One-off Disclosure Control by Heterogeneous Generalization

Olga Gkountouna, *University of Liverpool;* Katerina Doka, *National Technical University of Athens;* Mingqiang Xue, *Tower Research;* Jianneng Cao, *Bank Jago;* Panagiotis Karras, *Aarhus University*

https://www.usenix.org/conference/usenixsecurity22/presentation/gkountouna

This paper is included in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

One-off Disclosure Control by Heterogeneous Generalization

Olga Gkountouna
University of Liverpool

Katerina Doka
National Technical University of Athens

Mingqiang Xue
Tower Research

Jianneng Cao
Bank Jago

Panagiotis Karras
Aarhus University

Abstract

How can we orchestrate an one-off sharing of informative data about individuals, while bounding the risk of disclosing sensitive information to an adversary who has access to the global distribution of such information and to personal identifiers? Despite intensive efforts, current privacy protection techniques fall short of this objective. *Differential privacy* provides strong guarantees regarding the privacy risk incurred by one's participation in the data at the cost of high information loss and is vulnerable to learning-based attacks exploiting correlations among data. *Syntactic anonymization* bounds the risk on specific sensitive information incurred by data publication, yet typically resorts to a superfluous clustering of individuals into groups that forfeits data utility.

In this paper, we develop algorithms for disclosure control that abide to sensitive-information-oriented syntactic privacy guarantees and gain up to 77% in utility against current methods. We achieve this feat by recasting data *heterogeneously*, via bipartite matching, rather than homogeneously via clustering. We show that our methods resist adversaries who know the employed algorithm and its parameters. Our experimental study featuring synthetic and real data, as well as real learning and data analysis tasks, shows that these methods enhance data utility with a runtime overhead that is small and reducible by data partitioning, while the β-likeness guarantee with heterogeneous generalization staunchly resists machine-learning-based attacks, hence offers practical value.

1 Introduction

Data sharing risks privacy violations and disclosure of sensitive information. As much as 99.98% of the USA population can be re-identified in naively anonymized datasets using 15 *quasi-identifying* (*QI*) attributes, such as age, occupation, marital status, residence type, or gender [47], while machine learning techniques can re-identify individuals and expose the values of *sensitive attributes* (*SA*s) [14, 33].

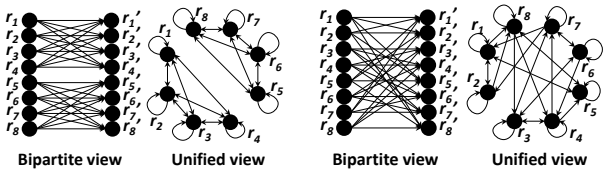
It is desirable to publish data in a way that is both *informative*, including potentially sensitive information, such as health condition or work class, about a population, and at the same time *safe*, curbing sensitive disclosures about individuals. This aim differs from another consideration arising in predictive modeling, where it is desirable to *remove all* information on protected variables, so as to, e.g., eliminate racial disparities in predictions [30]. To achieve the aim of informative and safe data publishing, one may employ differential privacy (DP) [17, 61]. The state-of-the-art approach for differential-privacy-based data publishing, PrivBayes [61], uses the Laplace mechanism [17] to inject noise. A similar discrete alternative, the *geometric* mechanism [23], offers a universally optimal tradeoff between privacy and utility in single-predicate count queries [22, 23]; however, such universal optimality is impossible in conjunctive count queries involving more than one attributes [8, 9]. More crucially, while differential privacy bounds the privacy risk incurred by the contribution of one's *independent* data to a database, it fails to control the privacy risk incurred by *correlated* data [34, 45, 49]. Those using DP in the hope of bounding inferences implicitly *assume* tuple independence [35, 36, 42], modeling data contributions as *causal interventions* bearing no associative effects [52]. A privacy-aware individual whose data is correlated to data of others would have to lobby those others to abstain from the data collection process [31, 52].

Institutions that share data in the real world commonly publish their data in an *one-off* manner. For example, the 2020 US census published data once, applying differential privacy[1] with $\epsilon = 19.61$, which means that the participation of any individual changes the probability of any outcome by a factor of no more than ~ 328 million. We infer that the US Census Bureau lacks a practically applicable, meaningful, and robust privacy guarantee for the one-off sharing of census data. In this context, *syntactic* privacy guarantees [11, 43, 48] offer a useful option: they consider an adversary having access to the global distribution of sensitive information, and bound the specific privacy risk on that information incurred by one-off publication [11]. To enforce such a guarantee, they *generalize*

[1] https://www.census.gov/newsroom/press-releases/2021/2020-census-key-parameters.html

QI attribute values to (i.e., substitute them with) less precise ones by eliminating precision digits or publishing a range, a set, or a category, instead of a given value [13, 20, 21].

Still, syntactic anonymization algorithms typically cluster records in groups and publish the records in each group as a unit sharing identical *QI* values; that is, for any two records in the same group, r_i, r_j, the respective published forms, r'_i, r'_j, have identical *QI* values; this enforced *homogeneity* results in loss of precision, hence low utility [3, 10]. A value generalization can be represented by a directed graph, the *generalization graph* [59], which shows how the values of original records match those of anonymized ones. In the *bipartite view* of the graph, an edge from the vertex standing for original record r_i, to the one standing for recast record r'_j indicates that the *QI* values of r_i are included in (*match*) those of r'_j. In the *unified view*, a single vertex represents both the original record r_i and its recast form r'_i.



(a) Homogeneous (b) Heterogeneous

Figure 1: Generalization types in graph view

Figure 1(a) shows an example homogeneous generalization graph [21, 38, 48]; in the bipartite view it forms two disconnected complete subgraphs of four vertices in each side (i.e., two $K_{4,4}$ *bicliques*); the *QI* values of a tuple on the left side are generalized to those of all its matches on the right side. In the unified view, these subgraphs appear as complete digraphs with self-loops. Contrariwise, heterogeneous generalization [16] abandons homogeneity; it generalizes records *heterogeneously*, as Figure 1(b) illustrates; thereby, it enhances utility because it allows for more flexibility in generalization, while retaining the same privacy guarantee [16].

Table 1: Example of heterogeneous generalization

ID	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Age	59	57	39	28	41	37	40	53
Salary	25	27	47	41	20	59	35	34

ID	Age	Salary
t'_0	53-59	25-34
t'_1	53-59	25-34
t'_2	28-39	41-59
t'_3	28-41	20-59
t'_4	40-59	20-35
t'_5	28-39	41-59
t'_6	39-41	20-47
t'_7	40-57	27-35

Original	Matches	Anon/zed	Matches
t_0	t'_0, t'_1, t'_4	t'_0	t_0, t_1, t_7
t_1	t'_0, t'_1, t'_7	t'_1	t_0, t_1, t_7
t_2	t'_2, t'_5, t'_6	t'_2	t_2, t_3, t_5
t_3	t'_2, t'_3, t'_5	t'_3	t_3, t_4, t_5
t_4	t'_3, t'_4, t'_6	t'_4	t_0, t_4, t_6
t_5	t'_2, t'_3, t'_5	t'_5	t_2, t_3, t_5
t_6	t'_4, t'_6, t'_7	t'_6	t_2, t_4, t_6
t_7	t'_0, t'_1, t'_7	t'_7	t_1, t_6, t_7

Table 1 illustrates a concrete example of heterogenous generalization. The top table presents the values of eight tuples on *QI* attributes *Age* and *Salary*. Heterogeneous generalization may anonymize these tuples as in the bottom left table; each tuple is recast to a *range* of values, so as to be compatible with, or *match*, three original tuples, and vice versa, as the bottom right table shows. Past work [16, 56]; has used heterogeneous generalization with k-anonymity [48], which only protects against re-identification. An attempt to use heterogeneous generalization to achieve a guarantee on the disclosure of the values of a sensitive attribute [56] followed a fixed generalization motif and abode by ℓ-diversity [43]. Yet ℓ-diversity allows an *arbitrary* increase of an adversary's confidence in some tuple's *SA* value (SV) after publication. A robust alternative, β-likeness [11], bounds the *growth* of an adversary's confidence in each tuple's *SA* value caused by publication, yet has only been applied with homogeneous generalization.

In this paper, we design *heterogeneous generalization* schemes that bound the posterior to prior *confidence gain* in sensitive values by β-likeness [11], hence resist attacks exploiting data correlations, and also yield high utility; to do so, we first develop a baseline solution for ℓ-diversity [43]. We use a randomization scheme that resists attacks reverse-engineering the algorithm [55], following [16, 56, 59]. Our experiments show that the proposed solutions outperform previous syntactic anonymization schemes in data utility with a small runtime overhead that is reducible by data-parallel execution. Moreover, β-likeness with heterogeneous generalization surpasses a state-of-the-art DP-based solution in terms of *both* utility in a practical data analysis task *and* resistance to a practical learning-based attack on census data.

2 Background and Related Work

Syntactic anonymization reduces the *precision* of data, but not its *accuracy*: the published, *recast* data is truthful, albeit imprecise [10, 38]. The earliest syntactic privacy model, k-anonymity [48], requires that each original record r_i has *at least* k equally probable *matches* among recast ones; a match of r_i is a record r'_j whose imprecise *QI* values agree with those of r_i, hence may be the true representative of r_i. These relationships form a bipartite ***generalization graph*** [59].

Enhancing Privacy. The k-anonymity model does not prevent the inference of non-*QI sensitive* attribute (*SA*) values (SVs); the ℓ-diversity model [21, 43] amends k-anonymity, postulating that each record be matched to at least ℓ *well-represented* SVs, yet provides insufficient privacy when the SV *distribution* among a record's plausible identities differs substantially from the global distribution [40]. However, ℓ-diversity treats all SVs equally, regardless of the adversary's prior confidence in them arising from their overall distribution. To fix this problem, β-likeness [11] requires that an adversary's confidence in an SV does not increase, due to the publication of generalized *QI*s, by more than a relative-increase threshold β, while preventing confidence bounds exceeding 1.

Abandoning Homogeneity. Early syntactic anonymization schemes [11, 21, 43, 44] imposed a Procrustean *homogeneity requirement* [10], forcing data in groups of identical *QI* values. Later research waived this constraint, first trying a solution that forfeited equiprobable associations and caused excessive

information loss [24, 50], then a *fixed-motif* non-homogeneous form [56, 59], and eventually fully heterogeneous generalization [16], which was applied only with k-anonymity.

To guarantee equiprobable associations, it is necessary and sufficient that the *generalization graph*, as the one in Figure 1(b), be k*-regular* [16], i.e., confers each original record exactly k matches among published ones and vice versa [56]. A k-regular generalization graph induces several sets of k disjoint assignments, each representing a possible one-to-one correspondence between original and recast records [16, 56]. To resist attacks based on knowledge of the algorithm, we generate a random set of k disjoint assignments and choose one of those *uniformly at random* [16, 56, 59]. For self-containment, we reiterate the argument in [56, 59].

Lemma 1. *In a digraph G, where each vertex v has equal indegree and outdegree, d_v, each edge belongs to a cycle.*

Proof. Assume an edge $e = (u \rightarrow v)$ does not belong to a cycle, hence there is no path from v to u. We classify vertices to which there is a path from v into group X, with $v \in X$. Vertices in X have the same total number of incoming and outgoing edges, $\sum_{w \in X} d_w$. Each outgoing edge from X must point to another vertex in X, since there is no path from any vertex in X to u. As one incoming edge to X, namely e, originates outside X, there are at most $\sum_{w \in X} d_w - 1$ incoming edges from X to X, contradicting the pigeonhole principle. □

Lemma 2. *In an ℓ-regular directed graph G, each edge belongs to a perfect matching.*

Proof. Consider any edge e. By Lemma 1, e belongs to a cycle C. If C contains all vertices of G, it is a perfect matching itself. If not, we rewire G to G' as follows: for each edge $e = (u \rightarrow v)$ in C, we substitute u by a new vertex u/v inheriting the outgoing edges of u and the incoming edges of v; edges in C become self-loops in G'. Since G is ℓ-regular, G' is ℓ-regular too. We say that all vertices in C have been *matched*. We select an arbitrary unmatched vertex $w \in G'$ and one of its outgoing edges e' at random, and repeat the process to find a cycle C'. If C' contains a previously matched vertex w', we update the matches of w' according to C'. As at least one new vertex gets matched in each iteration, the process leads to a perfect matching. □

Theorem 1. *We can generate a set of ℓ disjoint assignments (i.e., perfect matchings) from an ℓ-regular directed graph G.*

Proof. We pick a node n_i and an outgoing edge e randomly. By Lemma 2, e belongs to an assignment a. We remove the edges in a from G, to get graph G'. As each node in G' has exactly $\ell-1$ incoming/outgoing edges, G' is $(\ell-1)$-regular. We repeat iteratively to find ℓ disjoint assignments. □

We aim to achieve β*-likeness by heterogeneous generalization* for the sake of data utility. As a preliminary step, we examine the ℓ-diversity case, then proceed to β-likeness. We assume an attacker having no other prior knowledge apart from the global SV distribution and any individual *QI* values.

Measuring Utility. Our schemes target a general-purpose utility measure based on the extent of generalization ranges, the Global Certainty Penalty (GCP), that has been extensively used in previous works [4, 5, 6, 7, 11, 12, 16, 21, 25, 26, 27, 28, 29, 32, 39, 41, 46, 53, 54, 56, 58, 63]. For a numerical attribute A_j, the Normalized Certainty Penalty (NCP), for a recast tuple q'_i is $NCP_j(q'_i) = \left(u_i^j - l_i^j\right)/(U^j - L^j)$, where u_i^j (l_i^j) is the largest (smallest) value of attribute A_j among the matches of q'_i, and U^j (L^j) is the largest (smallest) value in the domain of A_j. If A_j is categorical, then $NCP_j(q'_i) = {}^{(count_j(q'_i)-1)}/_{(|A_j|-1)}$ for $count_j(q'_i)$ distinct values in $\mathcal{V}(q_i^j)$, and $|A_j|$ the domain cardinality of A_j. The GCP of a set of recast tuples Q' with d attributes is $GCP(Q') = {}^{\sum_{q'_i \in Q'} \sum_j NCP_j(q'_i)}/_{d \cdot |Q'|}$. In Section 5 we also consider an application-oriented utility measure, accuracy in conjunctive count queries. These utility measures reflect the value imprecision introduced for the sake of privacy.

3 Heterogeneous ℓ-diversity

Consider a dataset $\mathcal{D} = (Q, S)$ of n tuples. $Q = \{q_1, \ldots, q_n\}$, where q_i is the *QI* of tuple i and $S = \{s_1, \ldots, s_n\}$, where s_i is the SV of tuple i. We have to *recast* $\mathcal{D}$ to a sanitized form $\mathcal{D}' = (Q', S')$, recasting the value of each $q_i \in Q$ on a *QI* attribute A_j, q_{ij}, as a *set* of values $\mathcal{V}(q_{ij}) = q'_{ij} \in Q'$ [11, 21, 38, 48], and each SV $s_i \in S$ by an SV $s'_i \in S'$. We publish the *range* of values in q_{ij} for numerical attributes, and the set itself for categorical attributes. A tuple q_i and a recast tuple q_k *match* each other when each q_{ij} is included in $\mathcal{V}(q_{kj})$.

Definition 1 (Threat model)**.** *We consider an adversary who knows the overall SV distribution in S, identical to that in S', and all QI values, and aims to infer the SV of any individual.*

Definition 2 (ℓ-diversity)**.** *A* recast *data set $\mathcal{D}' = (Q', S')$ satisfies* ℓ-diversity *wrt $\mathcal{D} = (Q, S)$ iff each $q_i \in Q$ matches* at least *ℓ records in Q', having, to an adversary, equal probability, at most ${}^1/_\ell$, to be the* true match *of q_i and different SVs from each other, yet including the SV of q_i.*

Thus, an adversary knowing all *QI*s in Q can identify the true match and SV of any $q_i \in Q$ with probability at most ${}^1/_\ell$.

Definition 3. *Given a data set $\mathcal{D} = (Q, S)$ and its recast version, $\mathcal{D}' = (Q', S')$, an* assignment *from $\mathcal{D}$ to $\mathcal{D}'$ is an one-to-one mapping such that each $q_i \in Q$ is matched to exactly one $q'_j \in Q'$; their SVs, $s_i \in S$ and $s'_i \in S'$, may or may not coincide.*

To achieve ℓ-diversity, i.e., ℓ equiprobable matches for each tuple, it suffices to use an ℓ-regular generalization graph, ensuring that each record $q_i \in Q$ has ℓ distinct matches among the anonymized ones in Q', each with a unique SV, yet including the SV of q, and each anonymized record $q'_i \in Q'$ has ℓ distinct matches among the ones in Q [16, 56]. Such a graph contains several sets of ℓ disjoint assignments from Q to Q'. The set of possible values of each $q'_i \in Q'$ on an attribute A_j, $\mathcal{V}(q_{ij})$, is the set of values of its ℓ matches. To assign SVs to records

in Q', we regenerate a set of ℓ disjoint assignments, select one of them *uniformly at random*, and use it to carry SVs from tuples in $\mathcal{D}$ to $\mathcal{D}'$; randomness ensures equiprobability [16, 56]. We aim to build an ℓ-regular generalization graph that incurs low information loss in terms of GCP.

Problem 1. *Transform a data set $\mathcal{D} = (Q, S)$ to a form $\mathcal{D}'$ that satisfies ℓ-diversity and minimizes $GCP(Q')$.*

Table 2: Notations

Symbol	Meaning
$\mathcal{D}$	dataset (collection of tuples)
n	the number of tuples in $\mathcal{D}$
QI	Set of Quasi-Identifiers
SA	Sensitive Attribute
SV	set of SA values
$sup(s)$	support of the sensitive value $s \in SV$
$\mathcal{B}$	set of buckets
B_i	the i^{th} bucket, $B_i \in \mathcal{B}$
$temp$	set of tuples having the same SA value
C	set of candidate buckets to place $temp$ in
A_t^x	tuple-to-tuple assignment at step $x \in [0, \ell]$
G_j	set of matches for the j^{th} tuple
$\mathcal{GA}$	set of global assignments

For ℓ-diversification to work, the input data should be ℓ-eligible [21, 43], i.e., the frequency of any SV should be *at most* $\lfloor n/\ell \rfloor$. As we build our ℓ-regular generalization graph, if we assign too many low-frequency SVs as matches to one tuple q_i, we may run out of available low-frequency matches to assign to another tuple q_j so as to ensure diversity.

For example, consider six tuples with SVs $\{a, a, b, b, c, d\}$. As the highest SV frequency is $1/3$, the data is 3-eligible [43], i.e., we can extract 3 assignments such that each tuple gets 3 matches of different SV. We write an assignment as a *permutation* of the six SVs. Let the first assignment be a self-assignment: $[a, a, b, b, c, d]$. Assume the second assignment is $[b, b, a, a, d, c]$. Given these two assignments, we try to extract a third, so that each tuple (position) gets an SV different from the two it already got. Unfortunately, we cannot: each of the four first tuples can only accept a match with SV c or d, but there are only two tuples of such SVs. This problem is due to assigning both low-frequency SVs c and d to each of the last two positions in the first two assignments. Thus, we should avoid using too many low-frequency values as matches for the same tuple. Such a problem does not arise in case all SVs have the *same* frequency, n/ℓ. In this case, we can group tuples in ℓ buckets, one for each SV, and assign to each tuple one match from each bucket. We *convert* all problem cases to variants of this case, by creating ℓ buckets *even if* these do not perfectly correspond to single SVs; more frequent SVs may share a bucket with less frequent ones. After *bucketization*, in an *assignment extraction* stage, we assign to each tuple exactly one match from each bucket. Table 2 lists our notations.

3.1 Bucketization

Without loss[2] of generality, we consider that $n \mod \ell = 0$. We create a set of ℓ **buckets**, each containing n/ℓ records as Algorithm 1 describes, provided that the ℓ-eligibility condition [43] holds (Lines 9–12). We first place each set of tuples having one of the top-ℓ most frequent SVs, $v_1, v_2, \ldots, v_\ell$, in its own bucket, $B_1, \ldots, B_\ell$, respectively (Lines 15–16). We fill up the rest available bucket space with tuples of less frequent SVs greedily (Lines 18–30): we obtain the set of tuples temp of the next most frequent SV s_i (Line 19), and assign *randomly* as many as fit in the *most empty* bucket $\hat{b}$ (Lines 22–30), and move on to the next most-empty bucket; thereby, tuples of the same SV may be shared among buckets. In Section 5, we examine an alternative way of filling buckets by information loss considerations, yet abandon it as it brings negligible utility benefits, while it may incur leakage. Allocating tuples to buckets takes $O(n + m \log m)$ time, for m distinct SVs.

ALGORITHM 1: Bucketization for ℓ-diversity

```
Data: set of tuples D; sensitive attribute SA; ℓ parameter
Result: set of ℓ buckets B = {B_1, B_2, ..., B_ℓ}
1  B := ∅;
2  for i = 1, ..., ℓ do
3      B_i := ∅; // Initialization.
4      B := B ∪ {B_i};
5  SV := {s | s = t[SA], t ∈ D}; // Set of all sensitive values
6  for s ∈ SV do
7      sup(s) := |{t ∈ D : t[SA] = s}|; // Support of sensitive value s
8  // Check ℓ-eligibility:
9  if |SV| < ℓ then
10     return ∅; // There are less than ℓ sensitive values
11 if max_{s∈SV} {sup(s)} > n/ℓ then
12     return ∅; // max SA support is larger than bucket size.
13 bucketSort tuples by SV, quickSort SVs by support;
14 // place tuples of ℓ most frequent SVs in ℓ buckets:
15 for i = 1, ..., ℓ do
16     B_i := {t ∈ D : t[SA] = s_i};
17 // remaining SA values:
18 for i = ℓ+1, ..., |SV| do
19     temp := {t ∈ D : t[SA] = s_i};
20     C := B \ {b ∈ B : |b| = n/ℓ}; // exclude full buckets
21     while temp ≠ ∅ do
22         b̂ := arg min_{b∈C} {|b|}; // the most empty bucket
23         C := C \ {b̂};
24         if |b̂| + |temp| ≤ n/ℓ then
25             //temp fits in the remaining space of b̂
26             b̂ := b̂ ∪ temp;
27         else
28             while |b̂| < n/ℓ do
29                 b̂ := b̂ ∪ {t ∈ temp};
30                 temp := temp \ {t};
31 return B;
```

3.2 Assignment Extraction

Next, by Algorithm 2, we extract $\ell - 1$ disjoint assignments among tuples in our ℓ buckets (*plus self-assignments*), so that each tuple gets a match from a different bucket in each round. As tuples of the same low-frequency SV may be shared among buckets, we avoid matching any tuple to more than one tuple of the same SV. In Section 3.4 we prove this is always possible. We first match each bucket B_i to all others (Line 4), and then expand each bucket-to-bucket match to n/ℓ tuple-to-tuple matches. For each pair of matched buckets $(B_b, B_{b'})$ (Line 2, Line 6), we generate a *local assignment* comprising matches from a tuple in B_b to one in $B_{b'}$ (Lines 9–11). The **Match** function (Line 9) seeks tuple-to-tuple matches that incur low information loss in NCP, hence GCP. To solve this informa-

[2] In case n is not divisible by ℓ, we insert $o = \ell - (n \mod \ell)$ dummy tuples, each with a unique SV.

tion loss minimization subproblem we use either the $O(n^3)$ Hungarian algorithm [18, 37, 51], or one of the heuristics in [16], $O(n^2)$ `Greedy` or $O(n^2 \log n)$ `SortGreedy`). Before each assignment, we update an NCP cost matrix (Line 8) to reflect the running state of affairs. Eventually, we obtain a set $\mathcal{GA}$ of ℓ disjoint *tuple-to-tuple assignments*, which determines generalized QI values. To defend against adversaries who know the algorithm, as in [16, 56, 59], we extract anew a random set of ℓ disjoint *tuple-to-tuple assignments* and pick one of those uniformly at random to allot SVs (Line 12).

ALGORITHM 2: Assignment Extraction

```
Data: set of ℓ buckets ℬ = {B_1, B_2, ..., B_ℓ}
Result: Set of ℓ tuple-to-tuple assignments 𝒢𝒜*
1  𝒢𝒜 := ∅;
2  for b = 1 to ℓ do
3      for i = 0 to n/ℓ do
4          G_{(b·n/ℓ)+i} := {B_b(i)}; // Phase A: bucket-to-bucket assignment
5          𝒢𝒜 := 𝒢𝒜 ∪ G_{(b·n/ℓ)+i};
6      for b' ∈ {b+1, ..., ℓ} ∪ {1, ..., b−1} and x ∈ [2, ℓ] do
7          for i = 0 to n/ℓ and j = 0 to n/ℓ do
8              Cost_{i,j} := NCP(G_{(b·n/ℓ)+i}, B_{b'}(j)); // cost matrix
9          A_t^x := Match(Cost, B_b, B_{b'}); // Phase B: tuple-to-tuple assignment
10         for i = 0 to n/ℓ do
11             G_{(b·n/ℓ)+i} := G_{(b·n/ℓ)+i} ∪ {B_{b'}(A_t^x(i))}; // update
12 𝒢𝒜* := Randomize(𝒢𝒜);
13 return 𝒢𝒜*;
```

3.3 Example

Consider 15 tuples of SVs $[a,a,a,b,b,b,c,c,c,d,d,e,e,f,f]$. As each SV has frequency at most $1/5$, the data is 5-eligible. If we try to extract 5 disjoint tuple-to-tuple assignments, shown as permutations of SVs (i.e., each match is shown at the respective position), so that no SV is repeated in the same position, we may encounter a deadend, i.e., a state from which we cannot progress. Table 3 shows such a case.

Table 3: Extracted assignments leading to deadend

Assignment 1	a a a	b b b	c c c	d d	e e	f f
Assignment 2	b b b	c c c	a a a	e e	f f	d d
Assignment 3	c c c	a a a	b b b	f f	d d	e e

After extracting these assignments, each of the first nine tuples should match to one of the six tuples with low-frequency SVs, d, e, or f. As there are only six such tuples, a deadend emerges. To avoid this, we create five buckets as in Table 4, and allow a tuple to get only *one* match from each bucket.

We share tuples of SV f among buckets B_4 and B_5, hence all six lower-frequency SVs are found in those two buckets only. Thus, each tuple can get at most two of the three low-frequency SVs, $\{d,e,f\}$, as matches in five assignments; the other three matches of each tuple are reserved for the high-frequency SVs $\{a,b,c\}$, resulting in good balance.

Table 4: Five Buckets

Bucket	Tuple IDs	SVs
B_1	1, 2, 3	a a a
B_2	4, 5, 6	b b b
B_3	7, 8, 9	c c c
B_4	10, 11, 14	d d f
B_5	12, 13, 15	e e f

Still, for bucket-based assignment extraction to work, we should ensure that no tuple is matched to both tuples with SV f in buckets B_4 and B_5. We work in two phases. In Phase A, we extract five bucket-to-bucket assignments, shown in Table 5 via a permutation of the bucket order.

Table 5: Five bucket-to-bucket assignments

B-to-B Assignment 1	$B_1\ B_2\ B_3\ B_4\ B_5$
B-to-B Assignment 2	$B_2\ B_3\ B_4\ B_5\ B_1$
B-to-B Assignment 3	$B_3\ B_4\ B_5\ B_1\ B_2$
B-to-B Assignment 4	$B_4\ B_5\ B_1\ B_2\ B_3$
B-to-B Assignment 5	$B_5\ B_1\ B_2\ B_3\ B_4$

In Phase B, we concretize each bucket-to-bucket match to a collection of tuple-to-tuple matches as in Table 6.

Table 6: Five tuple-to-tuple assignments

T-to-T Assignment 1	a a a	b b b	c c c	d d f	f e e
T-to-T Assignment 2	b b b	c c c	d d f	f e e	a a a
T-to-T Assignment 3	c c c	d d f	f e e	a a a	b b b
T-to-T Assignment 4	d d f	f e e	a a a	b b b	c c c
T-to-T Assignment 5	f e e	a a a	b b b	c c c	d d f

Each of 15 tuples obtains five different SVs, namely $\{a,b,c,d,e\}$, $\{a,b,c,d,f\}$, or $\{a,b,c,e,f\}$. Comparing Table 5 to Table 6, we see how bucket-to-bucket assignments guide tuple-to-tuple assignments; the position of tuples with SV f in buckets B_4 and B_5 varies to avoid conflicts.

3.4 Proof of Soundness

To lead to a sound result, our algorithm should always find a local tuple-to-tuple assignment, without ever assigning more than one match of the same SV to the same tuple. The following theorem shows that this is possible.

Theorem 2. *In the t^{th} round of local assignment extraction, $t \le \ell$, with bucket B_i matched to a set of buckets $\mathcal{M}(B_i) = \{B'^{i}_1, \ldots, B'^{i}_{t-1}\}$ in the previous $t-1$ rounds, we can extract a new local tuple-to-tuple assignment between tuples in B_i and those in any other bucket B'_j matched to B_i, without assigning more than one match of the same SV to the same tuple.*

Proof. Our bucketization (Section 3.1) groups tuples by SVs and assigns those of the i^{th} most frequent SV to the i^{th} bucket, $i \le \ell$. In general, such buckets have available space to be filled with tuples of other, less frequent SVs. However, each bucket B_i is characterized by the i^{th} most frequent SV, of which it holds the most tuples. We call this SV the *dominating* SV of B_i and denote it as SV_i. A tuple having the *dominating SV* is called a *dominating tuple* in B_i. Let s_i denote the number of dominating tuples in B_i. Then $b \ge s_1 \ge s_2 \ge \ldots \ge s_\ell$, where $b = n/\ell$ is the bucket size.

We need to prove that an assignment from B_i to B'_j can be extracted without using any match (i.e., edge) from a tuple $r \in B_i$ to a tuple $r' \in B'_j$ such that r has been matched to the SV of r' in a previous tuple-to-tuple assignment. We show that a bipartite graph built of all *allowed* matches from tuples in B_i to those in B'_j allows for an assignment to be extracted.

By design, a bucket's dominating SV appears only in itself. Thus, each of the s_j dominating tuples in B'_j has no SV conflict with any tuple in B_i, and therefore receives b incoming edges from B_i. In effect, the s_j dominating tuples in B'_j have **incoming degree** b. Assume B'_j contains non-dominating

tuples of p distinct SVs, $sv_1, \ldots, sv_p$. Let C_q be the (non-zero) number of non-dominating tuples with sv_q in B'_j, and K_q be the (possibly 0) number of records with sv_q in the set of buckets $\mathcal{M}(B_i)$ that B_i, and hence some tuple therein, has already been matched to. Recall that, by construction, SVs of non-dominating tuples cannot have higher overall frequency than SVs of dominating tuples, therefore:

$$C_q + K_q \leq s_j \quad 1 \leq q \leq p \tag{1}$$

By assumption, each of the K_q tuples with SV sv_q has been previously matched to a tuple in B_i; then each of the C_q tuples with SV sv_q in B'_j may only have $d = b - K_q$ incoming edges. Then Equation (1) yields:

$$d_r \geq b - s_j + C_q > b - s_j \quad 1 \leq q \leq p \tag{2}$$

Still, $b - s_j$ is the number of *all* non-dominating tuples in B'_j. Each of these has **incoming degree** *larger than their number*. By Hall's marriage theorem, a bipartite graph $G = (X \cup Y, E)$ has a perfect matching from X to Y *iff* for every subset $W \subset X$, $|W| \leq |N_G(W)|$, where $N_G(W)$ is the neighborhood of W, i.e., the set of all vertices in Y adjacent to a vertex in W. Let B'_j be X and B_i be Y, $|X| = |Y| = b$. A subset $W \subset X$ that contains any of the dominating tuples in X has $|N_G(W)| = b \geq |W|$; moreover, by the preceding analysis, a subset $W \subset X$ that contains exclusively non-dominating records in X has $|N_G(W)| > b - s_j \geq |W|$. It follows that the graph of *allowed matches* from B_i to B'_j always contains a perfect matching, i.e., allows for extracting a *local assignment*. □

3.5 Proof of ℓ-diversity

Our technique inherits the properties of heterogeneous generalization approaches that achieve k-anonymity [16, 56, 59] as well as ℓ-diversity via partition merging [56]. Our data recasting forms an ℓ-regular generalization graph: each original record r_i has exactly ℓ matches in the published data R' and vice versa. Given such an ℓ-regular generalization graph, we can extract one or more sets of ℓ disjoint assignments via random walks [56, 59] as Theorem 1 shows.

Theorem 3. *Consider a* recast *data set $\mathcal{D}' = (Q', S')$, produced from $\mathcal{D} = (Q, S)$ by Algorithm 1 for bucketization and Algorithm 2 for extracting assignments using those buckets, and an adversary who knows the overall SV distribution in S, identical to that in S', and all values in Q. $\mathcal{D}'$ satisfies ℓ-diversity, i.e., the adversary observes* at least *ℓ records in Q' matching each $q_i \in Q$, having distinct SVs that include the SV of q_i, and has equal confidence, at most $1/\ell$, that each of those be the* true match *of q_i.*

Proof. To an adversary, each of such ℓ assignments represents a *possible world*. We assign SVs to published records according to the *putative identities* indicated by one such assignment, which we select *uniformly at random* out of a set of ℓ disjoint assignments extracted from the ℓ-regular generalization graph by *randomization* [56, 59]. This process ensures that each of an original or published record's ℓ matches is equally likely to be chosen; thus, the SV we assign to a published record is equally likely to originate from any of its ℓ original-record matches, and, in reverse, the SV of an original record is equally likely to assigned to any of its ℓ published-record matches. An adversary considers these ℓ options as *equiprobable*; running the same algorithm on the same data yields different results, thanks to randomization [15]. The same anonymized data could be generated by ℓ different data instances, each record r associated with a different SV in each instance. The adversary cannot decide which instance corresponds to the input, hence cannot infer the true SV of r [57]. This *equiprobability* warrants k-anonymity in [16, 56, 59], ℓ-diversity in [56], and ℓ-diversity here: original record's matches provide a set of ℓ distinct possible sensitive values, whose equiprobability warrants ℓ-diversity, which limits the adversary's confidence in any *SA* value to $1/\ell$ due to Equation 5 in [43]. □

Our scheme comprises: (i) *privacy-oriented bucketization* that assures, with randomization, the equiprobability of each tuple's matches, and (ii) *utility-oriented assignment extraction* that selects specific matches. As in [16, 56, 59], no decisions consider both privacy and utility. As argued in [57], this *noninterference* between privacy and utility confers immunity to the *minimality attack* [55]. We revisit this matter in Section 6.

4 Heterogeneous β-likeness

Here, we inherit the assumptions regarding the adversary and the *bucketization-and-assignment-extraction* workflow of our solution for ℓ-diversity, and adapt it to β-likeness [11]:

Definition 4 (β-likeness)**.** *An* anonymized *data set $\mathcal{D}' = (Q', S')$ satisfies* β-likeness *with respect to the original data $\mathcal{D} = (Q, S)$ iff the confidence of an adversary regarding the sensitive value s_i of any original record $q_i \in Q$ does not increase, in relative terms, by more than a threshold $\beta \geq 0$ upon the publication of $\mathcal{D}'$, i.e., $(\psi_i - \phi_i)/\phi_i \leq \beta$, where ψ_i is the probability of any putative SV s'_i measured as its frequency in the* set of matches *in $\mathcal{D}'$ assigned to q_i, and ϕ_i is the frequency of s'_i in the overall sensitive value distribution.*

In effect, β-likeness bounds the *increase* in an adversary's confidence[3] on the likelihood of any sensitive value:

$$\frac{\psi_i - \phi_i}{\phi_i} \leq \beta \Leftrightarrow \psi_i \leq (1+\beta)\phi_i \ \forall i \tag{3}$$

We thus address the following problem:

Problem 2. *Transform a data set $\mathcal{D} = (Q, S)$ to a form $\mathcal{D}'$ that satisfies β-likeness and minimizes $GCP(Q')$.*

First, we consider whether β-likeness may be achieved via ℓ-diversity. We find that this is the case under conditions.

Theorem 4. *For $\beta \geq \phi_M/\phi_m - 1$, β-likeness on dataset $\mathcal{D}$ is achieved by ℓ-diversity with $\ell \geq 1/(1+\beta)\phi_m$, where ϕ_M is the maximum and ϕ_m minimum frequency of an SV in $\mathcal{D}$.*

[3] If confidence *drop* is a concern, our schemes can be extended to bound it.

Proof. By ℓ-diversity, any SV has probability at most $1/\ell$. This bound satisfies β-likeness iff it covers the least frequent SV, i.e., $1/\ell \leq (1+\beta)\phi_m$. By the pigeonhole principle, to achieve ℓ-diversity, the maximum SV frequency should be at most $1/\ell$, i.e., $\phi_M \leq 1/\ell \Rightarrow \phi_M \leq (1+\beta)\phi_m$, hence $\beta \geq \phi_M/\phi_m - 1$. □

By Theorem 4, if $\beta \geq \phi_M/\phi_m - 1$, we can achieve β-likeness via ℓ-diversification. Otherwise, we use a new bucketization scheme for β-likeness, along with other elements of our methodology: Once we define μ buckets, we extract assignments by Algorithm 2 to define a low-GCP generalization, craft a set of μ disjoint assignments, and select one of them uniformly at random to allot SVs; randomization thwarts adversaries who know the algorithm [16, 56, 59]. Henceforward, we discuss this new bucketization scheme for β-likeness.

4.1 Delimiting the Number of Buckets

Let $\mathcal{B}_i$ denote the set of buckets that include *at least one* tuple with SV v_i, and $\mathcal{B}$ the set of all μ buckets. The frequency of SV v_i within a tuple's *set of matches* is bounded via $\mathcal{B}_i$:

$$\psi_i \leq \frac{|\mathcal{B}_i|}{|\mathcal{B}|} \tag{4}$$

Here, we need to construct buckets in a way that leads to assignments satisfying β-likeness. Due to Equation (4), to achieve the condition of Equation (3), it suffices that:

$$\begin{aligned} & |\mathcal{B}_i|/|\mathcal{B}| \leq (1+\beta)\cdot\phi_i \\ \Rightarrow \quad & |\mathcal{B}_i| \leq (1+\beta)\cdot\phi_i\cdot|\mathcal{B}|, \ \ \forall i \in SA \end{aligned} \tag{5}$$

The condition of Equation (5) is sufficient (though not necessary) to obtain β-likeness; we call it *strict β-likeness*.

4.2 Purity of Buckets

We proceed by examining the conditions for achieving the strongest case of privacy by β-likeness, 0-likeness.

Definition 5 (Pure bucket)**.** *A* pure *bucket is a bucket that contains only tuples of a single SA value.*

Theorem 5. *Heterogeneous generalization achieves strict* 0*-likeness* iff *all buckets are pure.*

Proof. When all buckets are pure, no bucket contains more than one SA value; thus, adding up the numbers of buckets containing each SA value v_i, we obtain the total number of buckets; in reverse, when $\sum_i |\mathcal{B}_i| = |\mathcal{B}|$, each bucket is counted exactly once in the summation, hence is pure. Thus, all buckets are pure iff $\sum_i |\mathcal{B}_i| = |\mathcal{B}|$. Then it suffices to prove that, by strict β-likeness, $\beta = 0 \Leftrightarrow \sum_i |\mathcal{B}_i| = |\mathcal{B}|$. From Equation (5):

$$\sum_i |\mathcal{B}_i| \leq (1+\beta)\cdot\sum_i \phi_i\cdot|\mathcal{B}| = (1+\beta)\cdot|\mathcal{B}| \tag{6}$$

For β=0, that becomes $\sum_i |\mathcal{B}_i| \leq |\mathcal{B}|$. However, as each bucket contains tuples of at least one SA value, it cannot be $\sum_i |\mathcal{B}_i| < |\mathcal{B}|$; thus, strict 0-likeness effectively implies that $\sum_i |\mathcal{B}_i| = |\mathcal{B}|$. In reverse, when $\sum_i |\mathcal{B}_i| = |\mathcal{B}|$, Equation (6) holds for any $\beta \geq 0$, hence strict 0-likeness is achieved. □

Theorem 5 leads to the following corollary.

Corollary 1. *Heterogeneous generalization achieves strict* 0*-likeness* only if *the bucket size c is a common divisor of the support counts of tuples by SV, $(n_1, \ldots, n_{|SA|})$.*

Proof. Assume bucket size c that is not a common divisor of $(n_1, \ldots, n_{|SA|})$. Then at least one SV v_i has support n_i, s.t. $n_i \bmod c \neq 0$, hence at least one record of v_i is in a non-pure bucket; by Theorem 5, we cannot get strict 0-likeness. □

A smaller c yields more buckets, hence more matches per tuple. A limitation to fewer matches tends to reduce information loss. Thus, lower loss is more likely to be achieved with the largest allowed value of c, the GCD of support counts.

4.3 Delimiting β

As discussed, larger bucket size leads to fewer buckets, hence fewer matches per tuple, hence better utility. If we set bucket size to 1, we achieve highest privacy (0-likeness) but lowest utility; if we use a single bucket of size n, we achieve no privacy, but perfect utility. We follow an intermediate approach. Ideally, a bucket contains all tuples of the same SV. All buckets should be of equal size c; if necessary, we add *dummy tuples* to render the number of tuples n divisible by c; then $\mu = |\mathcal{B}| = n/c$. Let $n_i = \phi_i \cdot n$ be the support of SV v_i. We derive a lower bound on the privacy parameter β from Equation (5):

$$\begin{aligned} & |\mathcal{B}_i| \leq (1+\beta)\cdot\phi_i\cdot|\mathcal{B}|, \ \forall i && \Leftrightarrow \\ & \beta \geq |\mathcal{B}_i|/|\mathcal{B}| \cdot n/n_i - 1, \ \ \forall i && \Leftrightarrow \\ & \beta \geq c/\min_i\{n_i/|\mathcal{B}_i|\} - 1 \end{aligned} \tag{7}$$

Equation (7) provides a lower bound on the β achievable for a bucket size c and an allocation of tuples into buckets. To minimize that bound, we need to maximize $\min_i \{n_i/|\mathcal{B}_i|\}$.

Definition 6 (occupancy)**.** *Given a bucketization $\mathcal{B}$, an SV v_i with support n_i, and the set of buckets $\mathcal{B}_i$ that include tuple(s) with v_i, the* occupancy *of v_i in $\mathcal{B}$ is $occ_i = n_i/|\mathcal{B}_i|$, i.e., the average number of tuples of v_i per bucket.*

Then, minimizing the lower bound on β, i.e., optimizing privacy under bucket size c, raises the following problem:

Problem 3 (occupancy maximization, MaxMinOcc)**.** *Given a database $\mathcal{DB}$ of n tuples, in which each of n_i tuples is associated with SV v_i, $\sum_i n_i = n$, and a bucket size c, such that $n \bmod c = 0$, allocate c tuples to each of $B = n/c$ buckets to form a bucketization $\mathcal{B}$ that maximizes the* minimum occupancy*, $minocc = \min_i \left\{\frac{n_i}{|\mathcal{B}_i|}\right\}$, where $\mathcal{B}_i$ is the set of buckets that contain at least one tuple of SV v_i.*

Occupancy maximization calls for placing tuples of each SV in as few buckets as possible. In case the support n_i of an SV n_i is divisible by c, i.e., $n_i \bmod c = 0$, then the tuples of each such v_i are to be placed in n_i/c buckets with optimal occupancy, c. If $n_i \bmod c = r_i \neq 0$, it would be ideal to place each remainder of $r_i < c$ tuples in one bucket only, along

with tuples of other SVs. If we *avoid splitting* such remainder tuples into more than one buckets, then each SV occupies $\lceil n_i/c \rceil$ buckets, with $minocc = \min_i \{n_i/\lceil n_i/c \rceil\}$.

Theorem 6. MaxMinOcc *is* **NP**-*hard.*

Proof. Consider the decision version of the problem, which asks whether a certain *minocc* value is feasible. This problem is in **NP**, as it takes polynomial time to verify that a given solution achieves a certain occupancy value. Consider any instance of the Partition problem, which is to decide whether we can divide a set of m items, of sizes $\{n_1, n_2, ..., n_m\}$, with $\sum_i n_i = A$, into two partitions S, S', such that $\sum_{i \in S} n_i = \sum_{j \in S'} n_j = A/2$. We transform this input to $n'_i = A + n_i$, $\forall i$, and add m dummy items of size A. The sum of all items is $(2m+1) \cdot A$, while $\min_i n'_i \geq A > \max_i n'_i/2$. We apply an algorithm for MaxMinOcc on the same input, considering the m items as sensitive values and their sizes n_i as support counts, to decide whether there is a bucketization $\mathcal{B}$ into $B = 2$ buckets of size $c = mA + A/2$ that achieves $minocc = \min_i n'_i$. Now, if the count of any item is split among buckets, its occupancy will be at most $\max_i n'_i/2 < A$, hence $minocc < A \leq \min_i n'_i$. Thus, achieving $minocc \geq \min_i n'_i$ requires that no item be split among buckets. In reverse, if each item is fully contained in exactly one bucket, then $minocc = \min_i n'_i$. Thus, we can reach $minocc = \min_i n'_i$ *if and only if* we can divide all items into two buckets of size $A/2$. With bucket size c, exactly m items should be in each bucket to yield the mA contribution; the remaining $A/2$ contribution corresponds to a subset of input items that form a solution to Partition and vice versa. Therefore, an algorithm for MaxMinOcc effectively decides any instance of Partition. Since we performed a polynomial-time transformation, it follows that MaxMinOcc is **NP**-hard. □

4.4 Example

This **NP**-hardness result implies that finding the largest feasible bucket size c, under a privacy bound β, is also **NP**-hard. The following example provides more intuition.

Let $\mathcal{D}$ be a set of 25 tuples, six having SV 'a', seven having SV 'b', and twelve having SV 'd'. The GCD of {6, 7, 12} is 1. Then, to achieve 0-likeness we need to place the tuples in 25 buckets of size 1. With a higher β, we can opt for larger bucket capacity and achieve lower information loss. We may test several bucket sizes, starting from the GCD of $\{n_1, n_2, \ldots\}$, up to the maximum SV support $max\{n_i\}$, to find the largest that satisfies the desired privacy level β, using Equation 7. We show three example bucketizations.

Table 7: Example — $c = 4$

B_1	B_2	B_3	B_4	B_5	B_6	B_7
d	d	d	a	a	b	b
d	d	d	a	a	b	b́
d	d	d	a	b	b	b́
d	d	d	a	b	b	b́

(a) If we set $c = 4$, a good bucketization using dummy tuples of one SV only (b́) is as in Table 7. The minimum β we can achieve is:

$$\beta \geq c/\min_{i \in SV}\{n_i/|\mathcal{B}_i|\} - 1 = 4/\min\{6/2, 7/3, 12/3\} - 1 = 12/7 - 1 \approx \mathbf{0.71}$$

Table 8: Example — $c = 5$

B_1	B_2	B_3	B_4	B_5
a	a	b	d	d
a	b	b	d	d
a	b	b	d	d
a	b	d	d	d
a	b	d	d	d

(b) If we set $c = 5$, there is no need for dummy tuples, as 5 divides n exactly. Table 8 shows the ensuing bucketization. The minimum β we can achieve now is:

$$\beta \geq c/\min_{i \in SV}\{n_i/|\mathcal{B}_i|\} - 1 = 5/\min\{6/2, 7/2, 12/3\} - 1 = 5/3 - 1 \approx \mathbf{0.66}$$

(c) If we set $c = 6$, 'a' and 'd' fill one and two buckets respectively; 'b' fills one bucket, yet one leftover tuple is placed in a fifth bucket; we add 5 dummy tuples to the last bucket, as in Table 9. The minimum β we can achieve is:

$$\beta \geq c/\min_{i \in SV}\{n_i/|\mathcal{B}_i|\} - 1 = 6/\min\{6/1, 7/2, 12/2\} - 1 = 12/7 - 1 \approx \mathbf{0.71}$$

Table 9: Example — $c = 6$

B_1	B_2	B_3	B_4	B_5
a	d	d	b	b
a	d	d	b	b́
a	d	d	b	b́
a	d	d	b	b́
a	d	d	b	b́
a	d	d	b	b́

Interestingly, the minimum attainable privacy parameter β does not worsen monotonically as c grows, since it depends on the distribution of SVs in a complex manner. Thus, for $\beta > 0$, we start from c equal to the support of the most frequent SV, $max\{n_i\}$ (getting basic privacy and high utility), and proceed testing every $c \in \{max\{n_i\}, \ldots, GCD\}$ to find the largest c that yields a bucketization satisfying the given β.

4.5 Algorithm for β-likeness

Algorithm 3 outlines our bucketization for β-likeness; given a datataset $\mathcal{D}$ and the β parameter, it returns a set of buckets which we use to build a heterogeneous generalization. We find a utility-aware bucketization (Lines 17–39) by a heuristic. First, we search for a bucket size c that satisfies the desired privacy level. Starting from $c = \max n_i$, we greedily calculate a distribution of SVs in buckets to obtain occupancy ratios, $n_i/|\mathcal{B}_i|$ (Line 21), until the minimum β' a bucketization allows (Line 22) satisfies the desired β (Line 19). In the worst case, c reaches the GCD of SV supports, whereupon 0-likeness (hence any $\beta > 0$) is attainable.

With bucket size c set, we prepare the tuples for bucket placement, sorting them by decreasing SV support (Line 23) and rearranging them so that groups that fill buckets exactly ($n_i \mod c = 0$) come first (24). By this order, we place tuples in buckets, filling one bucket before moving on (Lines 26–37). If all tuples of one SV fit in the empty space of a bucket B_j, we place them all therein (Lines 29–31). Otherwise, we place in B_j a random subset of tuples that fit therein. After B_j reaches its capacity, we proceed to B_{j+1} (Line 37). We add dummy tuples as needed (Line 38) and return the resulting

bucket set (Line 39). Time complexity is dominated by the cost of scanning tuples for each examined value of c, hence it is $O(n^2)$; Line 21 performs the calculations of Lines 24–37 to obtain the number of buckets $|\mathcal{B}_i|$ that contain at least one tuple of each SV v_i, without placing tuples in buckets. Once buckets are set, we get a low-GCP generalization by Algorithm 2 and allot SVs by choosing an assignment uniformly at random from a random set of disjoint ones.

ALGORITHM 3: Bucketization for β-likeness

```
Data: set of tuples D; sensitive attribute SA; parameter β ≥ 0
Result: set of μ buckets B = {B_1, B_2, ..., B_μ}
1  B := ∅;
2  SV := {s | s = t[SA], t ∈ D}; // Set of all sensitive values
3  for s_i ∈ SV do
4  |   n_i := |{t ∈ D : t[SA] = s_i}|; // Support of sensitive value s
5  c := GreatestCommonDivisor(n_1, n_2, ... n_|SA|);
6  // Strict 0-likeness:
7  if β = 0 then
8  |   μ := n/c; // if c=1, solution is n buckets of size 1
9  |   quicksort tuples in D by their SA value;
10 |   for i ∈ 1, 2, ..., μ do
11 |   |   B_i := ∅; // Initialization.
12 |   |   for j ∈ 1, 2, ..., μ do
13 |   |   |   B_i := B_i ∪ {t_{i·μ+j}};
14 |   |   B := B ∪ {B_i};
15 |   return B;
16 // Otherwise, β > 0:
17 β' := ∞; // initially.
18 c := max_{s_i ∈ SA} {n_i} + 1;
19 while β' > β do
20 |   c := c − 1;
21 |   Calculate |B_i|, ∀i, required to place the tuples in the buckets;
22 |   β' := c / min_i{n_i/|B_i|} − 1; // attainable β for this c
23 quickSort tuples by SV support;
24 re-arrange so that groups with (n_i mod c) = 0 appear first;
25 j = 1; // bucket index
26 for i = 1, ..., |SV| do
27 |   temp:= {t ∈ D : t[SA] = s_i};
28 |   while temp ≠ ∅ do
29 |   |   if |B_j| + |temp| ≤ c then
30 |   |   |   // temp fits in the remaining space of B_j
31 |   |   |   B_j := B_j ∪ temp;
32 |   |   else
33 |   |   |   while |B_j| < c do
34 |   |   |   |   B_j := B_j ∪ {t ∈ temp};
35 |   |   |   |   temp := temp \ {t};
36 |   |   B := B ∪ B_j;
37 |   |   j := j + 1; // next bucket
38 Add c − |B_j| dummy tuples in B_j;
39 return B;
```

4.6 Proof of β-likeness

Theorem 7. *Consider a* recast *data set $\mathcal{D}' = (Q', S')$, produced from $\mathcal{D} = (Q, S)$ by Algorithm 3 for bucketization and Algorithm 2 for extracting assignments using those buckets, and an adversary who knows the overall SV distribution in S, identical to that in S', and all values in Q. $\mathcal{D}'$ satisfies β-likeness with respect to $\mathcal{D}$, i.e., the confidence of the adversary posterior to the publication of $\mathcal{D}'$ on the sensitive value s_i of any $q_i \in Q$ increases, relative to the prior confidence due to the knowledge of the SV distribution in S, by no more than $\beta \geq 0$.*

Proof. Our β-likeness algorithm produces a set $\mathcal{B}$ of μ buckets from which we build an μ-regular generalization graph by Algorithm 2. We generate a set of μ disjoint assignments therefrom by *randomization* [56, 59] and select on of those *uniformly at random* to assign SVs. By the same argument as in the proof of Theorem 3, an adversary considers these μ options as *equiprobable*. As each record's matches provide possible SVs whose distribution heeds β-likeness, this *equiprobability* of matches assures β-likeness. □

As argued in [57], the *noninterference* between privacy and utility confers immunity to the *minimality attack* [55].

5 Experimental Evaluation

We evaluate our algorithms for anonymization by ℓ-diversity and β-likeness against the following benchmarks:

PrivBayes: The state-of-the-art method for data publishing by differential privacy [61], which constructs a Bayesian network that models attribute correlations, injects noise into a set of low-dimensional marginals, constructs an approximate data distribution using these noisy marginals and the Bayesian network, and samples tuples from this distribution to yield a synthetic dataset. Whereas β-likeness bounds the risk incurred by the publication of the whole data set with regard to sensitive attribute values, differential privacy bounds the risk incurred by the *inclusion* of an individual's data in a publishable data set, with regard to *any* information about that individual. These two privacy notions have different goals; the former bounds a specific risk associated with data publication; the latter bounds a holistic risk associated with one's participation in the data. Yet it is interesting to compare the two under settings in which they provide the same bound in terms of confidence increase. We render PrivBayes comparable to β-likeness in that sense, by setting $\epsilon = \ln(1+\beta)$; by this setting, differential privacy bounds, by e^{ϵ} [17], the increase of an adversary's confidence on the output of any function $\mathcal{K}$, hence also on a sensitive attribute's value, incurred by the inclusion of a target individual's data in the published database.

NH: The state-of-the-art ℓ-diversification method by ring generalization [56]. NH partitions tuples, merges partitions until they satisfy ℓ-diversity, and applies ring generalization to each ensuing partition; runtimes include the time for partitioning, building rings, and the randomization scheme.

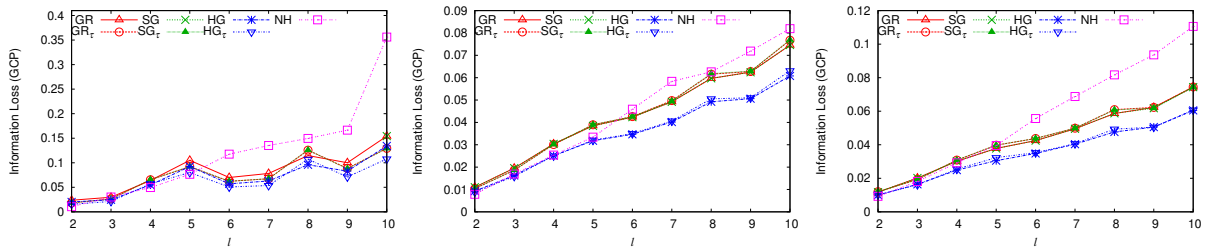
BuReL: The state-of-the-art homogeneous generalization scheme for β-likeness [11].

GR: Our methods employing the $O(n^2)$ `Greedy` algorithm of [16] for assignment extraction. Runtime includes bucketization, assignment extraction, and randomization [56, 59].

SG: Our algorithms that employ the $O(n^2 \log n)$ `SortGreedy` routine [16] for tuple matching, along with bucketization and assignment extraction by randomization.

HG: Our schemes utilizing the $O(n^3)$ Hungarian algorithm [18, 37, 51] to build assignments. HG minimizes NCP per iteration, but that does not translate to optimality across iterations; we include bucketization and randomization.

On ℓ-diversity, we test GR, SG, and HG in the *default* mode and in a *threshold* mode (GR_τ, SG_τ, HG_τ), which places tuples



(a) CENSUS (b) Uniform (c) Zipfian

Figure 2: Information Loss on ℓ-diversity: effect of ℓ.

in buckets having the most distant *QI*; these heuristics may bring utility benefit, ensuring distant tuples are not matched to each other; still, due to a concurrent consideration of privacy and utility, they are vulnerable to the minimality attack [55]; we include them to show the benefit we may gain at the cost of vulnerability [57]. NH is coded[4] in C++; BuReL, GR, SG, HG in Java. All experiments ran on a 24-core Intel Xeon CPU @2.67GHz with 48G RAM running Ubuntu 16.04 LTS.

Table 10: The CENSUS and COIL 2000 datasets

Attribute	Card.	Type
Age	79	num
Gender	2	cat
Education Level	17	num
Marital Status	6	cat
Race	9	cat
Work Class	10	cat
Country	83	cat
Occupation	51	cat

Attribute	Card.	Type
#Houses	10	cat
Household Size	6	cat
Age	6	cat
Customer Type	10	cat
Customer Subtype	41	cat

Data We use real data drawn form the CENSUS [2] and the COIL 2000 [1] datasets; CENSUS contains 500K tuples on 8 attributes, both numerical and categorical, while for COIL2000, which contains ~10K tuples, we maintain the first 5 attributes, all categorical (Table 10). The last attribute is the *SA*, while the rest constitute the *QI*s. To examine the effect of skew in SVs, we generate synthetic datasets of up to 500K tuples and 8 attributes; using the CENSUS data, we maintain the 7 *QI* attributes while assigning an SV drawn from the domain range of the 8th attribute following an (a) uniform or (b) zipfian distribution with various θ values[5].

Parameters. We study behavior with respect to: (a) the privacy parameters ℓ and β; (b) the number of *QI* dimensions d, (c) the partition size p, when allowing for data partitioning; (d) dataset size n; and (e) the θ parameter of zipfian data. The default settings are ℓ=10, d=8, n=10K, θ=0.5, and β=3.

Evaluation Metrics. We measure runtime, information loss (GCP), and accuracy in aggregation queries of this form:

```
SELECT COUNT(*) FROM Anonymized_data
WHERE pred(A_1) AND ... AND pred(A_λ) AND pred(SA)
```

This count query has a conjunctive predicate on λ randomly selected *QI* attributes and the *SA*; `pred(A)` has the form $A \in R_A$, where R_A is an arbitrary interval in the domain of A. We vary λ and selectivity Θ. Assuming uniform distribution, we achieve Θ if each attribute A selects records within a range of length $|A|\cdot\theta_A$, such that $(\theta_A)^{\lambda+1}=\theta$, where $|A|$ is the domain length of attribute A. In effect, the length of R_A should be $|A|\cdot\theta^{\frac{1}{\lambda+1}}$. Given a query, we compute the precise result $prec$ from the original data, and an estimate est from the anonymized data, assuming *uniform* distribution and *proportionality* of results to the intersection between the query and generalized attribute domains. We define $\frac{|est-prec|}{prec}\times 100\%$ as the *relative error*. If $prec$ in a query is 0, we drop that query.

[4]Our code and data are available at `http://github.com/discont`.

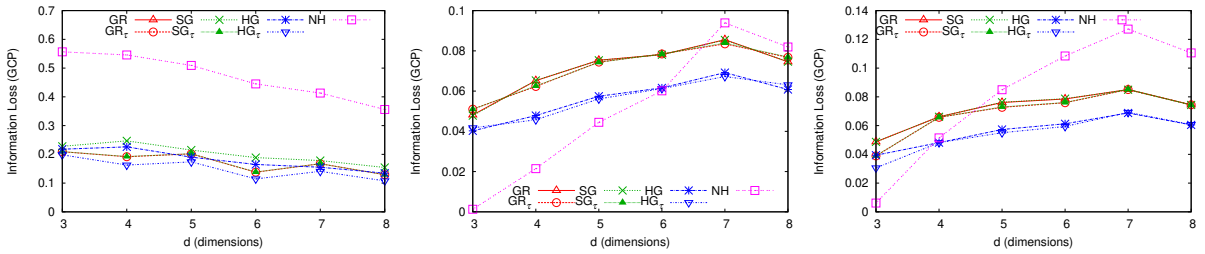
[5]Greater θ corresponds to more biased distributions.

5.1 Evaluation on ℓ-diversity

Effect of ℓ. We run all algorithms over three datasets of 10K tuples: (a) a subset of CENSUS that contains its first 10K tuples, (b) the Uniform dataset thereof and (c) the Zipfian dataset with $\theta=0.5$. Figure 2 shows that the utility gain of our schemes vs. NH increases as ℓ grows from 2 to 10. This gain is more visible with CENSUS and Zipfian, where NH incurs information loss up to 3.5x compared to HG for ℓ=10. This is due to the fact that NH, designed for k-anonymity, is adapted to ℓ-diversification by merely merging partitions, hence yields large partitions that exacerbate information loss [16].

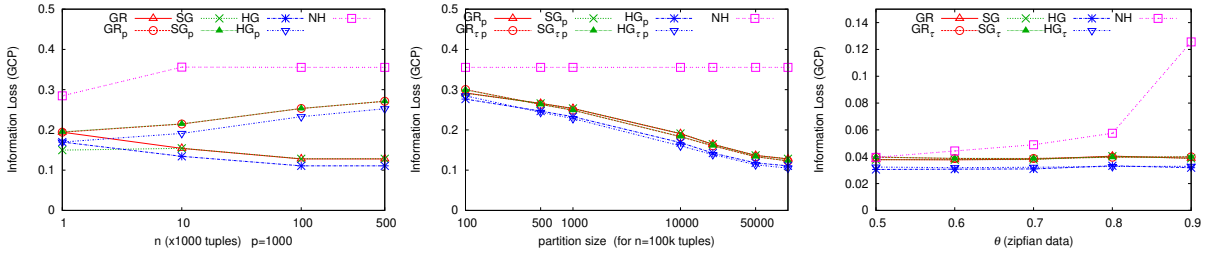
Effect of Dimensionality. We study the effect of *QI* dimensionality d on 10K datasets, with $\ell=10$. Figure 3 shows the results. Our schemes outperform NH in utility for all d values on CENSUS, for $d>3$ on Zipfian, and for $d>6$ on Uniform. As real data, CENSUS is inherently more complex. NH cannot handle this complexity. On CENSUS information loss drops as d grows. Our schemes resist the curse of dimensionality, while NH's homogeneity becomes detrimental.

Effect of Data Size. We investigate scalability on data sets of exponentially growing size, from 1k to 500k tuples, drawn from CENSUS, with d=8. Figures 4(a) and 5(a) present GCP and runtime results for $\ell=10$. Our algorithms consistently outperform NH in terms of GCP, with as much as 3.5x improvement in data utility. The execution of our algorithms in *threshold mode* took prohibitively long, due to the overhead of extra NCP calculations in the bucketization phase, without a significant utility benefit; therefore, we settle for using the regular versions of these algorithms. We also run the *same algorithms* over data partitions of size p (GR_p, SG_p, HG_p), i.e., let them anonymize each partition separately in a data-parallel environment [16]; these variants incur more information loss, yet approach NH in runtime, as Figure 5(a) shows.



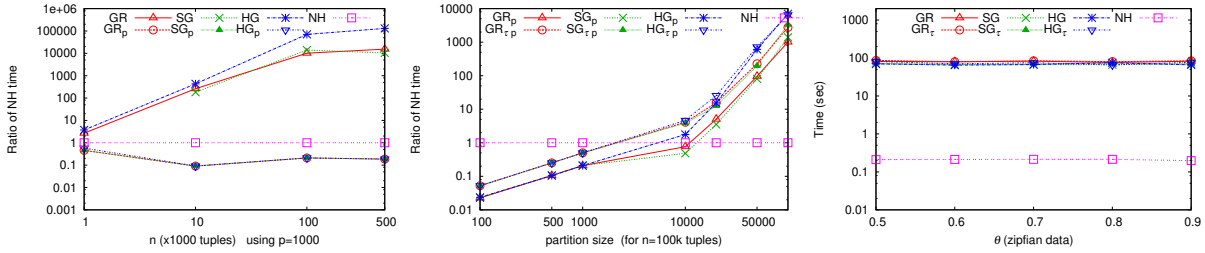
(a) CENSUS (b) Uniform (c) Zipfian

Figure 3: Information Loss on ℓ-diversity: effect of d.



(a) Effect of n [CENSUS] (b) Effect of p [CENSUS] (c) Effect of θ [Zipfian]

Figure 4: Information loss on ℓ-diversity: effect of the parameters n, p and θ.



(a) Effect of n [CENSUS] (b) Effect of p [CENSUS] (c) Effect of θ [Zipfian]

Figure 5: Execution time on ℓ-diversity: effect of the parameters n, p and θ.

Scalability by Partitioning. We study the partition-based variants further. Figures 4(b) and 5(b) plot GCP and runtime for a 100k subset of CENSUS divided in partitions of varying size. NH runs on data partitions by default, but its merging of partitions is non-parallelizable. As partition size drops (leftwards), execution time falls, eventually reaching that of NH, with a modest cost in GCP. Thus, our schemes achieve better utility than NH in equal runtime, while all compared methods achieve the same privacy guarantee.

Effect of Data Skew. We vary θ in the Zipfian data from 0.5 to 0.9, increasing skew. Figure 4(c) shows that our algorithms are resilient, while NH misses quality as θ grows. The utility gain of HG vs. NH rises to 4x for $\theta = 0.9$, while SG and GR follow suit. Figure 5(c) shows runtimes are stable.

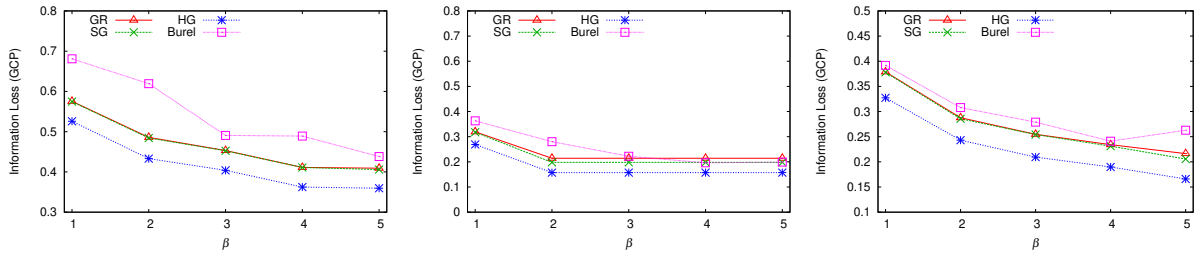
5.2 Evaluation on β-likeness

Effect of β. We study the effect of β over three data sets of size 10K. Figure 6 shows that, as β grows from 1 to 5, privacy becomes laxer, thus all methods afford better utility. HG outperforms BuReL across the board, while SG and GR follow HG with a slight gap; the gain is greater for smaller β, and more evident on CENSUS and Zipfian.

Effect of Data Size. Figure 7(a) shows that the utility gap of our schemes from BuReL grows with data size, on random subsets of CENSUS, with $\beta=3$. Figure 7(b) shows that they need more time, yet we may apply them in a data-parallel manner to obtain scalability, as in Figure 5(b).

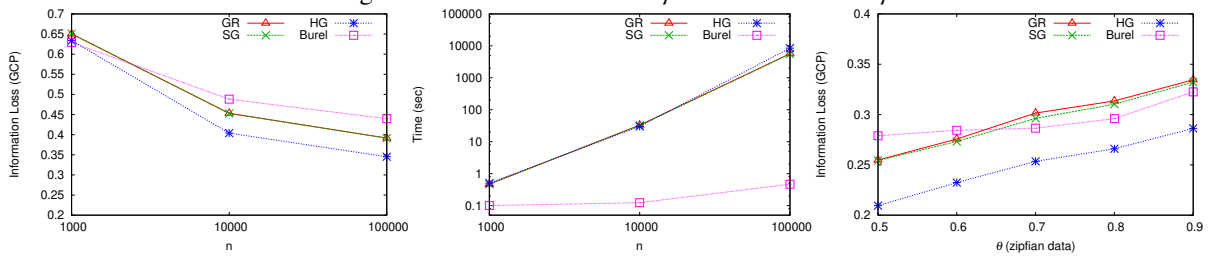
Effect of Data Skew. Figure 7(c) shows that more skewed zipfian distributions (larger θ) cause higher information loss. HG outperforms BuReL, with 33% more utility for θ=0.5.

Range Query Error. To compare against PrivBayes, include the COIL2000 data set, and measure the effect of *dummy tuples*, we report the median relative error of random range count queries, as reported in **Evaluation Metrics**. Figures 8(a) and 9(a) show the effect of β, with query selectivity 10% and dimensionality $\lambda = 3$ QIs, plus the SA. The top of the figures shows the values of $\epsilon = \ln(1+\beta)$ in PrivBayes. Our schemes maintain low relative query error, decreasing with β, outperforming BuReL and PrivBayes by up to 4x for CENSUS and 10x for COIL2000. Figures 8(b) and 9(b) show results for $\beta = 3$ and corresponding $\epsilon = \ln(1+\beta) = 1.386$ for PrivBayes, while varying λ from 1 to 6 for CENSUS and 1 to 4 for COIL2000. Our approaches surpass others by up



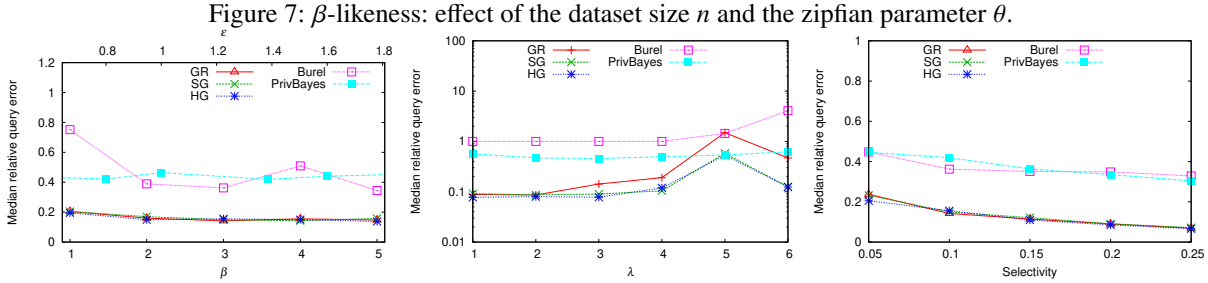
(a) CENSUS (b) Uniform (c) Zipfian

Figure 6: Information Loss on β-likeness: effect of β.



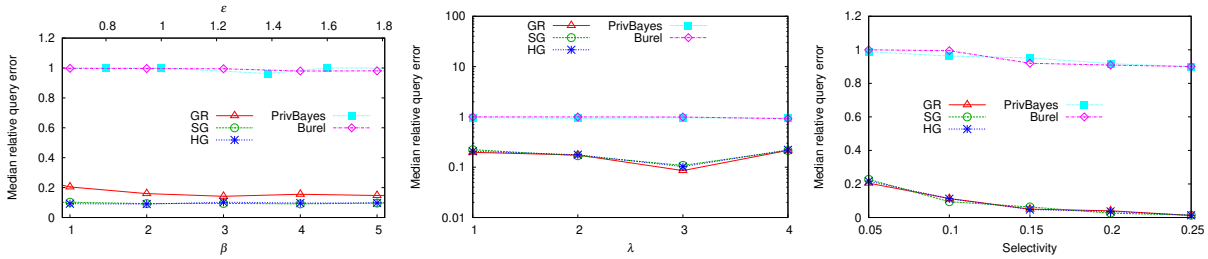
(a) GCP vs. n [CENSUS] (b) Time vs. n [CENSUS] (c) GCP vs. θ [Zipfian]

Figure 7: β-likeness: effect of the dataset size n and the zipfian parameter θ.



(a) Effect of β (b) Effect of λ (c) Effect of Θ

Figure 8: Median query relative error of the CENSUS dataset: effect of β, λ, and selectivity Θ.



(a) Effect of β (b) Effect of λ (c) Effect of Θ

Figure 9: Median query relative error of the COIL2000 dataset: effect of β, λ, and selectivity Θ.

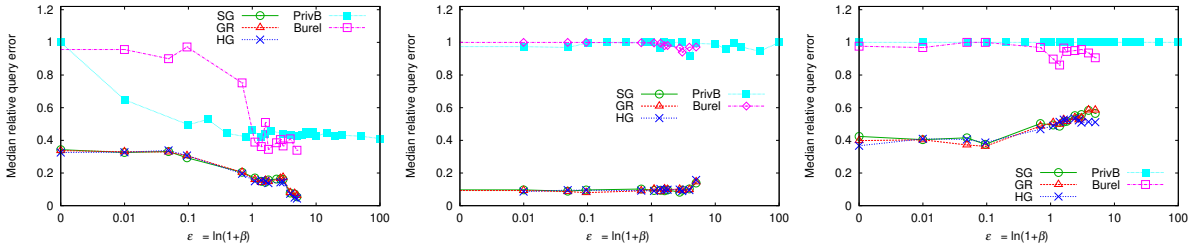
to 83%. Figures 8(c) and 9(c) show results for $\lambda=3$ and $\beta=3$, while varying selectivity Θ from 0.05 to 0.25. The query error of our schemes decreases as selectivity grows. We expand the results of Figures 8(a) and 9(a) to the full range of ϵ in PrivBayes and corresponding β, in Figures 10(a) and 10(b). For CENSUS, when $\beta \geq 49$, we employ ℓ-diversity for the sake of β-likeness, by Theorem 4. Our methods retain an advantage over PrivBayes. We also conduct an experiment using *prefix* ranges $[0,i]$ for the numerical *educational level* attribute and singe-valued conditions on *age*, *gender*, *marital status*, and *race*, as suggested by Census Bureau staff [60]. The results in Figure 10(c) show the advantage over PrivBayes remains.

6 Resistance to Attacks

We now discuss how β-likeness resists adversarial attacks.

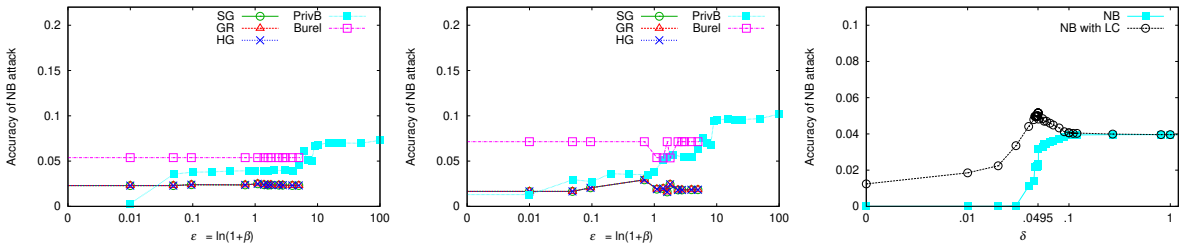
Vulnerability to a *minimality attack* [55] is due to [57]: (1) determinism; (2) unequal groupings; and (3) mingling SVs and *QI*s in decision-making. Our scheme is safe from (1) due to randomization [16, 56, 59]; from (2), as it assigns equal matches to each record; and from (3), as it considers only SVs when bucketizing and only *QI*s when generalizing values.

A *deFinetti attack* learns correlations of *SA* and *QI* values via a Naïve Bayes classifier [33], exploiting divergences between the *global SA* distribution and *local* knowledge. As β-likeness controls this divergence, it curbs the attack [11].



(a) CENSUS, range queries (b) COIL2000, range queries (c) CENSUS, prefix queries

Figure 10: Median query relative error: effect of β, ϵ for PrivBayes.



(a) NB, effect of β, ϵ (b) NB with LC, effect of β, ϵ (c) Effect of δ for PrivBayes

Figure 11: NB attack on CENSUS: effect of β with classic NB & NB with Lapl. correction, effect of δ with both for PrivBayes.

We consider a data vendor anonymizing data once for all purposes, contrariwise to works that train a representation for each task [62], hence succumb to *composition attacks* [19].

Cormode [14] suggests an attack based on a *Naïve Bayes (NB) classifier* that predicts the *SA* value of a tuple t with m *QI* values, t_j, $1 \leq j \leq m$, as $\hat{v}(t) = \arg\max_{v_i \in V} \Pr[v_i] \prod_{j=1}^{m} \Pr[t_j|v_i]$. The attack learns conditional probabilities $\Pr[t_j|v_i]$ from noisy count query results. While the noise conceals the contribution of any individual, its effect on $\Pr[t_j|v_i]$ is relatively small [14]; the classifier effectively exploits variations of the conditional probability $\Pr[t_j|v_i]$ values from their unconditional counterpart, $\Pr[t_j]$; β-likeness bounds exactly those variations [11], while DP bounds the effect of one's participation in the data [31, 52]. By Bayes' rule, $\Pr[t_j|v_i] = {}^{\Pr[v_i|t_j]}/_{\Pr[v_i]} \Pr[t_j]$. For a given SV v_i, $\Pr[v_i]$ is the prior confidence in v_i based on the global SV distribution, while $\Pr[v_i|t_j]$ is the posterior confidence that β-likeness bounds as $\Pr[v_i|t_j] \leq (1+\beta)\Pr[v_i]$, hence $\Pr[t_j|v_i] \leq (1+\beta)\Pr[t_j]$, i.e., β-likeness compromises an NB attack [11]. To validate this claim, we assess NB attacks estimating $\Pr[t_j|v_i] = {}^{(\text{count}(t_j|v_i)+\alpha)}/_{(\text{count}(v_i)+\alpha \cdot |QI_j|)}$, where $\text{count}(t_j|v_i)$ is the number of anonymized tuples with SV v_i and QI_j value matching t_j; we set α to 0 in the classic NB and to 1 with *Laplacian correction* (LC), yielding an estimate between empirical probability ${}^{\text{count}(t_j|v_i)}/_{\text{count}(v_i)}$ and uniform probability ${}^{1}/_{|QI_j|}$. In both cases, we return the v_i^* maximizing $\Pr[v_i|t]$. Figure 11 reports accuracy as the ratio of true predictions over all tuples. On generalized tuples, we count each t' having SV v_i and generalized t'_j containing t_j. On PrivBayes's data, we count noisy t'_j within a small normalized distance δ from t_j; we use $\delta = 0.1$ with classic NB in Figure11(a) and $\delta = 0.0495$ with Laplacian Correction in Figure 11(b), as these values yield best results; Figure 11(c) presents our investigation on PrivBayes with the δ parameter ranging from 0 to 1 and $\epsilon = 1.386$ corresponding to $\beta = 3$. In Figure 11(a), the attack accuracy on data produced by β-likeness algorithms is unaffected by β. PrivBayes, however, becomes susceptible as ϵ grows. In Figure 11(b), the Laplacian correction improves the attack accuracy on PrivBayes and BuReL data, yet not much for heterogeneous β-likeness. We emphasize that we measure resistance against the attack under the same regime for all methods.

Our results show that heterogeneous β-likeness is more resistant than PrivBayes and BuReL[6] to learning-based attacks. The only scheme that *formally* protects against learning-based attacks, Pufferfish [36], requires knowledge of all possible data distributions. Against adversaries knowing *arbitrary correlations*, no scheme achieves both privacy and utility [45].

7 Conclusions

We introduced one-off anonymization schemes that shield sensitive information by heterogeneous generalization, using randomization to thwart adversaries who know the algorithm. Our experiments show that these schemes incur lower information loss and yet, by β-likeness, provide stronger resistance than state-of-the-art differential privacy schemes to learning-based attacks under our adversary model on real-world data.

Our schemes assume a single sensitive attribute. In case several attributes are deemed sensitive, a non-trivial solution would delimit the adversary's confidence on a tuple's deviation from the *joint distribution* of sensitive values.

[6]Notably, BuReL offers the same β-likeness privacy guarantee as our schemes, yet creates homogeneous groups that facilitate learning.

Availability

Our code and data are at http://github.com/discont.

References

[1] https://kdd.ics.uci.edu/databases/tic/tic.html.

[2] http://www.ipums.org.

[3] Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *VLDB*, 2005.

[4] Khalil Al-Hussaeni, Benjamin C. M. Fung, Farkhund Iqbal, Junqiang Liu, and Patrick C. K. Hung. Differentially private multidimensional data publishing. *Knowl. Inf. Syst.*, 56(3):717–752, 2018.

[5] Fatemeh Amiri, Nasser Yazdani, and Azadeh Shakery. Bottom-up sequential anonymization in the presence of adversary knowledge. *Inf. Sci.*, 450:316–335, 2018.

[6] Fatemeh Amiri, Nasser Yazdani, Azadeh Shakery, and Amir H. Chinaei. Hierarchical anonymization algorithms against background knowledge attack in data releasing. *Knowl. Based Syst.*, 101:71–89, 2016.

[7] Fatemeh Amiri, Nasser Yazdani, Azadeh Shakery, and Shen-Shyang Ho. Bayesian-based anonymization framework against background knowledge attack in continuous data publishing. *Trans. Data Priv.*, 12(3):197–225, 2019.

[8] Hai Brenner and Kobbi Nissim. Impossibility of differentially private universally optimal mechanisms. In *FOCS*, pages 71–80, 2010.

[9] Hai Brenner and Kobbi Nissim. Impossibility of differentially private universally optimal mechanisms. *SIAM J. Comput.*, 43(5):1513–1540, 2014.

[10] Justin Brickell and Vitaly Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *KDD*, pages 70–78, 2008.

[11] Jianneng Cao and Panagiotis Karras. Publishing microdata with a robust privacy guarantee. *PVLDB*, 5(11):1388–1399, 2012.

[12] Jianneng Cao, Panagiotis Karras, Panos Kalnis, and Kian-Lee Tan. SABRE: a Sensitive Attribute Bucketization and REdistribution framework for t-closeness. *The VLDB Journal*, 20(1):59–81, 2011.

[13] Chris Clifton and Tamir Tassa. On syntactic anonymity and differential privacy. *Trans. Data Priv.*, 6(2):161–183, 2013.

[14] Graham Cormode. Personal privacy vs population privacy: learning to attack anonymization. In *KDD*, pages 1253–1261, 2011.

[15] Graham Cormode, Ninghui Li, Tiancheng Li, and Divesh Srivastava. Minimizing minimality and maximizing utility: Analyzing method-based attacks on anonymized data. *PVLDB*, 3(1):1045–1056, 2010.

[16] Katerina Doka, Mingqiang Xue, Dimitrios Tsoumakos, and Panagiotis Karras. k-anonymization by freeform generalization. In *ASIACCS*, pages 519–530, 2015.

[17] Cynthia Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.

[18] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. of the ACM*, 19(2):248–264, 1972.

[19] Srivatsava Ranjit Ganta, Shiva Prasad Kasiviswanathan, and Adam Smith. Composition attacks and auxiliary information in data privacy. In *KDD*, pages 265–273, 2008.

[20] Gabriel Ghinita, Panagiotis Karras, Panos Kalnis, and Nikos Mamoulis. Fast data anonymization with low information loss. In *VLDB*, pages 758–769, 2007.

[21] Gabriel Ghinita, Panagiotis Karras, Panos Kalnis, and Nikos Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. *ACM TODS*, 34(2):1–47, 2009.

[22] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *STOC*, pages 351–360, 2009.

[23] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. *SIAM J. Comput.*, 41(6):1673–1693, 2012.

[24] Aristides Gionis, Arnon Mazza, and Tamir Tassa. k-anonymization revisited. In *ICDE*, pages 744–753, 2008.

[25] Abigail Goldsteen, Gilad Ezov, Ron Shmelkin, Micha Moffie, and Ariel Farkash. Data minimization for GDPR compliance in machine learning models. *CoRR*, abs/2008.04113, 2020.

[26] Qiyuan Gong, Junzhou Luo, Ming Yang, Weiwei Ni, and Xiao-Bai Li. Anonymizing 1:M microdata with high utility. *Knowl. Based Syst.*, 115:15–26, 2017.

[27] Qiyuan Gong, Ming Yang, Zhouguo Chen, Wenjia Wu, and Junzhou Luo. A framework for utility enhanced incomplete microdata anonymization. *Clust. Comput.*, 20(2):1749–1764, 2017.

[28] Yeye He and Jeffrey F. Naughton. Anonymization of set-valued data via top-down, local generalization. *PVLDB*, 2(1):934–945, 2009.

[29] Tochukwu Iwuchukwu and Jeffrey F. Naughton. k-anonymization as spatial indexing: Toward scalable and incremental anonymization. In *VLDB*, pages 746–757, 2007.

[30] James E. Johndrow and Kristian Lum. An algorithm for removing sensitive information: Application to race-independent recidivism prediction. *Ann. Appl. Stat.*, 13(1):189–220, 03 2019.

[31] Shiva Prasad Kasiviswanathan and Adam Smith. On the 'semantics' of differential privacy: A Bayesian formulation. *J. Priv. Confidentiality*, 6(1), 2014.

[32] Rashid Hussain Khokhar, Rui Chen, Benjamin C. M. Fung, and Siu Man Lui. Quantifying the costs and benefits of privacy-preserving health data publishing. *J. Biomed. Informatics*, 50:107–121, 2014.

[33] Daniel Kifer. Attacks on privacy and deFinetti's theorem. In *SIGMOD*, pages 127–138, 2009.

[34] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *SIGMOD*, pages 193–204, 2011.

[35] Daniel Kifer and Ashwin Machanavajjhala. A rigorous and customizable framework for privacy. In *PODS*, pages 77–88, 2012.

[36] Daniel Kifer and Ashwin Machanavajjhala. Pufferfish: A framework for mathematical privacy definitions. *ACM Trans. Database Syst.*, 39(1):3:1–3:36, 2014.

[37] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.

[38] Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Workload-aware anonymization techniques for large-scale datasets. *ACM TODS*, 33(3):17:1–17:47, 2008.

[39] Dong Li, Xianmang He, Longbing Cao, and Huahui Chen. Permutation anonymization. *J. Intell. Inf. Syst.*, 47(3):427–445, 2016.

[40] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. Closeness: A new privacy measure for data publishing. *IEEE TKDE*, 22(7):943–956, 2010.

[41] Yuting Liang and Reza Samavi. Optimization-based k-anonymity algorithms. *Comput. Secur.*, 93:101753, 2020.

[42] Ashwin Machanavajjhala and Daniel Kifer. Designing statistical privacy for your data. *Commun. ACM*, 58(3):58–67, 2015.

[43] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM TKDD*, 1(1):3, 2007.

[44] Sadegh Nobari, Panagiotis Karras, HweeHwa Pang, and Stéphane Bressan. $\mathcal{L}$-opacity: Linkage-aware graph anonymization. In *EDBT*, pages 583–594, 2014.

[45] Vibhor Rastogi, Sungho Hong, and Dan Suciu. The boundary between privacy and utility in data publishing. In *VLDB*, pages 531–542, 2007.

[46] Surapon Riyana, Srikul Nanthachumphu, and Noppamas Riyana. Achieving privacy preservation constraints in missing-value datasets. *SN Comput. Sci.*, 1(4):227, 2020.

[47] Luc Rocher, Julien M. Hendrickx, and Yves-Alexandre de Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature Communications*, 10(3069), 2019.

[48] Pierangela Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6):1010–1027, 2001.

[49] Shuang Song, Yizhen Wang, and Kamalika Chaudhuri. Pufferfish privacy mechanisms for correlated data. In *SIGMOD*, pages 1291–1306, 2017.

[50] Tamir Tassa, Arnon Mazza, and Aristides Gionis. k-concealment: An alternative model of k-type anonymity. *Transactions on Data Privacy*, 5(1):189–222, 2012.

[51] N. Tomizawa. On some techniques useful for solution of transportation network problems. *Networks*, 1:173–194, 1971.

[52] Michael Carl Tschantz, Shayak Sen, and Anupam Datta. Sok: Differential privacy as a causal property. In *IEEE Symposium on Security and Privacy*, pages 354–371, 2020.

[53] Jinyan Wang, Kai Du, Xudong Luo, and Xianxian Li. Two privacy-preserving approaches for data publishing with identity reservation. *Knowl. Inf. Syst.*, 60(2):1039–1080, 2019.

[54] Wendy Hui Wang and Ruilin Liu. Hiding outliers into crowd: Privacy-preserving data publishing with outliers. *Data Knowl. Eng.*, 100:94–115, 2015.

[55] Raymond Chi-Wing Wong, Ada Wai-Chee Fu, Ke Wang, and Jian Pei. Minimality attack in privacy preserving data publishing. In *VLDB*, pages 543–554, 2007.

[56] Wai Kit Wong, Nikos Mamoulis, and David Wai Lok Cheung. Non-homogeneous generalization in privacy preserving data publishing. In *SIGMOD*, pages 747–758, 2010.

[57] Xiaokui Xiao, Yufei Tao, and Nick Koudas. Transparent anonymization: Thwarting adversaries who know the algorithm. *ACM TODS*, 35(2):1–48, 2010.

[58] Jian Xu, Wei Wang, Jian Pei, Xiaoyuan Wang, Baile Shi, and Ada Wai-Chee Fu. Utility-based anonymization using local recoding. In *KDD*, pages 785–790, 2006.

[59] Mingqiang Xue, Panagiotis Karras, Chedy Raïssi, Jaideep Vaidya, and Kian-Lee Tan. Anonymizing set-valued data by nonreciprocal recoding. In *KDD*, 2012.

[60] Dan Zhang, Ryan McKenna, Ios Kotsogiannis, George Bissias, Michael Hay, Ashwin Machanavajjhala, and Gerome Miklau. ϵKTELO: A framework for defining differentially private computations. *ACM Trans. Database Syst.*, 45(1):2:1–2:44, 2020.

[61] Jun Zhang, Graham Cormode, Cecilia M. Procopiuc, Divesh Srivastava, and Xiaokui Xiao. PrivBayes: Private data release via Bayesian networks. *ACM Trans. Database Syst.*, 42(4):25:1–25:41, 2017.

[62] Han Zhao, Jianfeng Chi, Yuan Tian, and Geoffrey J. Gordon. Trade-offs and guarantees of adversarial representation learning for information obfuscation. In *NeurIPS*, 2020.

[63] Bin Zhou and Jian Pei. The k-anonymity and ℓ-diversity approaches for privacy preservation in social networks against neighborhood attacks. *Knowl. Inf. Syst.*, 28(1):47–77, 2011.