = 12$ use the storage precision; they score higher but do not survive rounding below 12 decimals. *Text columns use the global hyperplane, whose null is wider than a standard normal; each is read against its empirical null over fresh keys (Table 14), where the three marked columns lie 11.5, 10.0 and 12.5 null standard deviations out, and the wrong-key scores (8.13 and 24.29) lie only 1.1 and 2.0 out, below any calibrated threshold. `Wine` text is marked on a 4,000-row sample.

Table	Column (predicate)	m	z marked	z clean	z wrong key
Melbourne Airbnb	latitude (exact, c=6)	15,383	114.3	−1.82	
	longitude (exact, c=6)	15,643	116.5	−0.29	
	reviews per month	left exact: measured and stored to two decimals			
	numbers combined	31,026	**163.2**	−1.49	−0.67
	storage precision c=12	36,495	190.2	−0.56	−0.87
	summary (global hyperplane)*	17,757	**83.1**	−3.41	8.13
AutoML California listings	tax assessed value (exact, c=6)	23,442	124.0	0.07	
	annual tax amount (exact, c=6)	17,076	63.8	1.44	
	lot size (exact, c=6)	7,260	43.6	−1.29	
	numbers combined	47,778	**141.9**	0.40	0.79
	storage precision c=12	53,014	163.1	1.47	−1.40
	summary (global hyperplane)*	37,663	**118.6**	0.55	24.29
Wine reviews	price	left exact: whole dollars, unit-grid marking is visible			
	description (global hyperplane)*	4,000	**34.2**	0.13	2.37

Each table is measured under a single key, named in the released scripts. Scores move slightly with the key (a second secret marks the same Airbnb release at 163.4 rather than 163.2, over 31,063 distinct items rather than 31,026). On Airbnb, the two coordinate columns are nearly unique and carry the numerical evidence, each at $z > 110$ with edits no larger than $1.3 \cdot 10^{-5}$, about 1.5 m; `reviews_per_month` is measured and stored to two decimals, so no declarable grid hides an edit and it is left exact. Together, the two coordinates give $z = 163.2$. The summary column, marked under the global hyperplane, reaches $z = 83.1$: about half of the cells were already valid, about a third were rewritten, and the rest exhausted their 8 paraphrases, consistent with Proposition 2. Its null over 200 fresh keys on the released column has standard deviation 7.2 (3.0 on the clean column), so the column is read against that empirical null, 11.5 null standard deviations out. The ten-class price accuracy changes by -0.18 points over 20 paired seeds, with 95% interval $[-0.99, +0.64]$, which contains zero. On AutoML California listings, the three tax and lot columns carry the numerical evidence at $z = 141.9$ with edits below $1.2 \cdot 10^{-5}$, and the summary column adds $z = 118.6$ against a release null of standard deviation 11.9, 10.0 null standard deviations out. Its three school-distance columns are stored to one decimal, so the smallest valid step of 0.1 would move values by up to 0.8 while adding only a few hundred distinct values; their measurement and storage precision coincide, so no declarable grid hides an edit (Section 1.3) and the construction leaves them exact. Regression R^2 is 0.9362 clean and 0.9365 watermarked, a change inside ± 0.001. In both tables, eligible numerical columns and the text column contribute evidence under one key; categorical and coarse-precision columns remain unchanged.

The default global hyperplane detects on both tables at 10–12 null standard deviations and survives any table reshuffling. The row-seeded ablation separates marked from clean releases 7 to 14 times more strongly once each is standardized by its own null, but needs every text cell to stay

Table 14: Text evidence under the two hyperplane designs, same releases and paraphrase pools. The *global* hyperplane is one secret hyperplane per text column, shared by every cell; *row-seeded* hyperplanes are derived from the key and each row's context fields. Null SD is the spread of z over 200 fresh keys on the released column; z/SD is z divided by that spread, a scale for comparison and not a p-value. Reselection draws each text cell from the owner's cached candidate pool, excluding the released candidate. The global hyperplane survives any change to the pairing of text and row, but its null widens with the column and a wrong key scores well above 3.09; row-seeded hyperplanes keep the null near unit spread but fail if the text column is permuted or a context column is dropped.

Table	Hyperplane	z	Null SD	z/SD	Wrong key	Reselection z
`Airbnb`	global	83.1	7.23	11.5	8.13	-10.3
	row-seeded	87.9	1.11	79.1	-0.40	-8.7
`California`	global	118.6	11.85	10.0	24.29	-17.0
	row-seeded	135.7	0.99	137.0	2.09	-9.8

in its row. The global null is wide because every text cell shares one hyperplane, so the positive average pairwise arcsine between centered embeddings accumulates over m cells, as Proposition 1 describes. Selection changes the geometry of the released column, and with it the fresh-key null: 7.2 on the released Airbnb column against 3.0 on the clean one. On the 4,000-row family-wise releases, Proposition 1 applied to each column's own embeddings matches the measured spread (2.87 predicted against 2.86 measured on the `Airbnb` release; Appendix F.1.1); since variance alone does not fix the tail, decisions use the empirical null on the suspect column.

Wine reviews test two things Airbnb and California cannot. With the default global hyperplane, a 4,000-row sample reaches $z = 34.2$ against a release null of standard deviation 2.7 and changes grape-variety accuracy by -0.15 points (95% interval $[-0.33, +0.02]$). Its only numeric column, whole-dollar price, is left exact because marking it on the unit grid is visible (Appendix B.3), so the watermark lives entirely in the description, and the label, the grape variety, is predicted mainly from that description. The rest of this paragraph uses the row-seeded ablation on all 84,123 descriptions, which stresses both calibration and utility. First, calibration: the row fields that seed the hyperplanes (country, province, points) take only 2,296 distinct values across 84,123 rows, so 37 cells share a hyperplane on average, and the null widens as Proposition 1 predicts. Over 20 fresh secrets, a clean column has null mean 1.67, standard deviation 4.56, and maximum 9.83: the nominal 3.29 threshold is invalid here, and the observed clean score of 4.29 is an ordinary draw from that null rather than a false positive. The widening is not a surprise after the fact: Proposition 1 predicts it from the key structure alone, giving 4.38 against the 4.56 measured, and drawing fresh directions per key group gives 4.24. Detection survives the correction, at $z = 181.3$. Second, utility: marking 30.5% of the descriptions costs 0.86 accuracy points (75.96% to 75.10%), with 95% interval $[-1.01, -0.71]$, negative in every seed. The cost of watermarking text scales with how much the task depends on the text it rewrites, and it is measurable on a text-dominant task.

F.1.1 The Declared Decision, End to End

The tables above report per-column scores, which are descriptive. This section makes the decision the construction declares: a p-value for every marked column, a Bonferroni correction across the columns tested, and one verdict per table. It uses the shipped components: the numeric detector applied one column at a time. Hence, the correction has something to correct, and the semantic predicate has its own direction generator and distinct-draw rule.

The text column needs care on two counts. First, the null must be drawn on the column that was *released*, under fresh keys, because a blind verifier holds the release and nothing else; scoring the original column would use data the threat model denies it. Second, Proposition 1 supplies that null's variance but not its shape, so the p-value is the empirical upper tail over fresh keys rather than a normal tail read off the standardized score. An empirical tail over N keys cannot fall below $1/(N+1)$; with $N = 20{,}000$ the floor is $5.0 \cdot 10^{-5}$.

The two tables exercise different halves of the decision. `Wine` is the stronger test of the text component: its only numeric column, `price`, is whole dollars and is left exact, so nothing but text

Table 15: Family-wise decisions. Family-wise level 0.001, Bonferroni-corrected across the marked columns, 4,000 rows each, text under the global hyperplane. Text p-values are empirical upper tails over 20,000 fresh keys on the released column; numeric columns use the distinct-value normal tail, where the per-item bits are independent. `Wine` leaves its price column exact, so its verdict rests on text alone. Only the watermarked arm is accepted in any table.

Table	Release	Driving column	p	Decision
`Airbnb`	watermarked	`latitude`, z=60.8	$< 10^{-300}$	**watermarked**
C=3	clean control	`longitude`, z=0.29	0.387	no evidence
	watermarked, wrong key	`summary`, $z = -0.15$	0.518	no evidence
`California`	watermarked	`Tax assessed value`, z=54.9	$< 10^{-300}$	**watermarked**
C=4	clean control	`Tax assessed value`, $z = -0.4$	0.671	no evidence
	watermarked, wrong key	`Tax assessed value`, z=0.8	0.220	no evidence
`Wine`	watermarked	`description`, z=40.4	$5.0 \cdot 10^{-5}$	**watermarked**
C=1	clean control	`description`, z=1.47	0.144	no evidence
	watermarked, wrong key	`description`, $z = -2.09$	0.676	no evidence

can carry the verdict. Proposition 1, applied to each column's own centered embeddings, predicts a fresh-key null standard deviation of 1.35 on the clean column (measured 1.35) and 4.37 on the watermarked release (measured 4.38). Selection changes the released geometry and so widens its null; the formula still holds, but it gives only the variance, so the decision uses the empirical tail on the released column. The watermarked release reaches the floor at $p = 5.0 \cdot 10^{-5}$ and rejects at the 0.001 level. This run uses its own key, so its scores ($z = 40.4$) differ from Table 13 (34.2).

`Airbnb` exercises the correction. The two coordinate columns reject on their own at $z > 60$ over thousands of distinct values each, and the summary column now rejects as well: $z = 41.6$ reaches the floor $p = 5.0 \cdot 10^{-5}$, well below the corrected threshold of $3.3 \cdot 10^{-4}$. Its release null has standard deviation 2.86, against 2.87 from Proposition 1 on the same released embeddings (1.55 on the original column). Neither control is accepted in either table. On `California` (C=4, threshold $2.5 \cdot 10^{-4}$) the three tax and lot columns reject at $z \geq 31.8$, and the summary column alone also clears the threshold ($z = 41.1$, $p = 5.0 \cdot 10^{-5}$); neither control is accepted.

Decisions under attack. Table 16 applies the same decision, with 20,000 fresh keys per text column and Bonferroni across columns, to attacked releases and to attacked clean and wrong-key controls. Rounding to two decimals and reselecting every text cell from the owner's pool are channel-specific attacks; reselection assumes access to the owner's candidate pool and is stronger than a blind paraphrase.

Each channel carries the verdict when the other is attacked, and removing both removes it. Text evidence also survives a fresh paraphrase by a different model and a paraphrase aimed with the release-only direction estimate; only reselection from the owner's own candidate pool removes it. The text column alone decides the rounded `Airbnb` release, the case where the numerical evidence was destroyed.

F.1.2 Component-Design Ablation

A mixed table could in principle be marked by running a numeric scheme on its numeric columns and a text scheme on its text column, and adding the two scores. We built and measured that composition. The numeric side is a green-list rule in the style of Ngo et al. (2026) applied per column; the text side chooses among the same 8 paraphrases per cell, but under a direction shared across rows and scored uncentered, as a scheme designed for a single document would. Both components are given the same key, the same columns, and the same paraphrase pool as HashMark, which we run with its default global hyperplane and with row-seeded hyperplanes. The remaining differences are statistical (direction sharing, centering, and what is counted), and the comparison isolates those. We report both schemes on the same releases and on *clean* tables the marking never touched.

On the watermarked `Airbnb` release (published at $c = 6$), the composition scores 105.4 against 91.2 for HashMark with its default global hyperplane, and 37.3 after rounding and paraphrase

Table 16: Family-wise decisions on attacked releases (level 0.001, Bonferroni, 20,000 fresh keys, global hyperplane; 4,000 rows for `Wine`, 2,000 for attacked `Airbnb`). Each cell gives the verdict and, in brackets, the driving column's p-value. No clean or wrong-key control is accepted under any attack. *A 2,000-row release attacked with a different paraphraser (PAWS-tuned T5, 4 candidates per cell, attacker's own sampling): *fresh* releases a random candidate; *guided* releases the candidate furthest to the invalid side of the direction the attacker estimates from the table it attacks. Clean controls are attacked the same way, with the attacker's paraphrases of the clean text.

Table	Attack	Watermarked release
`Airbnb`	none	**watermarked** (coordinates, $p < 10^{-300}$)
C=3	round numbers to 2 decimals	**watermarked** (summary, $5.0 \cdot 10^{-5}$)
	reselect every text cell	**watermarked** (coordinates, $p < 10^{-300}$)
	both	no evidence (min $p = 0.16$)
	fresh paraphrase*	**watermarked** (text z=20.3, $5.0 \cdot 10^{-5}$)
	guided paraphrase*	**watermarked** (text z=12.3, $5.0 \cdot 10^{-5}$)
	round numbers + guided*	**watermarked** (summary, $5.0 \cdot 10^{-5}$)
`Wine`	none	**watermarked** (description, $5.0 \cdot 10^{-5}$)
C=1	reselect every text cell	no evidence ($p = 0.38$)
	fresh paraphrase*	**watermarked** (z=20.2, $5.0 \cdot 10^{-5}$)
	guided paraphrase*	**watermarked** (z=12.0, $5.0 \cdot 10^{-5}$)

against 0.3. On a table that was never marked, however, it scores 35.5, above several of its own attacked detections, where HashMark scores 1.3. Read against the nominal threshold, the composed scheme reports a watermark on unmarked data.

A threshold between 35.5 and 105.4 would still separate these examples, so we calibrated both schemes empirically and compared them on equal terms. For each, we drew 500 fresh keys, scored the clean column under each one to obtain a null for the combined statistic, and standardized every arm by that null. Table 17 reports the result.

Table 17: Composed baseline and HashMark after empirical calibration. Every score is standardized by that scheme's own null, measured over 500 fresh keys on the clean column, which puts the schemes on a common scale but does not normalize any score. Separation is the gap between the watermarked release and the clean control, in units of the scheme's null standard deviation. `Airbnb` compares the default global hyperplane, the row-seeded ablation, and the composition on one release published at $c = 6$; the `California` columns are an earlier run of the row-seeded ablation at stored precision.

	`Airbnb`			`California`	
	global	row-seeded	composed	row-seeded	composed
null standard deviation	1.33	1.00	25.09	0.99	23.72
watermarked	68.70	89.22	4.20	90.98	4.34
clean control	0.94	0.14	1.41	−0.33	0.44
round numbers	22.26	27.41	1.83	31.13	1.73
paraphrase text	46.65	61.20	3.85	59.26	3.54
round *and* paraphrase	0.21	−0.61	1.48	−0.59	0.92
separation (null sd)	67.8	89.1	2.8	91.3	3.9

These are standardized scores, and standardizing is not by itself calibration: dividing by a null standard deviation does not make a non-normal null normal, so the composed column must not be read against the 3.09 normal reference. The table supports comparing separations. The composed statistic sits 2.8 and 3.9 null standard deviations above its clean control, against 67.8 for HashMark's global default and 89.1 and 91.3 with row-seeded hyperplanes. Two-decimal rounding, which a release pipeline performs without an adversary, moves the composition to 1.83 and 1.73, inside the range its clean controls already occupy, while HashMark stays above 22. The gain therefore

comes from centering and distinct-item counting, and it holds for the default design, not only for the row-seeded ablation.

The mechanism is statistical rather than architectural. A direction shared across rows makes the per-cell signs dependent, so the variance of the sum is not $m/4$, and Proposition 1 gives the inflation explicitly. Uncentered embeddings do not shift the mean across fresh directions (the sign count still has mean $m/2$ when the direction is drawn at random), but for any *fixed* key a corpus lying in a narrow cone produces a large systematic deviation, which is what a deployed detector with one key actually faces. Centering on a published reference mean and counting distinct items remove both effects, which is why the global default's clean control sits at 1.3 rather than 35.5.

G Robustness to Evidence Loss

G.1 Deletion, Partitioning, and Dilution

These transformations reduce the share or amount of useful evidence without requiring a rewrite of every marked value. For uniform deletion of scored items with fixed valid fraction, Proposition 3 gives $\mathbb{E}[z'] = z\sqrt{m'/m}$. Dilution instead retains m_w marked items of valid fraction v and adds distinct, chance-valid filler:

$$\mathbb{E}[z] = \frac{(2v-1)m_w}{\sqrt{m_w + m_c}} = (2v-1)\sqrt{m_w f}, \qquad f = \frac{m_w}{m_w + m_c}.$$

The square-root scaling is related, but the protocols and nulls are different.

Table 18: Representative evidence-loss results, with original protocols kept separate. Numerical deletion uses 200 `Shoppers` tables of 500 rows; text deletion uses 30 secrets and 50 subsets per secret, $q = 32$, and a fixed published reference; dilution uses 20 trials with 500 marked rows and distinct filler from a held-out synthetic pool. Rates are marked/clean fractions exceeding the raw 3.09 reference, not independent certification of a 0.1% FPR. In particular, global-text rates are descriptive, not calibrated detection rates. Here m counts distinct numerical items or surviving text cells; dilution's m includes filler.

Transformation	m	z (prediction)	Marked rate	Clean rate
Numerical deletion and partitioning				
None	1481	34.0	1.00	0.000
Keep 5% of rows	106	6.3	1.00	0.000
Keep 2% of rows	47	3.9	0.87	0.000
Drop 6 of 7 marked columns	229	12.8	0.99	0.005
20 disjoint partitions	106	6.3	1.00	0.001
Text deletion: global hyperplane (default)				
All cells	500	19.79 (19.79)	1.00	0.000
Keep 20%	100	8.85 (8.85)	1.00	0.000
Keep 5%	25	4.44 (4.43)	0.98	0.001
Text deletion: row-seeded ablation				
All cells	500	19.72 (19.72)	1.00	0.000
Keep 5%	25	4.40 (4.41)	0.98	0.000
Dilution: marked share among distinct items				
100%	2632	51.18 (51.30)	1.00	0.00
20%	13103	22.59 (22.99)	1.00	0.00
1%	258143	4.77 (5.18)	1.00	0.00

Deleting rows at random is not the same as deleting distinct values at random: a value that appears f_i times survives when a fraction a of rows is kept with probability $1 - (1-a)^{f_i}$. Repeated values, which we leave unedited, therefore outlive the edited ones that appear once, and the score falls a little faster than Proposition 3 predicts. For partitions, the rate is the fraction of pieces detected. Missing 95% of marked cells gives a similar result to retaining 5% of rows ($m = 106$, $z = 6.4$, marked rate 1.00).

Text deletion preserves the measured valid fraction near 0.94, and predicted and measured mean scores agree within 0.01 across the tested sizes. The average score crosses the threshold at about 13 surviving cells, but that says nothing about any single table; at 25 cells we measured detection in 98% of trials. Dilution leaves every marked value intact; at the smallest share the prediction runs slightly high because some filler values equal marked ones and are counted once. Both analyses describe random deletion and random filler, not an attacker who chooses what to delete or add.

H Numerical Representation Attacks

H.1 Choosing the Serialization Precision

We write each released value to exactly c decimals and hash that value (Algorithm 1a). Rounding or truncating it to c or more decimals leaves it unchanged, so every bit survives; on our releases, rounding one decimal below c removes the numerical evidence. The owner chooses c, and it need not be the column's stored precision: Melbourne `Airbnb` stores coordinates to twelve decimals, where marking tolerates virtually no precision reduction.

Table 19: Declared precision and rounding tolerance for `Airbnb` latitude and longitude (18,316 rows). The recommended *publish-at-c* protocol rounds the whole column before marking. Displacement is the mean absolute change from the originally stored values, including publication rounding, at 111.3 km per degree; it is not a worst-case bound (the observed maximum is 1.5 m at $c = 6$). Both rounded clean and marked columns lie entirely on the declared grid.

				z after rounding to d decimals				
c	grid step	z	displ. (m)	d=8	d=6	d=5	d=4	d=3
6	11 cm	163.22	0.113	163.2	163.2	0.6	−1.7	−2.6
4	11 m	28.90	3.611	28.9	28.9	28.9	28.9	1.7

Table 19 shows the trade-off: $c = 6$ tolerates six-decimal serialization at a mean total displacement of 0.113 m, whereas $c = 4$ tolerates four decimals at 3.611 m. For context, the 128-bin quantized predicate on longitude has mean displacement 347 m and cell-count $z = 90.8$ after two-decimal rounding, against 31.1 on the clean column (Appendix H.2). That wider-null statistic requires its own calibration, and the positive-value log-bin plan excludes negative latitude. This is a different rounding-tolerance operating point, not a matched comparison at equal tolerance.

Lower limit on c. Coarsening merges distinct values, increasing the repeated values left unedited and reducing capacity. At $c = 6$, 31,026 distinct items remain; at $c = 4$, 6,575 remain. At $c = 3$, 1,428 remain and the unattacked score is 5.03, above the 3.09 reference, but mean total displacement is already about 29 m. At $c = 2$, only 197 distinct items remain, 4.5% are singletons, and the score falls to 0.21 despite approximately 281 m displacement. At $c = 1$, no singleton values remain and $z = 0$. An owner therefore cannot buy arbitrary rounding tolerance by declaring a coarser grid: distortion increases while repeated values exhaust capacity.

Why round the whole column first. Why round the whole column, rather than only the values we edit? An earlier version snapped only edited values to the grid. At $c = 6$ the share of longitude values with exactly six decimals then rose from 9.8% to 88.7%, a pattern anyone could spot even though a classifier trained to separate the two columns did no better than chance (0.500). Rounding the whole column first puts every value on the grid, so nothing stands out. The cost is extra movement from rounding: at $c = 4$, 3.611 m on average, against 1.056 m when only edited values move.

H.2 Quantized Operating Points: Fidelity, Rounding, and Noise

HashMark-Q buys some tolerance to noise that scales values up or down; rounding is already handled by the declared precision of exact marking. What it costs in fidelity depends on the bin width (Table 41). A classifier trained to tell released columns from unmarked ones reaches AUC 0.898 at 32 bins, 0.635 at 256 and 0.510 at 1024, against 0.490 for an unmarked control run through the same pipeline (0.5 means it cannot tell them apart). Every quantized number in this paper comes from the offset-preserving construction of Remark 5, evaluated by the protocol recorded in Table 20.

Table 20: Protocol for each quantized result. All rows use the same embedding construction, which moves a cell by whole bins and preserves its offset inside the bin; a value whose coordinate falls outside the fitted grid is placed at the target bin's center. The bin domain is the natural logarithm of the value, defined on positive values only. C2ST: accuracy of a classifier telling released rows from unmarked ones (0.5 is chance); TPR: share of marked tables detected; FPR: share of clean tables flagged.

Result	Statistic	Threshold	Sample	Metric
Tab. 41, fidelity	N/A	N/A	500 rows, 10 trials	C2ST, 5-fold
Tab. 22, rounding	cell count	attacked clean tables	30 tables, 500 rows	survival, z
Tab. 30, noise	cell count	empirical FPR	600 paired trials	TPR, 1%/0.1%
Tab. 21, occupancy	N/A	N/A	20 grid offsets	total variation

All quantized detection rows use the same cell-count statistic. Because cells occupying the same bin share a validity bit, the statistic is not binomial under the null. The rounding experiment therefore reports it beside the same statistic on attacked clean tables; the noise experiment reports TPR against thresholds taken from clean controls, which at 600 controls resolve a 1% rate but only nominally a 0.1% rate.

Bin domain. TabularMark quantizes a bounded *perturbation* and Ngo et al. (2026) move to the nearest valid bin of the value range; HashMark-Q quantizes the *logarithm of the value*, so the bin width sets a relative tolerance. The owner therefore sizes the bin width to the noise it wants to tolerate; Table 30 and Table 41 give the two sides of it on `Shoppers`.

Marginal structure. On `ExitRates` at 1024 bins the unwatermarked column occupies 568 bins, 276 of them invalid; the released column occupies 321 bins, all valid.

Blind detection and bin occupancy. TabularMark keeps the original table and can therefore quantize the perturbation. A blind detector evaluates validity on the released value alone, so the released column occupies valid bins at every width: sweeping from 64 to 65,536 bins on `ExitRates`, it occupies 25 to 4986 bins, all valid. The bin width therefore sets both the noise tolerance and the occupancy pattern, consistent with the value-only occupancy mechanism analyzed under the assumptions of Theorem 2.

Table 21: Histogram separation of a quantized release, `ExitRates` (log domain), 64 bins of width $\delta = 0.110$, robustness scale $\varepsilon = \delta/2$. Total variation between clean and released histograms, averaged over 20 random grid offsets. Separation is large at every bin width below ε and decays above it, consistent with the scoped separation bound of Theorem 2 where its lower bound is positive.

h/ε	0.125	0.25	0.5	1	2	4	8
TV	0.448	0.441	0.429	0.399	0.311	0.209	0.120

H.3 Rounding and Truncation

We round or truncate the marked columns of 30 quantized `Shoppers` TabSyn tables (500 rows, 32 bins) to d decimals. For every marked value whose bin satisfies the width condition of Proposition 5, we compare the measured probability of staying in its bin with the prediction $1 - s/w(x)$, where s is the mean absolute shift ($h/4$ for rounding and $h/2$ for truncation to $h = 10^{-d}$), and report the cell-count detection statistic on the attacked release and on the attacked clean table.

Survival is predicted to within 0.007 at every precision, and cell-count detection holds throughout, with $z \geq 28.13$ even when values are rounded to a single decimal. Clean tables stay near their unattacked level except at two decimals, where rounding merges values into repeats and lifts the clean mean to about 2.6; there the threshold should be read from the attacked clean distribution rather than a normal tail. At one decimal 35–40% of values are sent to zero, where the log-domain bin plan does not score them, so those cells leave the statistic rather than counting against it.

Table 22: Quantized survival under rounding and truncation, `Shoppers`, 30 tables. *In range* is the share of marked values meeting the width condition; *to zero* the share rounded to 0. Detection is the cell-count z, in which every scorable cell is one trial; because repeated values share a bit this statistic is not binomial under the null, so the same statistic on the clean table under the same attack is shown beside it. Unattacked: $z = 40.91$, clean 1.03.

Attack	d	In range	Survival (predicted)	To zero	z	z clean
Round	5	1.00	1.000 (0.999)	0.00	40.89	1.02
Round	4	1.00	0.995 (0.995)	0.00	40.63	0.99
Round	3	0.99	0.958 (0.959)	0.00	38.50	0.98
Round	2	0.74	0.934 (0.932)	0.03	31.07	2.65
Round	1	0.54	0.998 (0.997)	0.35	28.74	0.69
Truncate	5	1.00	0.999 (0.999)	0.00	40.85	1.02
Truncate	4	1.00	0.989 (0.990)	0.00	40.32	0.99
Truncate	3	0.96	0.938 (0.939)	0.00	36.42	0.98
Truncate	2	0.64	0.960 (0.953)	0.10	28.13	2.54
Truncate	1	0.54	0.996 (0.995)	0.40	29.87	0.57

I Text Attacks and Direction-Policy Ablations

I.1 Semantic Predicate and Attack Ablations

I.1.1 Global-Hyperplane Results

We evaluate the text watermark of Section 2.1 on the Women's E-Commerce Clothing Reviews dataset, a table combining numerical (`Age`, `Rating`, `Positive Feedback Count`), categorical (`Division`, `Department`, `Class`) and free-form text (`Review Text`) columns, with a binary downstream label (`Recommended IND`). We watermark only the text column, on $m = 4000$ rows. Downstream classifiers use the one-hot encoded tabular columns concatenated with the sentence embedding of the text column, over five random $75/25$ splits.

Setup. Candidates come from a public T5-based paraphraser. The watermarked column scores $z = 45.09$ over $m = 4000$ cells, against $z \approx 0$ unwatermarked. The mean embedding must be the published one; recomputing it on the tested column drives z to chance. Appendix A.2 gives the paraphraser, the budget artifact it introduces, and the centering measurement.

Robustness: how strong an attack, at what cost to the attacker. The attack above uses the embedder's own paraphraser and decoding parameters. We sweep attack strength on two axes: decoding (*mild* T=0.8, *matched* T=1.2, *aggressive* T=1.5, *extreme* T=2.0) and the number of successive rewrites (1–3), for 24 settings in total. To keep the sweep tractable it runs on a 1000-row subsample, which reproduces the full run's per-cell behavior closely: identical valid fraction $v = 0.856$, and $z/\sqrt{m} = 0.712$ against 0.713 at m=4000.

Figure 4 and Table 27 show survival under the tested matched rewriting protocol: under the matched attacker it survives all three rounds ($z = 9.30 \to 5.38$), and only aggressive and extreme rewrites, which incur greater embedding cosine loss, remove it.

Across all 24 settings the score depends mainly on how far the attacker moves the meaning, not on how it does so, and the angle model of Appendix C.2 reproduces the curve without fitting (Appendix I.2).

Comparison with a Semantic-LSH Baseline Hou et al. (2024a) select sentences by locality-sensitive hash of an embedding and detect with a one-proportion Z-test, which is a related sentence-level construction. Our baseline is a global-hyperplane adaptation within the table protocol, not a full reproduction of SemStamp. We build it on our own candidate pools so that only the predicate differs: same cells, same pools, same test. We take a single hyperplane in each arm, so the arms are comparable cell for cell, and estimate the null on the *unwatermarked* column over 200 independent hyperplanes, which is what a false-positive rate is about.

Table 23 reports the result.

Table 23: Semantic-LSH baseline, $m = 500$ cells of clothing reviews, MiniLM embeddings, null over 200 hyperplane draws. The final column is a two-sided diagnostic: the fraction of clean draws with $|z| > 3.29$ (nominal two-sided Gaussian level 0.1%). It is not the one-sided ownership test.

Predicate	z (watermarked)	Null	Two-sided exceedance $\lvert z\rvert > 3.29$
Uncentered LSH (Hou et al., 2024a)	$+19.86$	-0.05 ± 10.55	77.5%
+ margin rule	$+14.67$	-0.68 ± 10.86	82.5%
Centered LSH (ours)	$+19.86$	$+0.07 \pm 0.71$	$\mathbf{0.0}\%$

The uncentered and centered first-valid arms have the same observed release score in this run, but centering can change which candidates are selected. Uncentered LSH has $155/200$ two-sided exceedances (77.5%; exact 95% interval $[71.1, 83.1]\%$); the margin arm has $165/200$. The centered global-hyperplane arm has $0/200$ (95% upper bound 1.83%), with spread 0.71 rather than 1, which is why we decide text against fresh keys. The comparison concerns this table adaptation of SemStamp, not SemStamp's own protocol.

Release-Only Hyperplane Estimation Recovering a watermark's secret structure from released outputs is the attack of Jovanović et al. (2024) in the language-model setting, and we construct its analog for the semantic predicate. The encoder and the mean embedding $\bar{e}$ are public, and marking pushes every released cell toward the valid side of r. Averaging the centered embeddings of a release therefore points partly along r: the estimate $\hat{r} \propto \sum_i (e_i - \bar{e})$ has $\cos(\hat{r}, r) = 0.31$, against 0.00 on an unmarked column.

Rescaling each direction of the embedding space to equal spread before averaging (whitening, computed from the release alone) raises $\cos(\hat{r}, r)$ to 0.41. If the attacker could move embeddings freely, pushing each cell just across the estimated hyperplane would drive the column from $z = +22.52$ to $z = 0.00$ at a mean cosine cost of only 0.0044. Real attackers must move text, not vectors, which is much harder, as the next subsections show.

Across the tested encoders, corpora, and selection rules, release-only estimation achieves cosine 0.17–0.53 with the secret direction, against approximately zero on clean columns. The embedding-space cost of crossing the raw 3.29 reference is at most 0.0044; first-valid selection is more exposed than margin selection in the MiniLM comparison (0.0009 versus 0.0044). These are unconstrained vector displacements, not realizable text edits or calibrated attack decisions. Section I.1.1 and Table 25 retain the operational test: selecting actual paraphrases using the estimate.

Fresh-Text Attack with the Strongest Paraphraser The frontier was traced with small sequence-to-sequence paraphrasers. DIPPER (Krishna et al., 2023) is the standard paraphrase attack in this literature: an 11B T5-XXL fine-tune with explicit lexical- and order-diversity controls, roughly fifty times the size of the attacker used above and built for this purpose.

Table 24: DIPPER against the semantic watermark, $m = 500$, marked column at $z = +16.10$. *Damage* is $1 - \cos$ to the released cell. p is the measured per-cell bit survival, and $\hat{p}$ the prediction of $p = 1 - \bar{\theta}/\pi$ from the measured angle alone.

lex. div.	ord. div.	Damage	p	$\hat{p}$	z	$z > 3.29$
20	0	0.087	0.896	0.866	11.99	yes
40	40	0.205	0.788	0.793	8.41	yes
60	60	0.306	0.738	0.744	6.71	yes
80	80	0.451	0.614	0.685	1.34	**no**

DIPPER does not beat the frontier: it reaches mean embedding cosine loss 0.451 to bring the column under the threshold, where the smaller paraphrasers of Section I.1.1 needed about 0.30. Per unit of damage it is the less efficient attacker, as the frontier predicts.

The angle model is confirmed on an attacker it has never seen. Feeding only the measured cosine into $p = 1 - \bar{\theta}/\pi$ reproduces the bit survival to within 0.006 at two of the four settings and 0.03 at a third. The exception is the most aggressive setting, where the model overpredicts survival (0.685 against

0.614); the same residual appears under the smaller attackers and is discussed in Appendix I.2. The relation between damage and score thus holds for the largest paraphraser we tested.

Fresh-Text Realization of the Estimate Section I.1.1 operates in embedding space. An adversary in text space needs selection rather than inversion: given q paraphrases of a released cell, they keep whichever crosses the estimated hyperplane while staying closest to the original. Neither step needs the secret.

We test that with $q = 8$ DIPPER paraphrases per cell at moderate diversity, $m = 300$. All three arms draw from the same pool, so only the selection rule differs: *blind* takes a random candidate, *quality* takes the one with maximum cosine, and $\hat{r}$*-guided* takes the closest candidate among those on the invalid side of $\hat{r}$.

Table 25: Text-space attacks drawing from a shared DIPPER pool. $m = 300$ clothing-review cells, marked column at $z = +13.05$, threshold 3.29; damage is $1 - \cos$ to the released cell. All three attackers draw from the same DIPPER pool, so only the selection rule differs.

	Blind		Quality		$\hat{r}$-guided	
q	z	Damage	z	Damage	z	Damage
8	6.00	0.198	9.70	0.097	8.31	0.119
16	6.93	0.190	10.05	0.075	7.27	0.117
32	7.16	0.190	10.51	0.069	6.00	0.122
64	6.93	0.188	10.28	0.060	**5.66**	0.130

The guided attacker gains with budget at constant damage and reaches $z = 5.66$ at $q = 64$, still well above the detection threshold.

The gain decelerates: successive doublings move z by -1.04, -1.27 and -0.34.

Crossing the plane requires a candidate on its far side, and the angular spread of a paraphraser's output is bounded, so the supply of crossing candidates saturates. The displacement that is cheap in embedding space is expensive in text, and detection holds at every budget tested.

I.1.2 Row-Conditioned Directions (Ablation)

This ablation compares our global hyperplane with hyperplanes seeded per row, each derived from the key and a few fields κ_i of its row. The global hyperplane does not depend on the rest of the row, but it is one direction that an attacker can estimate from the whole column. Per-row hyperplanes spread the signal over many directions, but the verifier then needs the named fields and needs each text cell to stay in its row; shuffling the text or deleting a named field prevents verification. The experiments below keep both intact.

We compare four key choices on the same 500 clothing-review cells with 32 candidate paraphrases each: a single global hyperplane, the product class (18 keys), department, class and rating (60 keys), and every stable tabular column (498 keys). The attacker's estimate is the whitened column-wide direction, replaced by a per-group mean wherever at least five rows share a key. *Removal* pushes each marked cell across its estimated hyperplane in embedding space, and *forgery* pushes each clean cell onto the valid side to fake a mark; the selection attack picks, among real paraphrases, the one the estimate calls invalid that is closest to the release.

For the estimator tested here, keying on every stable column reduces its cosine with the secret hyperplanes to 0.001; displacement leaves detection at $z = 19.32$, forgery leaves a clean column at $z = -0.45$, and the selection attack does not remove the mark. These measurements do not cover context-column deletion or independent text permutation. Owner-pool reselection that releases a different cached candidate, at a mean cosine cost of 0.29, removes the mark under every key choice ($z = -1.25$ with full row seeds). This pool-access test measures selection dependence. Detection is calibrated for every key choice by drawing the null with one fresh hyperplane per distinct key: rows sharing a class key correlate and widen the null to 2.36, and the 99.9th-percentile threshold estimated from these calibration draws is 6.62; held-out tail validation is a separate requirement while every release still clears its threshold by a factor of three or more. Thus the ablation exchanges resistance

Table 26: Row-seeded hyperplanes, 500 cells, $q = 32$. Null SD and the calibrated 0.1% threshold come from 5000 fresh secrets with one hyperplane per distinct key. Damage and cost are $1 - \cos$ to the released cell.

Key	z release	Null SD	Threshold	$\cos(\hat{r}, r)$	Removal z (cost)	Forgery z (cost)	Selection z (damage)
Global hyperplane	20.12	0.72	2.50	0.390	-9.30 (0.029)	$+12.16$ (0.008)	-4.38 (0.198)
Class	20.30	2.36	6.62	0.129	-11.81 (0.175)	$+0.63$ (0.001)	$+9.30$ (0.165)
Department, class, rating	19.41	1.64	4.74	0.087	-6.53 (0.162)	-2.95 (0.000)	$+9.57$ (0.142)
All stable columns	19.77	1.00	3.22	0.001	$+19.32$ (0.005)	-0.45 (0.001)	$+6.26$ (0.174)

to one estimator for dependence on stable row context; it does not identify a universally preferable direction policy.

I.2 Fresh-Text Attacks: Global-Hyperplane Frontier

Across the 24 settings of the global-hyperplane sweep, the score falls steadily as the attacker moves the text further from the release (correlation 0.978 between score and cosine loss). Every attack that keeps cosine similarity at 0.698 or more leaves the score above the threshold, and every attack at 0.656 or less brings it below; measured after subtracting the mean embedding, the two boundaries are 0.57 and 0.51. These are empirical boundaries for this encoder, corpus, and sample size; Appendix I.4 varies those choices.

The angle model $p = 1 - \theta/\pi$ predicts the fraction of bits that survive with a typical error of 0.042. It overstates survival by 0.06–0.09 for the most aggressive rewrites, as expected since released cells were chosen using the hyperplane (Remark 6). Margin selection retains more evidence than first-valid in this sweep. A second independent attack draw gives the same ordering, with Pearson $r = 0.988$ and Spearman $\rho = 0.993$.

Table 27: Paraphrase-robustness sweep for the semantic (1-bit LSH) predicate, margin selection, m=1000, fixed published reference. p is the per-cell probability that the bit survives; *cos* is the mean cosine similarity the attack retained, i.e. the damage the adversary did to its own text. Scores $z \leq 3.29$ fall below the raw reference 3.29, which is not a calibrated threshold for the global hyperplane. The final blocks report the detector-reference convention and clean-column null diagnostics.

Attacker	Rounds	p	z	z pred. Eq. (3)	cos
No attack: $z = 22.52$, valid fraction $v = 0.856$					
mild (T=0.8)	1	0.791	12.52	13.19	0.847
	3	0.715	7.97	9.44	0.770
matched (T=1.2)	1	0.748	9.30	10.97	0.811
	3	0.655	5.38	7.16	0.698
aggressive (T=1.5)	1	0.668	5.82	7.98	0.747
	2	0.599	0.89	5.06	0.656
	3	0.578	-0.82	4.17	0.583
extreme (T=2.0)	1	0.613	1.96	5.65	0.644
	3	0.525	-5.63	1.93	0.414
Detector convention, watermarked column, no attack:					
fixed published reference				$z = +22.52$	
recomputed from column under test				$z = +0.00$ (first-valid), $+2.15$ (margin)	
Null calibration, 200 *hyperplanes, unwatermarked column:*					
uncentered				$z = +0.96 \pm 15.65$	
centered on published $\bar{e}$				$z = +0.05 \pm 0.80$	

An exact surface-string hash is not paraphrase-stable: under the matched rewrite attack its score fell to 0.51 on 4000 cells (Remark 2). This separate diagnostic is not part of the semantic frontier.

I.3 Fresh-Text Attacks: Row-Seeded Ablation

We compare global and row-seeded hyperplanes on the same corpus and blind paraphraser over 10 secrets. Selection is redone for each keying, with row directions derived from the structured fields. This comparison tests how context diversity affects both the mean score and its variation across keys.

Table 28: Attacker-cost frontier, global versus row-seeded hyperplanes. Same 1000 cells (932 distinct row seeds), same BART attacker, mean $z\pm$ s.d. over 10 secrets at matched semantic damage (cosine to the marked release).

Rule	**Hyperplane**	marked	0.04	0.08	0.17	0.29	0.44
margin	row-seeded	22.52±0.51	20.12±0.40	18.01±0.38	14.17±0.87	10.27±1.08	6.23±1.32
margin	global	22.03±1.35	19.37±1.61	16.70±2.27	12.85±2.56	8.74±4.24	4.76±5.72
first	row-seeded	22.52±0.51	16.91±0.64	12.78±0.68	8.44±0.60	4.86±0.94	2.80±0.70
first	global	22.03±1.35	15.59±1.61	11.62±1.78	7.65±2.65	3.71±4.03	1.74±6.18

Context diversity matters. On 500 clothing reviews with 498 distinct contexts, margin selection gives $z = 7.64 \pm 1.14$ with row seeds and 7.70 ± 3.09 with a global hyperplane at cosine loss 0.40. On AG News with only four contexts, the corresponding scores are 3.52 ± 1.97 and 4.09 ± 0.92 at loss 0.46. The news setting does not show the variance reduction observed with diverse review contexts.

With one global hyperplane the outcome depends on which secret was drawn: at damage 0.44 the global arm has s.d. 5.72 and individual secrets range from $z = 13.16$ down to -2.91, so the same attack removes the mark under some keys and leaves it far above threshold under others. Row seeds average over 932 independent directions and cut the spread to 1.32.

The variance reduction also appears with `all-mpnet-base-v2`. At cosine loss 0.42, margin selection gives 7.05 ± 0.55 with row seeds against 8.49 ± 6.27 with a global hyperplane; first-valid gives 3.46 ± 0.58 against 4.51 ± 6.38. Mean effects differ between encoders, while diverse row contexts reduce across-key spread in both review experiments.

DIPPER under row seeds (ablation). We compare the two keyings under the 11B DIPPER attacker on 500 reviews with 498 contexts and cached $q = 32$ paraphrase pools. At maximum diversity, the row-seeded score remains 5.90, while the global-hyperplane score is -2.68 (Table 29). This matched comparison uses a larger embedding budget than the $q = 8$ global-hyperplane experiment in Appendix I.1.1.

Table 29: DIPPER attack on global and row-seeded hyperplanes. 500 cells, 498 distinct row seeds, 11B DIPPER at four diversity settings, $q = 32$; reported cosine loss ($1 - \cos$ to the released cell). Threshold 3.29. *Valid* is the fraction of cells still on the valid side after the rewrite. Generation failures were 0–1 cells per setting.

Damage	0.10	0.22	0.31	0.45
row-seeded, z	14.76	11.45	8.32	5.90
row-seeded, valid	0.830	0.756	0.686	0.632
global, z	10.64	6.17	2.59	-2.68
global, valid	0.738	0.638	0.558	0.440

The DIPPER comparison reports one set of scores rather than an across-key interval. A separate owner-pool reselection test gives $z = -0.45$ with row seeds and $+1.97$ with a global hyperplane. That test grants access to the owner's alternatives, as in Appendix E.2.

I.4 Fresh-Text Attack Generalization

The measured frontier is configuration-dependent. A second paraphraser follows the curve fitted to the first within the tested settings (mean residual -0.006, RMSE 0.029; pooled correlation $r =$

0.986 over 18 settings). Changing the encoder from MiniLM to `all-mpnet-base-v2` reduces pool exhaustion from 14.4% to 8.4% at $q = 8$; at retained cosine 0.414, the reported scores are -5.63 and 14.23, respectively. Changing the corpus to AG News raises exhaustion to 26.7%, with unattacked $z = 14.74$, compared with 22.52 on reviews. These experiments vary one component at a time, except that the second-attacker run uses 500 rather than 1000 cells, so its transfer assessment uses per-cell survival rather than raw-score comparisons. They support treating encoder and pool budget as design parameters, not a universal robustness boundary; global-hyperplane scores remain descriptive without attack-specific calibration.

J Structural Attacks and Baseline Comparisons

J.1 Structural and Value Attacks against Baselines

Each comparison specifies what the attacker changes and what data the methods share. Detection rates show how often a marked table is found; false-positive rates concern unmarked controls.

The HashMark rows below were re-measured under the current Edit Mode, after the change that leaves repeated values exact, using a standalone harness that reproduces the protocol of the MUSE benchmark code (its data pipeline, attacks and metrics) from the same TabSyn pool (200 tables of 500 rows). The MUSE rows are unaffected by that change and are carried over. We evaluate five adversarial scenarios on the Shoppers dataset with HashMark in edit mode: row shuffling, row replacement, column replacement, cell replacement, and value alteration. The three replacement attacks overwrite the selected rows, columns, or cells with values drawn from an independent clean table, so the released table keeps its shape, although the number of distinct scored items can change. Literal deletion, which removes items and therefore shrinks the evidence, is studied separately in Appendix G.1. Results are summarized in Table 32.

Here α is the attack intensity, the fraction of rows, cells or columns attacked (not a significance level). At low-to-moderate intensities ($\alpha \leq 0.4$), both HashMark and MUSE achieve near-perfect detection. The methods differ when most of the table is overwritten ($\alpha = 0.8$): MUSE collapses on the structural attacks (row replacement TPR $\rightarrow$ 0.26, cell replacement TPR $\rightarrow$ 0.34) while HashMark does not.

Row and cell replacement and shuffling leave TPR at 1.00 even when 80% of the data is overwritten, and column replacement at 80% leaves 0.73. Each surviving row still carries its own evidence, so overwriting part of the table leaves the rest scorable; this is a statement about replacement, not about deletion. This suits releases that are heavily subset or partitioned, since the evidence is carried by each row's own values rather than by the table's global statistics.

Two attacks remove the exact mark, for the same reason. The exact predicate (with 10^{-6} precision) drops to near-zero TPR under multiplicative Gaussian noise, and it is also broken by *rounding*: rounding to two decimals drives the statistic to $z = -0.2$ at 500 rows (Table 37). A pipeline that writes `%.4f`, a cast to `float32`, or a column typed `NUMERIC(10,2)` removes the mark without an adversary. Both follow from hashing the exact value, which makes every cell sensitive to arbitrarily small perturbations, and both require the released column to carry more precision than its semantics need: the $c = 6$ setting used throughout assumes at least eight significant decimals of slack, which many real numeric columns do not have. It is reduced, but not removed, by the *quantized* predicate of Remark 5. Table 30 evaluates HashMark-Q, which hashes a log-domain bin index rather than the raw value, under multiplicative Gaussian noise $x' = x(1 + \alpha\varepsilon)$ on all numerical columns. Against rounding the quantized predicate retains detection at every tested precision, though at one decimal 35–40% of values leave the bin plan (Appendix H.3). Against noise it buys a bounded amount: where the exact predicate collapses to TPR ≈ 0 at $\alpha = 0.2$, HashMark-Q at 16 bins detects in every trial, and at $\alpha = 0.8$ the best configuration retains TPR 0.88 at the 1% operating point and 0.52 at 0.1%. The amount bought depends on bin width and on the column's dynamic range, and the paragraph after the table gives the condition.

Quantization restores what the exact predicate loses to noise, but not without limit. At $\alpha = 0.2$ a quantized release is detected in essentially every trial where the exact predicate is detected in none; by $\alpha = 0.8$ the coarsest setting still retains TPR 0.88 at a 1% false-positive rate, and for $\alpha \geq 0.4$ the ordering down each column is monotone in bin count. Tolerance is bought by widening bins,

Table 30: Robustness to multiplicative Gaussian noise $x' = x(1+\alpha\varepsilon), \varepsilon \sim \mathcal{N}(0,1)$, on `Shoppers`. The quantized rows use the cell-count detector on 600 paired trials of 500 rows. We report TPR at nominal 1% / 0.1% FPR, with thresholds set at the 0.99 and 0.999 quantiles of 600 clean controls subjected to the same attack. 600 controls resolve the 1% level; the 0.1% column is nominal, since its empirical p-value floor is $1/601 \approx 0.17\%$ and the false-positive rate is not tested on an independent clean set. Table 41 reports the fidelity cost of wider bins.

Method	α=0	α=0.2	α=0.4	α=0.6	α=0.8
HashMark (exact)	1.00 / 1.00	0.00 / 0.00	0.03 / 0.01	0.00 / 0.00	0.06 / 0.04
HashMark-Q, 8 bins	0.99 / 0.99	0.99 / 0.99	**0.99 / 0.98**	**0.95 / 0.81**	**0.88 / 0.52**
HashMark-Q, 16 bins	1.00 / 1.00	**1.00 / 1.00**	0.98 / 0.84	0.83 / 0.32	0.65 / 0.18
HashMark-Q, 32 bins	1.00 / 1.00	1.00 / 1.00	0.73 / 0.39	0.32 / 0.07	0.16 / 0.03
HashMark-Q, 64 bins	1.00 / 1.00	0.97 / 0.85	0.18 / 0.06	0.07 / 0.03	0.04 / 0.01
HashMark-Q, 128 bins	1.00 / 1.00	0.34 / 0.04	0.03 / 0.00	0.03 / 0.01	0.03 / 0.01

while Table 41 shows that a wide bin is what a two-sample test detects: at 32 bins the release is separable, at 1024 it is at the control floor and carries no noise tolerance. In the configurations evaluated here the two requirements pull in opposite directions, and none of the tested bin counts is simultaneously at the control floor and noise-tolerant. That is a statement about these datasets, bin counts and detectors, not an impossibility result.

Scope of the robustness comparison. Table 32 compares HashMark against MUSE on a *shared* pool of TabSyn-generated tables: both schemes watermark the same tables, the same attacks are applied to each, and detection uses a common empirical TPR-at-FPR computed over 200 tables of 500 rows. This makes the two columns directly commensurable. MUSE is the natural head-to-head baseline because its detector, like ours, is blind and post-hoc.

A fully shared-table comparison is not possible for every prior scheme. TabWak embeds its watermark *during* generation and detects by inverting samples back through its own diffusion model, so it cannot watermark a table produced by a different generator; its watermark also lives in the latent geometry rather than in cell values, and its detector is a parametric resampling test rather than an empirical ROC. We nonetheless evaluate TabWak (and its stronger variant TabWak*) directly, training it on the same dataset and subjecting its generated tables to the same attacks its released code supports: row deletion, cell deletion, and value perturbation ("noise"), using its own detector and authors' parameters. Because the generation, watermark carrier, and detection statistic all differ, we report these results in a separate Table 31 rather than interleaving them with the shared-table numbers, and read them qualitatively. TabWak's numbers here reproduce the qualitative behavior reported by its authors (robust to row operations, fragile to cell- and value-level edits), which cross-validates our reimplementation.

TabularMark (Zheng et al., 2024) is edit-based like HashMark, with perfect clean-data detection on our tables (TPR $= 1.0$). Its detection is non-blind: it re-locates a fixed set of key tuples and reads their perturbation *relative to the owner's stored originals*. In its original setting this re-location uses a primary key; synthetic release tables carry none, so we identify key tuples by a content hash of the remaining attributes. Under this proxy TabularMark inherits its expected strength, namely perfect detection under row shuffling and row deletion at every intensity, since content matching is order-invariant and survives row drops, and collapses under cell deletion, column deletion, and value alteration. The latter three reflect the content hash: any change to another cell or column changes the hash, so the key rows can no longer be found (deleting one column changes the hash of every row, so column deletion reads 0 everywhere). The underlying difference is architectural: TabularMark requires the original dataset and stable tuple identity at detection time, whereas HashMark detects blindly from the released table alone, and HashMark additionally withstands the cell- and column-level edits that defeat identity-based re-location.

Across the schemes we can place on a common footing, each has a distinct blind spot (MUSE degrades under row and cell replacement, TabWak collapses under cell deletion and value perturbation), while HashMark retains TPR ≥ 0.98 under every structural attack tested except column

Table 31: TabWak comparison (TPR, Shoppers). TabWak generates and detects on its own tables via diffusion inversion, and its TPR is parametric rather than an empirical ROC. Row shuffling and column deletion are omitted (unsupported by its attack code). Both variants are robust to row deletion but collapse under cell deletion and value perturbation, the regime where HashMark retains TPR $= 1.0$.

	TabWak				**TabWak***			
Attack (α)	**0.2**	**0.4**	**0.6**	**0.8**	**0.2**	**0.4**	**0.6**	**0.8**
TPR @ 1% FPR								
Row Deletion	0.83	0.87	0.97	0.97	1.00	1.00	1.00	1.00
Cell Deletion	0.00	0.00	0.00	0.00	0.11	0.00	0.00	0.00
Value Perturb.	0.10	0.00	0.00	0.00	0.61	0.31	0.01	0.01
TPR @ 0.1% FPR								
Row Deletion	0.55	0.61	0.83	0.85	1.00	0.99	0.99	1.00
Cell Deletion	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00
Value Perturb.	0.04	0.00	0.00	0.00	0.34	0.10	0.00	0.00

replacement at 0.8 (0.73), weakening under aggressive value alteration and failing under noise. Which blind spot matters depends on the pipeline a release passes through.

Table 32: TPR for HashMark and MUSE under structural and value attacks on 200 `Shoppers` tables of 500 rows. Both methods use the same TabSyn pool and attacks; 200 controls resolve false-alarm rates down to 0.5%. †Quantized variant, cell-count z, TPR at the 1% threshold of 600 clean controls under the same noise (Table 30); 8 bins is visible to a two-sample test.

	HashMark (Ours)				**MUSE** (Baseline)			
Attack	**0.2**	**0.4**	**0.6**	**0.8**	**0.2**	**0.4**	**0.6**	**0.8**
Row Replacement	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.93	0.26
Cell Replacement	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.96	0.34
Column Replacement	**1.00**	**0.99**	**0.98**	**0.73**	**1.00**	0.99	0.94	0.54
Row Shuffling	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Value Alteration	**1.00**	**1.00**	0.95	0.70	**1.00**	**1.00**	**1.00**	**1.00**
Gaussian Noise (exact)	0.00	0.03	0.00	0.06	**1.00**	**1.00**	**1.00**	**1.00**
Gaussian Noise (HashMark-Q, 8 bins)†	0.99	0.99	0.95	0.88	as above			

Two further datasets. To check that these outcomes are not specific to `Shoppers`, we run the same grid on `Magic` and `News`, two datasets from the MUSE benchmark whose raw UCI tables have real-valued columns that Edit Mode can mark directly, with no generator (200 tables of 500 rows each; Table 33). Unattacked scores average $z = 68.2$ on `Magic` and 32.8 on `News`, against clean controls near 0, and a gradient-boosted two-sample test cannot tell marked from independent clean tables (accuracy 0.50 and 0.49). `News` carries fewer markable values per table, so column replacement and value alteration at intensity 0.8 reduce its TPR to about 0.4; exact-value marking does not survive Gaussian noise on either.

MUSE on `Magic` and `News`. To widen the sweep beyond TabSyn, we use a second synthetic-data generator: both methods draw from one shared candidate pool of $400{,}000$ rows per table, sampled from a Gaussian copula fitted to it (empirical marginals, Gaussian dependence), and run through the unchanged MUSE harness with its attacks and 200 tables of 500 rows. MUSE uses its `separate` score function, as in Table 32. Using the same generator for both methods isolates the watermarks under a common candidate distribution.

On `News` at intensity 0.8 these rates differ from Table 33 because the protocols differ: here the tables come from the shared copula pool and TPR uses the 1% threshold of attacked controls, while Table 33 marks the raw UCI tables and uses $z > 3.09$.

Table 33: TPR at $z > 3.09$ under the attacks of Table 32 on `Magic` and `News`, 200 tables of 500 rows each, Edit Mode on the raw UCI tables. Columns are attack intensities. †Noise is measured on the quantized variant, whose purpose it is: cell-count z, TPR at the 1% quantile of 200 clean controls under the same noise; nonpositive values fall outside the log-domain bin plan and are not scored. Bins wide enough to tolerate this noise are visible to a two-sample test (C2ST 0.898 at 32 bins on `Shoppers`, Table 41).

	Magic				News			
Attack	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
Row replacement	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Cell replacement	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Column replacement	1.00	1.00	1.00	1.00	0.99	0.96	0.78	0.42
Row shuffle	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Value alteration	1.00	1.00	1.00	1.00	1.00	0.93	0.84	0.44
Gaussian noise, exact	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01
Gaussian noise, HashMark-Q 8 bins†	1.00	0.70	0.55	0.37	1.00	0.92	0.58	0.24
Gaussian noise, HashMark-Q 16 bins†	0.87	0.23	0.10	0.04	0.97	0.20	0.06	0.02

Table 34: TPR at 1% FPR on `Magic` and `News` in the MUSE harness, shared Gaussian-copula candidate pool, 200 tables of 500 rows. Replacement attacks refill deleted rows, cells or columns from an independent table.

		Magic				News			
Attack	Method	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
Row replacement	MUSE	1.00	1.00	0.99	0.35	1.00	1.00	0.98	0.26
	HashMark	1.00	1.00	1.00	**1.00**	1.00	1.00	1.00	**1.00**
Cell replacement	MUSE	1.00	1.00	1.00	0.53	1.00	1.00	1.00	0.40
	HashMark	1.00	1.00	1.00	**1.00**	1.00	1.00	1.00	**1.00**
Column replacement	MUSE	1.00	1.00	1.00	0.91	1.00	1.00	1.00	0.62
	HashMark	1.00	1.00	1.00	**1.00**	1.00	1.00	1.00	**1.00**
Row shuffle	MUSE	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	HashMark	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Value alteration	MUSE	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
	HashMark	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

MUSE on `Default`. Every column of `Default` is integer-valued, so Edit Mode at a sub-unit precision has nothing to mark. Its twelve bill and payment columns, however, are amounts in whole currency units with hundreds of distinct values per table, and we mark them on their own unit grid ($c = 0$): an invalid amount that occurs once steps up one unit at a time until its keyed bit is valid (the variant of Remark 4), which keeps every released value an integer. The credit limit (multiples of 10,000), age and the coded categories stay exact. The setup is otherwise that of Table 34; Table 35 reports the result.

Table 35: TPR at 1% FPR on `Default` in the MUSE harness, shared Gaussian-copula pool, 200 tables of 500 rows. HashMark marks the twelve amount columns at $c = 0$; unattacked $z = 60.5$. A gradient-boosted two-sample test on 500 marked against 500 clean rows scores 0.517 on all columns and 0.501 on the amount columns.

Attack	Method	0.2	0.4	0.6	0.8
Row replacement	MUSE	1.00	1.00	0.99	0.43
	HashMark	1.00	1.00	1.00	**1.00**
Cell replacement	MUSE	1.00	1.00	0.98	0.36
	HashMark	1.00	1.00	1.00	**1.00**
Column replacement	MUSE	1.00	1.00	0.87	0.45
	HashMark	1.00	1.00	1.00	**0.68**
Row shuffle	MUSE	1.00	1.00	1.00	1.00
	HashMark	1.00	1.00	1.00	1.00
Value alteration	MUSE	1.00	1.00	1.00	1.00
	HashMark	1.00	1.00	1.00	1.00

Marking `AGE` the same way shows where the unit grid stops helping: a 500-row table holds only about 41 distinct ages, almost all repeated and therefore left exact, and age alone reaches $z = 1.4$ (TPR 0.02). Low-cardinality integers carry too few distinct values to mark, as with the `price` column of `Wine` (Appendix B.3).

MUSE on `Beijing`. `Beijing` has one continuous, high-cardinality column, the cumulative wind speed `Iws`; the date fields, wind direction and rain and snow counts are codes or small counts, and the target is excluded. Dew point, temperature and pressure are integers with 60–69 distinct values. HashMark therefore marks `Iws` alone, under the setup of Table 34.

Table 36: TPR at 1% FPR on `Beijing` in the MUSE harness, shared Gaussian-copula pool, 200 tables of 500 rows. HashMark marks the single column `Iws`; unattacked $z = 8.9$, TPR 0.56.

Attack	Method	0.2	0.4	0.6	0.8
Row replacement	MUSE	1.00	1.00	0.90	0.17
	HashMark	0.92	0.59	0.16	0.04
Cell replacement	MUSE	1.00	1.00	0.98	0.36
	HashMark	0.80	0.27	0.08	0.03
Column replacement	MUSE	1.00	1.00	0.99	0.50
	HashMark	0.56	0.56	0.90	0.90
Row shuffle	MUSE	1.00	1.00	1.00	1.00
	HashMark	0.56	0.56	0.56	0.56
Value alteration	MUSE	1.00	1.00	1.00	1.00
	HashMark	0.01	0.01	0.00	0.01

With one marked column the evidence is thin and fragile. A 500-row table holds only 171–197 distinct `Iws` values, drawn from a few thousand that recur across tables, so under a fixed key the clean controls vary widely and TPR at 1% is 0.56 although the mean score is 8.9. Value alteration rescales the one marked column and removes the mark, and every attack that thins it lowers detection. MUSE, which scores whole rows, is stronger here at every setting except 80% column replacement. Marking the three integer columns on their unit grid, again moving only values that occur once and stepping upward (Remark 4), lifts unattacked TPR to 1.00 (temperature or pressure with `Iws`, or all three) and to 0.65 with dew point; a gradient-boosted two-sample test on 500 marked against 500 clean rows stays at 0.46–0.52 in every configuration, and each column keeps its

distinct values because repeated values are left exact. We report this as we did the `price` column of `Wine` (Appendix B.3): the unit grid adds evidence at no measured fidelity cost here, but these columns have too few distinct values to carry a table on their own.

Bin width, not bin count. A fixed bin count is a different bin width on every table, and tolerance also depends on how many column–bin cells a table occupies: on `Wilt` and `Housing`, whose marked columns span a narrow range, the same 8 bins give TPR at most 0.20 under the noise of Table 30.

J.2 Comparison with TAB-DRW

TAB-DRW (Zhao et al., 2026) normalizes columns and edits selected imaginary Fourier coefficients of each row toward pseudorandom signs. We run its released defaults ($\gamma = \delta = 0.5$, discrete columns rounded) on the same `Shoppers` TabSyn tables as HashMark: 30 tables of 500 rows. TAB-DRW's score is standardized against 200 clean tables. Table 37 retains the distinct trade-offs; the full two-size comparison remains in the source archive.

Table 37: Compact TAB-DRW comparison on 500-row tables. Attack rows report descriptive scores, not commensurate calibrated significance levels. Noise is multiplicative Gaussian. Two-sample AUC uses a gradient-boosted classifier against the same clean rows (0.50 is indistinguishable). The HashMark-Q entries retain the original comparison protocol; current cell-count noise calibration is reported separately in Table 30.

Condition	TAB-DRW	HashMark	HashMark-Q
None	9.5	34.0	8.9
One column deleted	−0.7	30.4	8.1
Noise 1%	9.5	−0.1	5.9
Noise 5%	9.4	0.2	3.2
Rounded to 2 decimals	9.0	−0.2	7.1
Clean table	0.0	−0.1	0.3
Two-sample AUC	0.93	0.50	0.92
Categorical cells changed	20%	0%	0%

TAB-DRW retains its score under noise and rounding that remove exact-value evidence, but loses it under the tested column deletion because its coefficients mix columns. HashMark retains numerical evidence under column deletion, leaves categorical cells unchanged, and has lower measured distinguishability; HashMark-Q trades fidelity for noise and rounding tolerance. The shipped TAB-DRW implementation uses a fixed seed and hard-coded secret; a deployment would need to replace those defaults rather than treat them as private.

K Fidelity and Downstream Utility

K.1 Independent Text Quality

Preserving predictive accuracy is not the same as preserving the meaning of individual records. Here we compare the original and released text directly, using models separate from the watermark's own encoder.

We regenerate candidates for 500 review cells with a 512-token budget, so no input is truncated, and release them under the default policy: a cell already on the valid side stays unchanged, an invalid cell takes its valid candidate furthest from the hyperplane, and a cell with no valid candidate stays unchanged. We compare released and original text with models unrelated to the watermark encoder: DeBERTa-large-MNLI entailment in each direction and RoBERTa-large BERTScore. The *paraphrase* arms release a random candidate from the same pool without watermarking, over all cells and over the cells the release changed, which isolates the cost of choosing a valid candidate.

Because two thirds of the cells are released unchanged, the default release is much closer to the original than a column of paraphrases: mutual entailment 0.746 against 0.282, BERTScore 0.964

Table 38: Text quality under independent models, 500 cells, $q = 8$, no truncation, default release policy. The release changes 165 cells (33%), leaves 65 cells with no valid candidate unchanged and invalid, and scores $z = +16.55$.

Cells	Arm	Orig. ⇒ rel.	Rel. ⇒ orig.	Mutual	Contra-diction	BERTScore	Length ratio
all 500	identity	1.000	1.000	1.000	0.000	1.000	1.00
	random paraphrase	0.694	0.332	0.282	0.098	0.893	0.87
	HashMark release	0.892	0.774	0.746	0.060	0.964	0.96
changed 165	random paraphrase	0.721	0.352	0.297	0.079	0.893	0.84
	HashMark release	0.673	0.315	0.230	0.182	0.891	0.87

against 0.893. On the cells it does rewrite, choosing a valid candidate has a measurable cost over a random paraphrase of the same cells: mutual entailment falls from 0.297 to 0.230 and contradiction rises from 0.079 to 0.182, while BERTScore is unchanged. Entailment measures whether one text supports the other's content; mutual entailment requires support in both directions. Contradiction rates are model judgments, not human-verified factual errors. Absolute mutual entailment of paraphrases is low for this generator under both arms, so we describe the operation as paraphrase selection and report its measured cost rather than claiming meaning preservation.

K.2 Paired Downstream Utility

This experiment asks whether models trained on marked data perform differently from models trained on the same data before marking. Comparing the same split and synthetic table in each pair helps separate watermark effects from training variability.

For each of 20 seeds we split the real data, fit one Gaussian Copula generator, draw one synthetic table, and apply Edit Mode to that same table. A classifier with a fixed seed is trained on each and scored on the same held-out real data, so the paired difference isolates the watermark. Before running, we fixed a margin of ± 1 accuracy point and ask whether the change is confidently inside it: we test separately that it is above -1 and below $+1$, each at level 0.05 (the standard two one-sided tests for equivalence).

Table 39: Paired change in downstream accuracy (watermarked minus unwatermarked, percentage points), 20 seeds. Every configuration is equivalent to zero within the ± 1-point margin.

Dataset	Model	Base accuracy	Mean change	95% CI	Equivalent
Wilt	XGBoost	94.33	$+0.000$	$[-0.038, +0.038]$	✓
Wilt	Random forest	94.57	$+0.000$	$[+0.000, +0.000]$	✓
Housing	XGBoost	49.82	-0.019	$[-0.049, +0.012]$	✓
Housing	Random forest	53.54	$+0.001$	$[-0.001, +0.003]$	✓
Shoppers	XGBoost	85.69	-0.023	$[-0.379, +0.334]$	✓
Shoppers	Random forest	86.11	-0.058	$[-0.325, +0.209]$	✓

K.3 Baseline Comparison: Full Metrics

This appendix reports all fidelity metrics and both sample budgets for the baseline comparison of Table 2. Exact keyed predicates use HMAC-SHA-256, and Edit Mode increments a value by 10^{-c} until it hashes to 0. MUSE uses its released default joint score function. On 10,000 `Shoppers` TabSyn rows with six marked columns, Edit Mode embedding takes 0.20 ± 0.01 s and detection 0.07 ± 0.00 s (Apple M2 CPU, five repeats); it runs after generation and adds no generator calls, where tournament sampling multiplies them by the tournament size.

Evaluation Metrics. We assess data quality using standard tabular synthesis metrics (Zhang et al., 2024; Fang et al., 2026).

- **Marginal Distribution (Marg.):** the evaluation suite's normalized similarity derived from KS distance for continuous columns and TVD for categorical columns; 1 is best.
- **Pairwise Correlation (Corr.):** the evaluation suite's normalized similarity derived from Pearson-correlation differences (numerical) and contingency-table TVD (categorical); 1 is best.
- **Classifier Two-Sample Test (C2ST):** the evaluation suite's normalized score, in which a logistic classifier's accuracy at distinguishing synthetic from real is mapped so that 1 means indistinguishable. Higher is better. Appendix H.2 instead report *raw* accuracy from a gradient-boosted classifier, where 0.5 means indistinguishable; the two are not comparable and we name the detector wherever a C2ST appears.
- **MLE Gap:** Downstream classifier accuracy delta between models trained on real vs. watermarked synthetic data. Lower is better.
- **Detection:** area under the ROC curve (AUC), and TPR at a threshold corresponding to a nominal FPR. AUC is computed empirically against paired controls. Runs with 200 controls resolve false-alarm rates down to 0.5%, so their "0.1%" columns use the threshold a standard normal gives for 0.1%. Higher is better.

K.3.1 Evaluation of HashMark

Full Baseline Comparison. We implement HashMark in "edit" mode with perturbation 10^{-6} and compare against four baselines: two generative (MUSE (Fang et al., 2026), TabWak) and two edit-based (TabularMark, WGTD). Results appear in Table 40.

Detectability. HashMark achieves AUC 1.000 and TPR 1.000 at the nominal 0.1% threshold across the two reported datasets and both sample budgets (100 or 500 rows). MUSE and the edit-based methods reach the same reported detection values, while TabWak is weaker. On `Adult`, MUSE's C2ST is 0.883 versus 0.996 for HashMark; on `Shoppers`, several baselines match or exceed HashMark on individual fidelity metrics. The result supports a favorable fidelity–detection trade-off, not unique Pareto dominance.

Fidelity & Utility. We measure distortion via Marginal Distribution (Marg.), Pairwise Correlation (Corr.), and Classifier Two-Sample Test (C2ST), and downstream utility via the MLE Gap (test accuracy delta between real and watermarked data).

Because HashMark (edit mode) applies only a 10^{-6} perturbation to existing values, it matches the unwatermarked baseline on the reported Adult and Shoppers metrics, and paired downstream-accuracy changes are statistically equivalent to zero (Appendix K.2). On `Adult`, HashMark achieves Marg. = 0.994 and Corr. = 0.984, against 0.979 and 0.963 for MUSE.

Utility. Across the reported real-world datasets, three synthesizers and downstream models, watermarking leaves accuracy essentially unchanged, and paired 20-seed intervals establish equivalence to zero within a one-point margin (Appendix K.2).

Robustness to attacks. Against the replacement-based attacks defined in Appendix J.1, HashMark retains nominal-threshold TPR 1.00 under row and cell replacement and shuffling at all tested intensities; column replacement reaches 0.73 at $\alpha = 0.8$. The exact predicate's two failure modes, additive noise and rounding, and the quantized predicate's visible marginal, are quantified in Appendix J.1 and not repeated here.

Table 40: Full baseline comparison. HashMark vs. generative (MUSE, TabWak) and edit-based (TabularMark, WGTD) baselines. HashMark matches the unwatermarked point estimates; some baselines score higher on individual Shoppers metrics. TPR at 0.1% FPR uses the standard-normal threshold; 200 controls resolve rates down to 0.5%. Darker and lighter mark the best and second-best watermarked method per fidelity column within each dataset.

(a) Fidelity metrics: higher is better except MLE gap

Dataset	Method	Marg. ↑	Corr. ↑	C2ST ↑	MLE gap ↓
Adult	w/o WM	0.994	0.984	0.996	0.017
	TabWak	0.935	0.885	0.769	0.048
	TabularMark	0.983	0.949	0.987	0.021
	WGTD	0.987	0.972	0.978	0.019
	MUSE	0.979	0.963	0.883	0.017
	HashMark	0.994	0.984	0.996	0.017
Shoppers	w/o WM	0.985	0.974	0.974	0.017
	TabWak	0.903	0.886	0.548	0.132
	TabularMark	0.974	0.930	0.975	0.013
	WGTD	0.964	0.944	0.887	0.008
	MUSE	0.982	0.974	0.950	0.015
	HashMark	0.985	0.974	0.974	0.017

(b) Detection: TPR at nominal 0.1% FPR

		100 rows		500 rows	
Dataset	Method	AUC	TPR	AUC	TPR
Adult	w/o WM	–	–	–	–
	TabWak	0.844	0.089	0.990	0.592
	TabularMark	1.000	1.000	1.000	1.000
	WGTD	1.000	1.000	1.000	1.000
	MUSE	1.000	1.000	1.000	1.000
	HashMark	1.000	1.000	1.000	1.000
Shoppers	w/o WM	–	–	–	–
	TabWak	0.860	0.106	0.990	0.353
	TabularMark	1.000	1.000	1.000	1.000
	WGTD	1.000	1.000	1.000	1.000
	MUSE	1.000	1.000	1.000	1.000
	HashMark	1.000	1.000	1.000	1.000

L Extensions and Diagnostics

L.1 Detector Strength

A watermark can be detectable with its secret key yet hard to distinguish without that key. The tests below ask the second question: can a classifier tell marked data from unmarked data? Chance-level performance is 0.5 on these raw accuracy scales, unlike the normalized C2ST score used in the main baseline table. The fidelity comparison of Table 40 uses the classifier two-sample test supplied with the evaluation suite, which fits a logistic model to normalized features. Elsewhere in this appendix we judge our own releases with a gradient-boosted classifier, which was more sensitive on these releases, and it would not be fair to hold the baselines to the weaker instrument. Re-running all four methods on the same artifacts under both detectors gives the following.

Standard HashMark scores within 0.008 of chance under all four detectors, while MUSE reaches 0.547 under gradient boosting. An unwatermarked control run through the identical pipeline scores 0.490–0.504, which fixes the noise floor of the measurement.

The HashMark-Q rows are the reason bin width is a published parameter rather than an internal constant. A quantized release is a release whose values have been moved to satisfy a keyed predicate, and the distance moved is bounded by the bin width; at 32 bins that displacement is large enough

Table 41: The same releases under four detectors. C2ST accuracy against the unwatermarked release; 0.5 is indistinguishable, and distance from 0.5 in either direction is evidence of distortion. `Shoppers`, 500 rows, five-fold cross-validation.

Method	Logistic	Random forest	MLP	Gradient boosted
TabularMark (Zheng et al., 2024)	0.470	0.467	0.489	0.460
MUSE (Fang et al., 2026)	0.528	0.518	0.512	0.547
HashMark (standard)	0.503	0.508	0.505	**0.504**
HashMark-Q, 32 bins	0.566	0.875	0.717	0.898
HashMark-Q, 256 bins	0.497	0.604	0.509	0.635
HashMark-Q, 1024 bins	0.504	0.508	0.496	**0.510**

for a gradient-boosted classifier to find (0.898), at 256 bins it is marginal (0.635), and at 1024 bins it is at the control floor (0.510). Table 30 reads the same axis from the other end: 1024 bins is also where noise tolerance is gone. Bin count therefore sets the fidelity–robustness trade directly (Appendix H.3). Against rounding the dial has a solution; 512–1024 bins hold cell-count $z = 20$–22 after rounding to two decimals at a C2ST of 0.52–0.55; and against multiplicative noise of the size in Table 30 it does not. Failure to distinguish these particular releases is not equality of distributions, and the MLP detector does not converge within its iteration budget, so its column is the least informative of the four.

L.2 Extended Related Work

Watermarking Tabular Data. Watermarking tabular data has been extensively studied, evolving from classical database protection to modern techniques for generative models. We categorize existing approaches into classical modifications, binning strategies, and sampling-based protocols, distinguishing our construction, HashMark, from these priors.

Classical database watermarks embed bits in least-significant digits or subset statistics keyed on a primary key (Agrawal & Kiernan, 2002; Sion et al., 2003; Shehab et al., 2008). Binning schemes place values in keyed intervals (He et al., 2024; Ngo et al., 2026; Zheng et al., 2024); generative schemes watermark at sampling time (Zhu et al., 2025; Fang et al., 2025; 2026). Cryptographic language-model watermarks (Christ et al., 2024; Fairoze et al., 2025; Christ & Gunn, 2024) embed through the generator's sampling randomness, whereas HashMark edits a finished release. Table 1 in Section 1.2 compares data types, detector access and embedding.