CoP: Coordinated Perturbation for Controlled Disclosure Under Local Differential Privacy

Sandaru Jayawardana
The University of Sydney
Australia
sandaru.jayawardana@sydney.edu.au

Ming Ding
Technology, CSIRO
Australia
ming.ding@csiro.au

Kanchana Thilakarathna
The University of Sydney
Australia
kanchana.thilakarathna@sydney.edu.au

Abstract

Collecting multidimensional user data is essential for personalized services, yet it poses significant privacy risks. While privacy regulations like the GDPR and CPRA advocate for data minimization, attribute correlations can inadvertently amplify unintentional information disclosure, leading to correlation-induced information leakage (CIL). Although data collectors often possess rich prior knowledge of these correlations, existing Local Differential Privacy (LDP) mechanisms are inadequate for effectively leveraging this information to reduce CIL.

In this paper, we propose CoP, a coordinated perturbation mechanism designed to mitigate CIL in multidimensional data collection while preserving utility. Unlike traditional LDP approaches, CoP explicitly incorporates prior distribution knowledge to coordinate the perturbation process across attributes. By optimizing the perturbation strategy based on known correlations, CoP achieves a better privacy-utility trade-off. Extensive evaluations across both synthetic and real-world datasets demonstrate that CoP significantly outperforms state-of-the-art LDP mechanisms in reducing disclosure while preserving analytical accuracy.

Keywords

Local differential privacy, Coordinated perturbation, Information theory

1 Introduction

In the current information era, collecting multidimensional user data (i.e., data with multiple attributes) has become essential for personalization and analytics such as feature recommendation in online services, app/OS telemetry, and IoT/health/wearable measurements [15, 39, 46]. Usually, such personal information contains various sensitive data, e.g., personally identifiable information (PII), financial data, and health records, which could pose significant privacy and security risks [41, 46]. Moreover, service providers typically have rich datasets they have previously collected to improve their services, e.g., content recommendations in TikTok [39]. **Data minimization** is advocated by privacy regulations such as the General Data Protection Regulation (GDPR) [18] and the California Privacy Rights Act (CPRA) [5] to mitigate privacy risks inherent in personal data collection. However, current data collection practices **often exceed what is necessary** [19]. This is mainly because service providers tend to ignore potential privacy leakages caused by statistical dependencies, and/or ignore prior knowledge in privacy protection mechanisms [27]. This motivates the need for privacy mechanisms that enforce data minimization while retaining only the information required for the intended service task.

Current data minimization techniques are mainly enforced by collecting only a task-relevant subset of attributes, and coarsening values (e.g., bucketing/quantization) when high precision is unnecessary [18, 38]. When the remaining information is still sensitive, minimization should be supported by a formal privacy mechanism[1]. ε-Local Differential Privacy (ε-LDP) [13] provides such a guarantee by perturbing user data locally before disclosure.

In service-oriented settings, utility is also essential: the protected data must retain sufficient task-relevant information for downstream analytics, model learning, or adaptive decision-making. Here, utility **does not** mean exact reconstruction of each user's private attributes or deterministic per-user personalization. Rather, it refers to the expected usefulness of randomized reports for the intended service task. For example, in a *personalized content recommendation system* (e.g., *'For You'* feed in TikTok [39]), a platform may select content for a user and observe *stochastic* engagement feedback, such as a randomized click, rating, or a reward derived from dwell time. This type of service does not need to recover the user's exact private attributes; instead, it only requires the reported signal to remain informative enough to update the recommendation policy. Thus, LDP-protected reports can support learning or adaptive decision-making when the service can tolerate noisy inputs, and the reports preserve sufficient task-relevant information.

In multidimensional data, useful information is mainly provided by the accuracy of each perturbed attribute report, which we refer to as direct information leakage (DIL[2]). However, additional information may also be inferred from attribute dependencies, leading to correlation-induced information leakage (CIL[2]). Although CIL can sometimes improve prediction accuracy, this utility gain comes from *unintended dependency-driven disclosure*. In particular, CIL reveals information about an attribute *beyond* its own DIL. This creates tension with *data-minimization* and *purpose-limitation principles*, which require personal data to be collected for specified purposes and limited to what is necessary for those purposes, as reflected in GDPR [18] and CPRA [5]. Moreover, the amount of inferable information from CIL (i.e., additional disclosure) depends on the data collector's prior knowledge of attribute dependencies, making it difficult to specify, audit, or limit at the time of collection. Therefore, our goal is to preserve the DIL needed for the intended

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.
Proceedings on Privacy Enhancing Technologies 2026(4), 793–813
© 2026 Copyright held by the owner/author(s).
https://doi.org/10.56553/popets-2026-0145

[1] A mechanism is an algorithm that perturbs the actual user data.
[2] Formal definitions are given in Section 2.3.

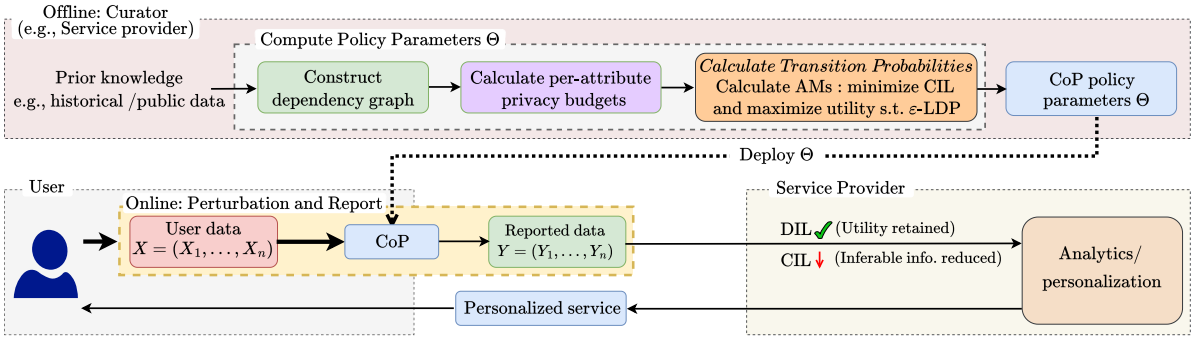


Figure 1: CoP workflow. *Offline* - A curator estimates dependencies from prior data and optimizes the policy parameters—dependency graph, per-attribute privacy budgets, and transition probabilities used for attribute perturbation—to retain utility (DIL) while reducing CIL under ε-LDP. *Online* - Users apply CoP with deployed Θ to perturb X into reports Y, enabling downstream analytics/personalization. CoP *selectively mitigates CIL*, thereby enhancing data minimization and reducing vulnerability to attribute inference attacks while retaining utility.

task while mitigating unnecessary CIL caused by attribute dependencies.

Existing LDP mechanisms for multidimensional correlated data mainly follow three approaches: (i) splitting the privacy budget among attributes, such as SPL [41] and GRF [44]; (ii) splitting the population, such as random sampling (RS) [41] and random sampling with fake data (RS+FD) [2]; and (iii) joint perturbation, such as key-value collection (PCKV) [20] and joint randomized response (JRR) [31]. Budget splitting increases the noise added to each attribute and can reduce utility. Population splitting can improve aggregate estimation, but is unsuitable when valid unit-level multidimensional records are required, because different attributes are collected from different user groups. Joint perturbation perturbs attributes together, but existing methods do not explicitly distinguish between useful direct information and unintended correlation-induced information. As a result, they cannot selectively mitigate CIL while preserving DIL.

To address this, we propose a novel coordinated perturbation mechanism, **CoP**, to mitigate CIL while preserving utility for a given privacy budget ε, as shown in Figure 1 (improving data minimization by retaining the required information while mitigating unintended additional disclosure). The policy parameters Θ (i.e., dependency graph, per-attribute privacy budgets, and transition probabilities used for attribute perturbation) must be computed by a curator (e.g., a service provider) using prior knowledge and then deployed to each user. Then, users use CoP to perturb their data before sending it to the service provider. Moreover, CoP can be used as a perturbation mechanism in application-specific LDP frameworks such as longitudinal data releasing [35], contextual bandit [48], etc.

In Section 3, we delve into the relationship between information leakages and attribute inter-dependency using Shannon information theory [8]. We theoretically show that *calibrated coordination* among attribute perturbation processes can reduce CIL while maintaining the useful direct information needed for utility. We make two assumptions: (i) attribute correlations are available as prior knowledge, and (ii) such correlation information is not itself private, as it can often be estimated from historical or public datasets.

Next, in Sections 4.1-4.5, we design CoP that mitigates CIL by leveraging prior knowledge based on information-theoretic findings in Section 3. Then, we develop a unified Algorithm 4 to compute the policy parameters Θ. Further, Section 4.4 shows that CoP provides ε–LDP guarantee. Next, Section 4.7 presents Algorithm 5 that perturbs user data using CoP policy parameters Θ at the user level.

Finally, in Section 5, we evaluate CoP on both synthetic and real-world datasets and show that it consistently outperforms state-of-the-art baselines. We also perform state-of-the-art attribute-inference attacks, demonstrating that CoP offers stronger resistance to inference attacks than competing methods. Together, these results validate the **effectiveness of coordinated perturbation** in mitigating CIL while preserving utility-relevant DIL.

The following are the main **contributions** of our work.

(1) We study CIL in LDP from an information-theoretic perspective and derive *leakage bounds* for a general coordinated perturbation approach. Here, we show that carefully coordinated perturbations can reduce CIL.
(2) We show that preserving the original correlation structure during perturbation provides a generic coordinated perturbation approach when correlation information is not private. We propose **CoP** as a prior-knowledge-aware coordinated perturbation mechanism to mitigate CIL while retaining DIL.
(3) Finally, we experimentally validate the information leakage, privacy guarantee, and utility of CoP using synthetic and real-world datasets. Results show that **CoP** outperforms state-of-the-art mechanisms.

2 Preliminaries and Problem Statement

In Sections 2.1-2.3, we present and review the related definitions and theorems required for the rest of the paper. Next, Section 2.4 presents the threat model considered. Finally, Section 2.5 formulates the problem statement.

2.1 Local Differential Privacy

In this work, we adopt the definition of LDP proposed by Duchi et al. [13].

Definition 2.1 (Local Differential Privacy [13]). *Let the input alphabet be $\mathcal{X}$ and the output alphabet be $\mathcal{Y}$. A mechanism Q is ε-locally differentially private if,*

$$\sup_{S\subseteq\mathcal{Y},\ x,x'\in\mathcal{X}} \frac{Q(S|x)}{Q(S|x')} \leq e^{\varepsilon}, \tag{1}$$

where $Q(S|x) = p(Y = y_i,\ y_i \in S \mid X = x)$ is the privatization mechanism. The function $p(\cdot)$ denotes the probability of an event. Here, $x \in \mathcal{X}$, $y_i \in \mathcal{Y}$ and $\varepsilon > 0$.

2.2 Measuring Information Leakage

Information-theoretic measures provide a principled way to quantify statistical dependence. In privacy and communication theory, Shannon measures are widely used to formalize information leakage [8, 9]. We adopt Shannon measures because they extend naturally to multivariate settings via the chain rule and support compositional analysis, whereas alternatives such as min-entropy are more worst-case-oriented [29].

Entropy. Entropy measures the uncertainty of a random variable. Entropy of discrete random variable X, $H(X)$ is defined as

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log(p(x)). \tag{2}$$

Here, $\mathcal{X}$ is its alphabet and the function $p(\cdot)$ denotes the probability of an event.

Mutual Information (MI). MI quantifies the reduction in uncertainty about one random variable given knowledge of another [7]. Intuitively, MI measures the dependency (linear and nonlinear) between variables. MI between random variables X and Y, $I(X;Y)$ is defined as

$$I(X;Y) = H(X) + H(Y) - H(X,Y). \tag{3}$$

Conditional Mutual Information (CMI). Let X, Y, Z be random variables. The CMI between X and Y given Z is

$$I(X;Y|Z) = H(X|Z) + H(Y|Z) - H(X,Y|Z). \tag{4}$$

Intuitively, $I(X;Y|Z)$ quantifies the information shared between X and Y when Z is known. Then, we can extend MI to multiple variables using the chain rule as

$$I(X_1, X_2, \ldots, X_n; Y) = \sum_{i\in[n]} I(X_i; Y|X_1, \ldots, X_{i-1}). \tag{5}$$

Pointwise Information (PMI). PMI quantifies the likelihood of simultaneous realizations of variables. For discrete random variables X and Y, PMI for realizations x and y is defined as

$$\text{pmi}(x;y) = \log\frac{p(x|y)}{p(x)} = \log\frac{p(x,y)}{p(x)\,p(y)}. \tag{6}$$

A realization y is *positively informative* about x if $\text{pmi}(x;y) > 0$. Then, observing y increases our expectation that x will occur. Conversely, if $\text{pmi}(x;y) < 0$, then observing y decreases our expectation that x will occur. Further, $I(X;Y)$ is given by,

$$I(X;Y) = \sum_{x,y} p(x,y)\ \text{pmi}(x;y). \tag{7}$$

2.3 Information Leakage of Perturbed Data

Let $X = (X_1, \ldots, X_n)$ be statistically dependent attributes, and let $Y = (Y_1, \ldots, Y_n)$ be perturbed values of X generated by random processes $M_1, \ldots, M_n$, respectively. Then, the information leakage for $X_k,\ k \in [n]$ is given by,

$$I(X_k; Y_1, \ldots, Y_n) = I(X_k; Y_k) + \sum_{i\in[n],\ i\neq k} I(X_k; Y_i|Y_k, Y_1, \ldots, Y_{i-1}). \tag{8}$$

We can group the information leakages in (8) as

(1) *Information leakage caused by the perturbed version of the same attribute:* $I(X_k; Y_k)$. This measures the information released about X_k by mechanism M_k, which contributes directly to utility. We refer to this as "*direct information leakage*" (DIL). Since this directly supports the utility of X_k, it is important to retain sufficient DIL to balance utility and privacy.

(2) *Information leakage caused by neighbouring attributes:* $\sum_{i\in[n],\ i\neq k} I(X_k; Y_i|Y_k, Y_1, \ldots, Y_{i-1})$. We refer to this as "*correlation-induced information leakage*" (CIL). Although CIL can improve the estimation of X_k, we treat it as additional leakage because it enables inference of X_k beyond the information **intentionally disclosed** through Y_k, thereby undermining the goal of minimizing collection. Therefore, mitigating CIL is important for improving the privacy-utility trade-off.

2.4 Threat Model

Suppose each user holds record $X = (X_1, \ldots, X_n)$. The service provider collects a perturbed report $Y = (Y_1, \ldots, Y_n)$ generated by randomization mechanisms. Then, we consider a threat model in which an adversary possesses correlation information derived from public/historic data, or, in the worst case, the ground-truth probability distribution. The adversary aims to perform *attribute inference attacks* by leveraging dependencies among attributes, even when local perturbation mechanisms are employed. A potential adversary (i) *the service provider itself*, or (ii) *an attacker who gains access to the collected reports*. In this work, adversarial knowledge gain is quantified using CIL. For example, CIL of X_k represents the possible additional learning about X_k contained in $Y_1, \ldots, Y_{k-1}, Y_{k+1}, \ldots, Y_n$ beyond what is already disclosed by Y_k.

2.5 Problem Statement

Let $X = (X_1, \ldots, X_n)$ denote a user data record, where each X_i is a discrete feature requested by a service provider to enable personalized functionality (e.g., recommendation, adaptive actions). The service provider collects $Y = (Y_1, \ldots, Y_n)$, a randomized version of X, which is sufficient to support the service.

A key issue is that statistical dependencies between attributes can reveal more information than intended. We model this *CIL* of X_k as

$$CIL(X_k) = \sum_{i\in[n],\ i\neq k} I(X_k; Y_i|Y_k, Y_1, \ldots, Y_{i-1}). \tag{9}$$

At the same time, the collected Y must retain sufficient task-relevant DIL to preserve utility. We quantify task-relevant utility of X_k as

$$Utility(X_k) = \text{U}(X_k, Y_k). \tag{10}$$

Here, $U(\cdot)$ denotes the utility function that assigns a score to reporting y when the true value is x. Equivalently, $U(\cdot)$ can be defined through a loss function, where higher utility corresponds to lower expected distortion (See Section 4.3 for more details). Our goal is to design a strategy that preserves this attribute-level expected utility while limiting CIL. Formally,

$$\begin{aligned} &\text{maximize} && \text{Utility}(X_k) \\ &\text{subject to} && \text{CIL}(X_k) \le \tau_k, \\ & && \varepsilon\text{-LDP constraints}, \quad \forall k \in [n]. \end{aligned} \tag{11}$$

Here, τ_k denotes the maximum allowable CIL for attribute X_k. Directly optimizing $\text{CIL}(X_k)$ is challenging because it depends on CMI over the released attributes. Therefore, Section 4.3 instantiates this high-level objective using a tractable f-divergence-based proxy for controlling CIL.

3 Analyzing Information Leakage in LDP

In this section, we theoretically analyze the information leakage of LDP mechanisms in statistically dependent environments.

Following the system model in Section 2.3, we can quantify the total information leakage about X_k caused by releasing Y by,

$$I(X_k; Y) = I(X_k; Y_k) + I(X_k; Y_{-k}|Y_k), \tag{12}$$

using the chain rule. Here, $Y_{-k} = (Y_1, \ldots, Y_{k-1}, Y_{k+1}, \ldots, Y_n)$, $I(X_k; Y_k)$ corresponds to DIL, and $I(X_k; Y_{-k}|Y_k)$ corresponds to CIL. Then, CIL of X_k, $CIL(X_k)$ (i.e., $CIL(X_k) = I(X_k; Y_{-k}|Y_k)$) can be written as in Theorem 3.1. Therefore, there are two main approaches to minimize $CIL(X_k)$: (i) minimizing $I(X_k; Y_{-k})$ or (ii) maximizing $I(Y_k; Y_{-k}) - I(Y_k; Y_{-k}|X_k)$.

Next, we discuss these two approaches.

Theorem 3.1. *Let X_k, Y_k, Y_{-k} be random variables (with Y_{-k} possibly vector-valued). Then,*

$$I(X_k; Y_{-k}|Y_k) = I(X_k; Y_{-k}) - I(Y_k; Y_{-k}) + I(Y_k; Y_{-k}|X_k).$$

Proof is provided in Appendix B.1.

• **Approach 1: Minimizing $I(X_k; Y_{-k})$.** This is fundamentally **limited** when the utility of $X_{-k} = (X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n)$ must be maintained. Intuitively, if Y_{-k} remains informative about X_{-k} (i.e., for utility of X_{-k}), then Y_{-k} inevitably carries some information about X_k due to correlation between X_k and X_{-k}. Formally:

Theorem 3.2. *Let $X_1, \ldots, X_n$ be statistically dependent attributes, and let $Y_{-k} = (Y_1, \ldots, Y_{k-1}, Y_{k+1}, \ldots, Y_n)$ be generated from $X_{-k} = (X_1, \ldots, X_{k-1}, X_{k+1}, \ldots, X_n)$ by randomized mechanisms $M_1, \ldots, M_{k-1}, M_{k+1}, \ldots, M_n$. Then,*

$$I(X_k; Y_{-k}) \ge \max\{0,\ I(X_k; X_{-k}) - H(X_{-k}) + I(X_{-k}; Y_{-k})\}.$$

*Consequently, for a fixed utility requirement on X_{-k} (captured by $I(X_{-k}; Y_{-k})$), the leakage $I(X_k; Y_{-k})$ **cannot** be made arbitrarily small.*

The proof follows from the chain rule and the Markov property (equivalently, data processing) and is provided in Appendix B.2. *This result highlights that leakage through neighboring releases scales with both the correlation strength (via $I(X_k; X_{-k})$) and the required utility of X_{-k} (via $I(X_{-k}; Y_{-k})$).*

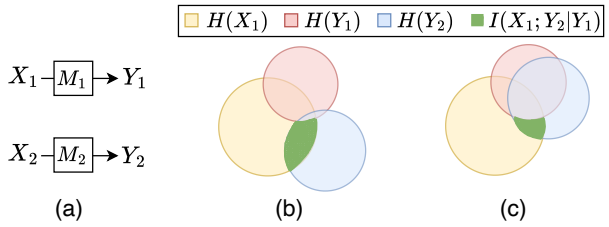


Figure 2: (a) Two attribute perturbations using LDP mechanisms. (b) Entropy Venn diagram for $H(X_1), H(Y_1)$, and $H(Y_2)$. Here, "Green region" illustrates the $CIL(X_1)$, $I(X_1; Y_2|Y_1)$ (before increasing $I(Y_1; Y_2)$). (c) Entropy Venn diagram for increased $I(Y_1; Y_2)$ by coordinated perturbation. Here, the intersected area between $H(Y_1)$ and $H(Y_2)$ is larger than Figure 2(b). Therefore, "green region" reduces as $I(Y_1; Y_2)$ increases.

• **Approach 2: Maximizing $I(Y_k; Y_{-k}) - I(Y_k; Y_{-k}|X_k)$.** The term $I(Y_k; Y_{-k}) - I(Y_k; Y_{-k}|X_k)$ represents the shared information between Y_k and Y_{-k} in a manner explained by X_k. Therefore, we should maximize the information shared between Y_k and Y_{-k} to minimize $CIL(X_k)$. Operationally, increasing the shared information between Y_k and Y_{-k} minimizes the extent to which Y_{-k} reveals about X_k, **beyond what is already revealed** by Y_k.

To illustrate this intuition, consider a simplified two-attribute scenario as shown in Figure 2a. Then, Figure 2b and Figure 2c depict entropy Venn diagrams, where each circle denotes an entropy and overlaps denote MI. In the independent-perturbation case (Figure 2b), the green region represents the $CIL(X_1)$, $I(X_1; Y_2|Y_1)$. When we increase the dependence between Y_1 and Y_2 while preserving their marginals (Figure 2c), the overlap between $H(Y_1)$ and $H(Y_2)$ increases (larger $I(Y_1; Y_2)$) and the green region shrinks, indicating reduced $CIL(X_1)$. For illustration, we treat $I(X_1; Y_1)$ as fixed because it is determined by the utility target for X_1. We also treat the overlap between $H(X_1)$ and $H(Y_2)$ as lower-bounded due to Theorem 3.2 under a fixed utility requirement for X_2.

In summary, coordinating the perturbation processes such that the released reports Y exhibit calibrated dependency provides a *principled route to mitigating CIL while retaining DIL* (i.e., mitigate inferable information from dependency while retaining utility-relevant information). Next, we propose a novel coordinated perturbation mechanism, CoP, based on this finding.

4 Designing CoP and Computing Policy Parameters

This section describes the design and computation of policy parameters for CoP. In Section 4.1, we explore how to calibrate dependencies among $Y_1, \ldots, Y_n$ to mitigate CIL (based on the findings in Section 3). We then present the design of CoP along with the supporting theories and privacy guarantees (Sections 4.2–4.5). Next, we develop a unified algorithm to compute the policy parameters Θ of CoP in Section 4.6. Finally, Section 4.7 presents the perturbation algorithm for CoP.

Moreover, as specified in the threat model (Section 2.4), in designing CoP, we assume that the attribute dependency structure is available as prior knowledge and is *treated as non-sensitive* in our

setting. This is reasonable when correlations can be learned from public/historic datasets [3, 30].

4.1 Coordinating the Perturbation Process

Let $X = (X_1, \ldots, X_n)$ denote a user data record, and let $Y = (Y_1, \ldots, Y_n)$ denote their released (perturbed) reports. In this work, we focus on finite-domain attributes (categorical or discretized numerical features). For simplicity, we assume that each report alphabet matches the corresponding input alphabet, i.e., $|\mathcal{Y}_i| = |\mathcal{X}_i|$ for all $i \in [n]$ (after any encoding/discretization), although our analysis does not fundamentally require equal alphabet sizes.

Our **goal** is to control the $CIL(X_k)$ (i.e., CIL of X_k), quantified by $I(X_k; Y_{-k}|Y_k)$. Theorem 3.2 indicates that leakage from neighbouring releases is fundamentally driven by the dependency between X_k and X_{-k} (e.g., via $I(X_k; X_{-k})$) together with the utility that must be preserved in Y_{-k} (e.g., via $I(X_{-k}; Y_{-k})$). In contrast, Theorem 3.1 shows that inducing dependency between Y_k and Y_{-k} can reduce CIL by increasing redundancy, i.e., by increasing $I(Y_k; Y_{-k})$. This motivates calibrating the dependency of (Y_k, Y_{-k}) to *track* the dependency structure of (X_k, X_{-k}), so that Y_{-k} does **not** act as *an additional independent source of evidence* about X_k beyond Y_k.

A natural way to introduce such dependency is to calibrate the joint distribution of the perturbed reports toward that of the original attributes:

$$P_{Y_1,\ldots,Y_n} \approx P_{X_1,\ldots,X_n}.$$

*This condition should be interpreted as a dependency-calibration objective, **not** as an exact equality requirement.* In particular, the goal is not to reproduce the original data distribution exactly, but to preserve enough of its dependency structure so that neighboring reports, i.e., Y_{-k}, are less likely to introduce *uncontrolled additional evidence* about X_k beyond what is already reflected in Y_k. Therefore, joint distribution calibration provides a high-level principle for controlling CIL.

However, directly enforcing this joint calibration is not scalable. It would require defining a mechanism over the full joint alphabet $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, whose size grows exponentially with the number of attributes as $|\mathcal{X}| = \prod_{i=1}^{n} |\mathcal{X}_i|$. To obtain a scalable mechanism, we approximate the dependency structure using local conditional factorization as

$$P_{X_1,\ldots,X_n} \approx \prod_{k=1}^{n} P_{X_k|X_{\mathrm{pa}(k)}},$$

where $\mathrm{pa}(k) \subseteq [n] \setminus \{k\}$ denotes the selected parent attributes of X_k in the dependency graph. For root attributes, $\mathrm{pa}(k) = \emptyset$, and the corresponding factor is the marginal distribution P_{X_k}.

Based on this factorization, we construct an attribute-wise coordinated perturbation mechanism. Attributes are perturbed sequentially, and the perturbation of X_k is allowed to depend on the perturbed outputs of its parent attributes as

$$Y_k \sim P_{Y_k|X_k, Y_{\mathrm{pa}(k)}}(\cdot|x_k, y_{\mathrm{pa}(k)}).$$

The local calibration objective is then

$$P_{Y_k|Y_{\mathrm{pa}(k)}} \approx P_{X_k|X_{\mathrm{pa}(k)}}.$$

That is, instead of matching the full joint distributions directly, we match local conditional dependencies between each attribute and its selected parents.

Next, let us consider an example.

Example - Let X_1 and X_2 be two statistically dependent binary attributes, and let Y_1 and Y_2 be their perturbed outputs (Figure 3). Suppose $\mathcal{X}_1 = \{x_{11}, x_{12}\}$ and $\mathcal{X}_2 = \{x_{21}, x_{22}\}$ are alphabets of X_1 and X_2. Since the input and perturbed output alphabets are the same, we have alphabets of Y_1 and Y_2 as $\mathcal{Y}_1 = \{y_{11}, y_{12}\} = \{x_{11}, x_{12}\}$ and $\mathcal{Y}_2 = \{y_{21}, y_{22}\} = \{x_{21}, x_{22}\}$, respectively. Suppose, for given user data, X_1 perturbs first, and the perturbed output is y_{11}. Then, to achieve $P_{X_2|X_1} \approx P_{Y_2|Y_1}$, we need to make sure Y_2 (i.e., perturbed output of X_2) has distribution $P_{X_2|X_1=x_{11}} \approx P_{Y_2|Y1=y_{11}}$. Therefore, we need two mechanisms (i.e., $M_{2,y_{11}}$ and $M_{2,y_{12}}$) for X_2 since P_{Y_2} should be changed based on Y_1 (See Figure 3).

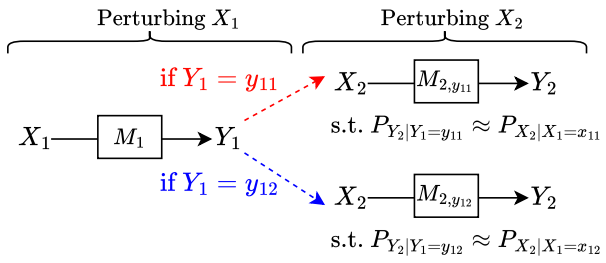


Figure 3: Example - Coordinated perturbation process of two attributes X_1 and X_2.

The optimization in Section 4.3 instantiates this idea using a tractable f-divergence constraint between the conditional distribution of the perturbed reports and the corresponding conditional distribution of the original attributes. This constraint provides a tractable local calibration condition for controlling CIL(X_k).

Next, we introduce a two-stage pipeline for coordinated perturbation by generalizing this sequential perturbation approach.

Stage 1 (Dependency graph (DG) construction). We build a DG that captures *salient inter-attribute relationships*. Constructing a correlation model (or a DG) is a common dimension-reduction technique in the literature for identifying statistically significant dependencies among attributes. Section 4.2 provides details on how we infer this dependency structure.

Stage 2 (Atomic mechanism (AM) instantiation). We instantiate *atomic* ε-LDP mechanisms for each attribute. For attribute X_k, the AMs are parameterized by a vector $Z_{k,l}$ that represents the probability distribution of X_k conditioned on l^{th} realization of its parent $\mathrm{pa}(k)$. Here, we limit the dependency to a single parent attribute because the joint alphabet of the parent mechanisms grows exponentially with the number of parent attributes. Concretely, if $\mathrm{pa}(k) \neq \emptyset$, we set $Z_{k,l} = P_{X_k|X_{\mathrm{pa}(k)}=l}$, where the conditional is computed from the prior knowledge. If $\mathrm{pa}(k) = \emptyset$, we use an uninformative default prior (uniform over $\mathcal{X}_k$) to avoid introducing dataset-specific bias through $Z_{k,l}$. We then compute the AMs for all attributes. Section 4.3 details the formulation and computation of these AMs.

Next, we describe how the DG is constructed.

4.2 Stage 1: Generating the Dependency Graph

We generate the (DG) using a two-stage screening procedure:

(1) **Attribute-level screening using MI.** Theorem 3.2 suggests that CIL increases with the shared information between attributes. Accordingly, we retain only strongly dependent attribute pairs. For each pair (X_i, X_j) where $i, j \in [n],\ i \neq j$, we estimate the $I(X_i; X_j)$ and keep the edge (i, j) if

$$I(X_i; X_j) > \mathrm{I}_{thr}\,. \tag{13}$$

Here, I_{thr} is a threshold.

(2) **Realization-level screening using PMI.** In our preliminary experiments, we observed that the high mutual information may be driven by a small number of particularly informative value co-occurrences, while many other realizations contribute little to CIL. Therefore, for each retained attribute pair (from attribute-level screening), we compute the PMI for each realization pair (x_i, x_j) and keep only those satisfying

$$\mathrm{pmi}(x_i; x_j) > \mathrm{PMI}_{thr}\,. \tag{14}$$

Here, PMI_{thr} is a threshold. See Appendix C.1 for additional details.

From this two-stage screening procedure, we identify the most significant attribute dependencies and reduce the dimensionality.

Algorithm 1, provided in Appendix D, summarizes the procedure for constructing the dependency graph (DG). Since Algorithm 1 learns the dependency structure from external or public sources, it does **not** consume the privacy budget of the users being sampled.

The algorithm iterates over all attribute pairs (line 2) and visits their joint realizations (line 4). Then, for each joint realization, it computes the corresponding PMI at line 5. It then retains only those MI between attributes $\mathrm{I}(X_i; X_j) > \mathrm{I}_{thr}$ and realization-level links with $\mathrm{pmi}(x_i; x_j) > \mathrm{PMI}_{thr}$ (lines 6–8). The resulting set of retained links constitutes the DG's edge set (i.e., E). Here, we use a dictionary variable (i.e., a key-value pair) to store the edge information.

Time Complexity Analysis. Since this algorithm visits all attribute pairs and their attribute realizations, the time complexity is $O(n^2 t^2)$. Here, n is the attribute count and t is the alphabet size of attributes, and it is assumed that MI and PMI values are already computed and cached. Next, we formulate the AMs.

4.3 Stage 2: Computing Atomic Mechanism

AMs are individual mechanisms for each attribute, perturbed based on the perturbed outputs of the parents. As defined in problem statement (11), our goal is to maximize utility while controlling $\mathrm{CIL}(X_k)$ under the LDP guarantee. Therefore, we optimize the transition probabilities of AMs to achieve the above requirements.

Next, we define the necessary terms to formulate the optimization problem to compute the transition probabilities of AM corresponding to X_k with realization $l \in \mathcal{Y}_{pa(k)}$ of the output of its parent. Here, $\mathcal{Y}_{pa(k)}$ denotes the alphabet of the perturbed output of the parent attribute $X_{pa(k)}$. Let X_k and Y_k be AM's input and output, and let $\mathcal{X}_k$ and $\mathcal{Y}_k$ be alphabets of X_k and Y_k respectively. Then, a prior distribution corresponds to realization l, $Z_{k,l}$ as

$$Z_{k,l} = \big(p(a_1|l), \cdots, p(a_t|l)\big);\ a_i \in \mathcal{X}_k,\ i \in [t],\ t = |\mathcal{X}_k| = |\mathcal{Y}_k|. \tag{15}$$

Let Q denote the transition probability matrix of an AM, and let E_U denote the distortion matrix, where each entry quantifies the error incurred when a true value is mapped to a perturbed output. The error function $F(\cdot)$ is chosen according to the attribute type, e.g., categorical or discrete numerical, or according to an application-specific distortion criterion. C denotes the uniform distribution of size t.

$$Q = \begin{pmatrix} p(b_1|a_1, l) & \cdots & p(b_t|a_1, l) \\ \vdots & \ddots & \vdots \\ p(b_1|a_t, l) & \cdots & p(b_t|a_t, l) \end{pmatrix}, \tag{16}$$

$$E_U = \begin{pmatrix} e_{1,1} & \cdots & e_{1,t} \\ \vdots & \ddots & \vdots \\ e_{t,1} & \cdots & e_{t,t} \end{pmatrix}, \tag{17}$$

$$C = \begin{pmatrix} c_1 & c_2 & \cdots & c_t \end{pmatrix}. \tag{18}$$

Here, $b_j \in \mathcal{Y}_k,\ a_i \in \mathcal{X}_k,\ l \in \mathcal{X}_{pa(k)}, e_{j,i} = |F(a_i, b_j)|,\ i, j \in [t],\ t = |\mathcal{X}_k|$, and $c_i = \frac{1}{t}$ for all $i \in [t]$. We have defined the objective function and the constraints of the optimization problem as follows.

The original problem in (11) involves two goals: maximizing $\mathrm{Utility}(X_k)$ and limiting $\mathrm{CIL}(X_k)$. In our optimization formulation, we control CIL by imposing constraints on the distribution of the perturbed report Y_k. Therefore, the objective function focuses on maximizing utility. The resulting objective function and constraints are described below.

Objective Function. We minimize the expected distortion of attribute X_k, which is equivalent to maximizing the attribute-level utility defined in Eq. (10).

Constraints. We introduce three constraints.

(1) First, we impose a local dependency-calibration constraint:

$$D_f(C^T Q || Z_{k,l}) \leq \alpha.$$

Here, $Z_{k,l} = P_{X_k|X_{\mathrm{pa}(k)}=l}$ is the prior conditional distribution, and $C^T Q$ denotes the corresponding perturbed conditional distribution $P_{Y_k|Y_{\mathrm{pa}(k)}=l}$. The parameter α controls the allowed deviation between these two distributions. This constraint provides a tractable local calibration condition for controlling CIL. Instead of directly optimizing the full CMI term $\mathrm{CIL}(X_k)$, we bound the discrepancy between the perturbed conditional distribution and the prior conditional distribution. By keeping this discrepancy small, the mechanism limits the extent to which neighboring reports (i.e., Y_{-k}) provide additional side information about X_k beyond its own report Y_k. See Section 4.1 for more details.

(2) Next, we have ε-LDP constraints where $Q(\cdot|a, l) \preceq e^{\varepsilon} Q(\cdot|a', l)$ for all $a, a' \in \mathcal{X}_k$. Here, operation $\preceq$ denotes element-wise $\leq$ comparison of two vectors.

(3) Finally, we have constraints related to probabilities (probabilities are non-negative and sum to one).

Based on the above objective function and constraints, we can formulate the optimization problem as

$$\begin{aligned}
\underset{Q}{\textbf{minimize}} \quad & \text{Expected distortion} = C^T \operatorname{diag}(QE_U) \\
\textbf{subject to} \quad & D_f(C^T Q \| Z_{k,l}) \preceq \alpha;\ \alpha = \text{dependency-calibration tolerance}, \\
& Q(\cdot|a,l) \preceq e^{\varepsilon} Q(\cdot|a',l);\ \forall a, a' \in \mathcal{X}_k, \varepsilon \text{ - privacy budget}, \\
& q \geq 0; \quad \forall q \in Q,\ \text{probabilities} \geq 0, \\
& \sum_{i \in [t]} Q(b_i|a,l) = 1;\ \forall a \in \mathcal{X}_k,\ b_i \in \mathcal{Y}_k, \\
& \qquad\qquad \text{sum of probability distribution} = 1.
\end{aligned} \tag{19}$$

Here, superscript T is the transpose of the matrix and '$\operatorname{diag}(QE_U)$' is the vector formed with diagonal values of QE_U. See Section 4.4 for a formal ε-LDP privacy guarantee for CoP. The minimization problem (19) is convex for f-divergence $D_f(\cdot\|\cdot)$ with convex f (e.g., KL-divergence, total variation [45]). We simplify this optimization further to a linear optimization by adopting an entry-wise bounded deviation constraint

$$|C^T Q - Z_{k,l}| \preceq \boldsymbol{\alpha}, \tag{20}$$

where $\boldsymbol{\alpha} = (\alpha, \alpha, \ldots, \alpha) \in \mathbb{R}^t$. This converts the distribution-matching component into $2t^2$ linear inequalities. Then, we can solve this linear program (LP) using the primal–dual interior-point method [4]. If we use the interior-point method to solve the LP, then we have $O(t^3)$ constraints. Further, we have simplified LDP constraints to $O(t^2)$ by adopting the (utility-aligned) diagonal-dominance condition (See Appendix C).

Time Complexity Analysis. We solve the problem using the interior point method. The time complexity of an AM is $O(t^7 \log(1/\tau))$ as there are t^2 optimization variables and $O(t^2)$ constraints. Here, τ is the precision of the LP solution [4], and t is the alphabet size of AM's inputs and outputs.

4.4 Privacy Guarantee of CoP

We next analyze privacy leakage of CoP under LDP by modeling attribute relationships using Markov chains. As stated in Theorem 4.1, prior knowledge of attribute dependencies allows us to derive a *tighter upper bound* on the *privacy leakage* than the standard sequential composition bound [14]. Here, 'privacy leakage' denotes the worst-case log-likelihood ratio between the output distributions induced by two possible values of the same input attribute, analogous to the LDP Definition 2.1. Thus, it quantifies how well an adversary can distinguish between two candidate values of an attribute after observing the perturbed output. Although the dependency-aware analysis can provide a tighter bound, the worst-case privacy leakage of CoP is still bounded by $\sum_{i \in [n]} \varepsilon_i$, following the sequential composition rule. Therefore, CoP satisfies ε-LDP whenever $\sum_{i \in [n]} \varepsilon_i \leq \varepsilon$, where ε_i denotes the privacy budget allocated to the i^{th} attribute.

We further develop a polynomial-time algorithm with complexity $O(n^2 t^3 \log t)$ to compute the optimal value of the upper bound in (22). The details of this computation are provided in Appendix E.

Theorem 4.1. *Let $X = (X_1, \ldots, X_n)$ be inputs of CoP and let $Y = (Y_1, \ldots, Y_n)$ be corresponding perturbed outputs. For an attribute X_i, define its privacy leakage as the worst-case log-likelihood ratio*

$$L_{X_i} = \max_{y_1 \in \mathcal{Y}_1,\ y_2 \in \mathcal{Y}_2, \cdots, y_n \in \mathcal{Y}_n,\ x, x' \in \mathcal{X}_i} \ln \frac{p(y_1, y_2, \ldots, y_n|x)}{p(y_1, y_2, \ldots, y_n|x')}. \tag{21}$$

Then, the privacy leakage L_{X_i} is bounded by,

$$L_{X_i} \leq \varepsilon_i + \max_{x, x' \in \mathcal{X}_i,\ C_{i,j},\ j \in [n],\ i \neq j} \sum_{j \in [n],\ i \neq j} \ln \frac{C_{i,j}^T\, G_{j,i}}{C_{i,j}^T\, G'_{j,i}} \leq \varepsilon. \tag{22}$$

Here, ε_i denotes the privacy budget of the i^{th} attribute, and $C_{i,j} \in \{1, e^{\varepsilon_j}\}^{t_j}$, where $t_j = |\mathcal{X}_j|$, captures the maximum contribution of the perturbation mechanism applied to attribute X_j. Here, $G_{j,i} = (p(x_{j,1}|x), \ldots, p(x_{j,t_j}|x))$, $G'_{j,i} = (p(x_{j,1}|x'), \ldots, p(x_{j,t_j}|x'))$ and $\mathcal{X}_j = \{x_{j,1}, \cdots, x_{j,t_j}\}$.

Moreover, the worst case privacy leakage of CoP is $\sum_{i \in [n]} \varepsilon_i$.

Proof is available in Appendix B.3.

4.5 Computing Per-Attribute Privacy Budgets for CoP

Using (22), we can analytically estimate the privacy leakage of CoP for given privacy budget ε. Therefore, we can maximize the per-attribute privacy budgets until the privacy leakage of CoP equals ε. This maximizes data utility for a given privacy guarantee. We develop a polynomial-time, $O(\frac{n^2 t^3 \log t}{\tilde{e}})$, Algorithm 3 (provided in Appendix F) to calibrate per-attribute privacy budgets. Here, $\tilde{e}$ denotes the precision of calibrated privacy budgets.

4.6 Computing Policy Parameters Θ

Next, we develop a unified Algorithm 4 (provided in Appendix G) to compute policy parameters Θ of CoP. First, we compute the DG using Algorithm 1 at line 1. Next, we calibrate per-attribute privacy budgets using Algorithm 3 at line 2. Next, we traverse all the edges in the DG and compute the corresponding AMs (lines 5-7). Next, we compute AMs for all attributes, assuming a uniform prior distribution to cover starting nodes (lines 8-10).

Time Complexity Analysis. One attribute can have at most t such AMs with another attribute. Thus, the overall time complexity of CoP is $O(n^2 t^8 \log(1/\tau))$ in the worst case, as there are n^2 combinations between attributes.

Space Complexity Analysis. One AM takes $O(t^2)$ space to keep transition probabilities. One attribute can have at most t such AMs with another attribute. Since there are n^2 combinations between attributes, the overall space complexity of CoP is $O(n^2 t^3)$ in the worst case.

Moreover, each attribute-specific AM can be computed in parallel, given the required inputs (e.g., priors and constraints). In Section 5.4, we report empirical runtime and memory usage across datasets, showing that CoP is practical to compute and benefits substantially from parallel execution. Since the curator-side computation is performed *one-time offline* and can be parallelized, polynomial-time complexity is acceptable in practice. We next describe the client-side perturbation procedure.

4.7 Perturbing Using CoP

Algorithm 5 (provided in Appendix H) summarizes the perturbation procedure of CoP. The key idea is to perturb attributes in a *stochastic traversal* over the dependency graph (DG) (See Appendix I for a discussion of why we adopt stochastic traversal).

In the algorithm, we maintain two sets: the set of perturbed attributes S_p and the set of unvisited attributes S_A. At each iteration,

we randomly select a starting attribute $\tilde{X} \in S_A$ (line 5), perturb it using its corresponding AM $M_r[\tilde{X}]$ (line 6), and add it to S_p (line 7). Next, we expand from the already perturbed but unvisited attributes. For each perturbed but unvisited attribute $X' \in S_p$, we visit its neighbors in the DG (lines 8-15). The *for-loop* (line 10) processes neighboring attributes in descending PMI order with respect to (X', l), where l is the perturbed output of X'. For each selected neighbor, we perturb it using the corresponding AM, conditioned on the parent output l (line 11), and add the newly perturbed attribute to S_p (line 12). We repeat this expansion until no further unperturbed neighbors remain; if S_A is still non-empty, we choose a new starting attribute from S_A and repeat. Finally, we reorder the perturbed outputs to match the original attribute order in the input record.

Time Complexity Analysis. The perturbation step of CoP has linear time complexity, $O(n)$, because each of the n attributes is perturbed once.

5 Evaluation

First, we describe the experimental setup in Section 5.1. We then evaluate CoP along three main aspects: (i) the effectiveness of coordinated perturbation in Section 5.2, (ii) robustness and practical configuration in Section 5.3, and (iii) scalability in Section 5.4.

5.1 Experimental Setup

5.1.1 Environment. We implement our experiments using Python 3.10 and run them on an AMD Ryzen Threadripper PRO 3955WX processor, 128GB of RAM, and 4TB of storage. We used the following open-source libraries for implementation: CVXPY[11] for convex optimization, HIGHS[21, 23] as the solver, and Pyitlib [16] for MI calculations.

5.1.2 Datasets. We select four real datasets that reflect practical attribute-collection scenarios where inter-attribute dependencies are known to be strong: biological attributes (CelebA [32]), demographic/socioeconomic surveys (Adult [3]), recommendation/interaction logs (KuaiRec [17]), and health surveys (Cardiovascular dataset (CVD) [37]). They cover diverse domains, sample sizes, and attribute alphabets, enabling the evaluation of CIL under heterogeneous dependency patterns.

The synthetic datasets provide controlled dependencies. We generate pairs of attributes (X', X'') using parametric relationships $X'' = f(X') + \eta$ with $f(\cdot) \in \{\sin(2\pi x),\ mx + c,\ x^2,\ \sqrt{c^2 - x^2}\}$, and $\eta \sim \mathcal{N}(0, \sigma^2)$. We vary σ^2 to control the dependency strength and discretize the variables into 10 bins. Table 2 in Appendix J summarizes the selected datasets. In these datasets, we have converted continuous attributes to discrete values via binning, as this work focuses on privacy leakages related to discrete/categorical data.

5.1.3 Baselines. To benchmark CoP, we compare against four state-of-the-art baseline mechanisms for *multidimensional* data collection under LDP, including both correlation-aware designs and standard composition-based solutions. Since our target setting is *service-oriented* (the server collects a perturbed report for every required attribute), we exclude attribute-sampling solutions (e.g., RS/RS+FD [41]) that report only a subset of attributes per user.

JRR [31] - Joint Randomized Response introduces *correlated perturbations* (via a joint randomization model) to reduce estimator variance while maintaining the same privacy guarantee as classical randomized response.

GRF [44] - GRF is a correlation-aware dimensionality-reduction baseline. It constructs an attribute-dependency graph using a correlation metric (e.g., Pearson correlation coefficient (PCC)) with a threshold to identify connected components. The privacy budget is allocated per component rather than over all attributes, reducing over-conservatism when dependencies are sparse. We report results for two threshold values (PCC=0.5 and PCC=0.3, weak and moderate correlation [1]) to reflect sensitivity to graph construction.

ITS [6] - ITS models correlation-induced privacy leakage in correlated data and provides a privacy/utility-aware calibration under common two-level perturbation protocols (e.g., GRR/OUE-style mechanisms). We use ITS as a correlation-aware baseline for multidimensional collection when dependencies are known or can be estimated from public/historical data.

SPL [41] - Split budget (SPL) is a standard multidimensional LDP baseline that perturbs *all* n attributes independently while allocating the total privacy budget across attributes via sequential composition [14] (e.g., ε/n per attribute under an overall ε). Although SPL is not correlation-aware, it is widely used and provides a strong reference point for the privacy-utility trade-off under full reporting. We implement SPL with Generalized Random Response (GRR) [41] in our evaluation.

5.1.4 Evaluation Metrics. Next, we present the metrics used in evaluations.

Total DIL - Total direct information leakage quantifies the sum of the DILs of each attribute, which usually contributes to data utility. We can interpret Total DIL as an attribute-level utility measure similar to $\sum_{k\in[n]} Utility(X_k)$. Here, $Utility(X_k)$ denotes the utility function defined in (10). Therefore, Total DIL is an information-theoretic measure of population-level utility of reported data.

$$\text{Total DIL} = \sum_{i\in[n]} I(X_i; Y_i), \tag{23}$$

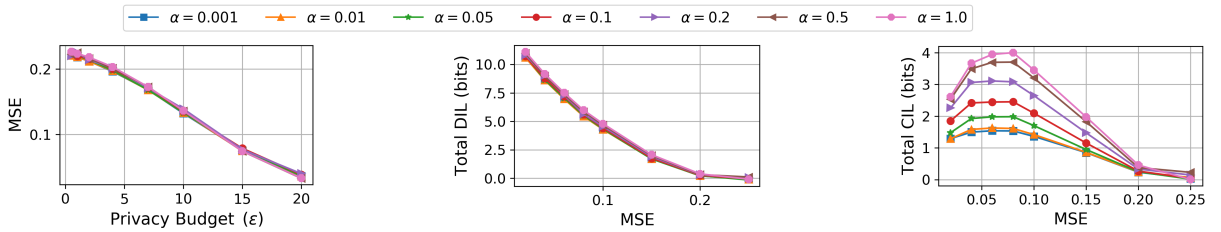
where Y_i is the perturbed value of X_i attribute. We empirically evaluate Total DIL using *Pyitlib* library.

Total CIL - Total correlation-induced information leakage quantifies the sum of the CILs for each attribute. Total CIL indicates the amount of unnecessary information leakage (inferable via correlations) that happens in data reporting.

$$\text{Total CIL} = \sum_{k\in[n]} \sum_{i\in[n],\ i\neq k} I(X_k; Y_i | Y_k, Y_1, \ldots, Y_{i-1}), \tag{24}$$

where Y_i is the perturbed value of X_i attribute. We empirically evaluate Total CIL using *Pyitlib* library.

MSE - In addition to the information-theoretic utility measurement 'Total DIL', we use mean squared error (MSE) between actual and reported data (i.e., deviation). In particular, MSE quantifies the average deviation between the original and reported multidimensional records over the population; hence, lower MSE indicates better utility. This complements (10), where Utility(X_k) is defined as an attribute-level expected utility, while MSE provides a record-level average distortion measure. Let J_{num} and J_{cat} be numeric and categorical attribute sets. Here, $n = |J_{\text{num}}| + |J_{\text{cat}}|$ as n



(a) (b) (c)

Figure 4: Evaluating the effect of the coordination parameter α in (20) on CoP using a synthetic dataset with $\sigma = 0.5$. Decreasing α strengthens coordination between perturbed outputs, leading to higher redundancy across reports and lower Total CIL**.**

is total attribute count. Then MSE of i^{th} record can be computed $\text{MSE}_i = \left(\sum_{j\in J_{\text{num}}} f_{num}(x_{ij}, y_{ij}) + \sum_{j\in J_{\text{cat}}} f_{cat}(x_{ij}, y_{ij})\right)/n$, where $f_{num}(x_{ij}, y_{ij}) = (x_{ij} - y_{ij})^2$ and $f_{cat}(x_{ij}, y_{ij}) = 0$ if $x_{ij} = y_{ij}$ and $f_{cat}(x_{ij}, y_{ij}) = 1$ otherwise. Then, MSE is

$$\text{MSE} = \frac{1}{R}\sum_{i\in[R]} \text{MSE}_i\,, \tag{25}$$

where R denotes the total number of records in the dataset.

5.1.5 Hyperparameters and Utility-Matrix Settings. Unless otherwise stated, we use $I_{\text{thr}} = 0.05$ and $PMI_{\text{thr}} = 0.1$ as the default hyperparameters for constructing the dependency graph in the main experiments. The sensitivity of these hyperparameters is evaluated in Section 5.3.2. For AM computation, we use a 0–1 loss matrix for categorical attributes and an ℓ_2-based loss matrix for discrete numerical attributes.

5.1.6 Attribute-Inference Attack Models. Total CIL is an information-theoretic metric that quantifies the additional privacy risk caused by inter-attribute dependencies. To evaluate the practical impact of this leakage, we perform attribute-inference attacks, where the adversary attempts to infer the *actual* value of a target attribute from a perturbed record. In our threat model, we assume a *worst-case* adversary that knows the perturbation mechanism and the underlying data distribution, but does not observe the exact ground-truth values of individual records.

Let $X = (X_1, \ldots, X_n)$ be actual user record, and let $Y = (Y_1, \ldots, Y_n)$ be the perturbed output. For each target attribute $k \in [n]$, we train an attacker $g_k : Y_{-k} \mapsto X_k$, where Y_{-k} denotes all released attributes except Y_k. Here, inference is primarily driven by inter-attribute dependencies. During training, we split the datasets into a *disjoint* training set and an evaluation set. We apply the mechanism to the training records to obtain perturbed reports Y, and train g_k on pairs (Y_{-k}, X_k).

Attackers - We instantiate g_k using two attackers: (i) *a neural TabTransformer-based attacker* ($\mathcal{A}_{Trans}$) and (ii) *a non-neural Maximum a Posteriori (MAP) attacker* ($\mathcal{A}_{MAP}$). See Appendix K for more details.

Metric - We measure attribute-inference success using *Macro-F1* [34]. For each attribute X_k, $k \in [n]$, we compute the Macro-F1 score between the attacker's prediction $g_k(Y_{-k})$ and the ground truth X_k, and then average across attributes.

$$\text{Inference Accuracy} = \frac{1}{n}\sum_{k\in[n]} \text{MacroF1}\big(g_k(Y_{-k}),\, X_k\big). \tag{26}$$

Lower 'Inference Accuracy' indicates **stronger resistance** to attribute inference.

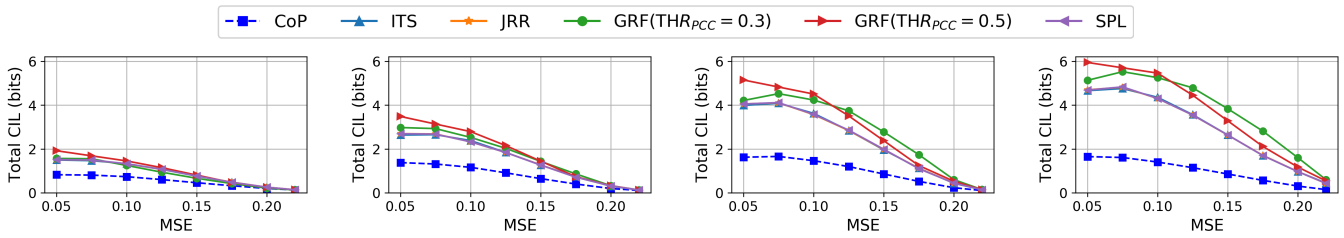
Next, we present the experimental results.

5.2 Effectiveness of Coordinated Perturbation

5.2.1 Impact of Coordinated Perturbation on CIL. First, we evaluate the impact of coordinated perturbation on CIL. α ($0 < \alpha \leq 1$) in (20) controls the amount of impact on an attribute's perturbed value from already perturbed parent attributes—the strength of coordination among attributes decreases as α increases. For the evaluation, we use a synthetic dataset with $\sigma = 0.5$ (average MI between attributes = 0.9 bits). First, we evaluate MSE versus privacy budget ε with different $\alpha = \{0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 1\}$ as shown in Figure 4a. As expected, changing α has a negligible effect on the utility of the data as MSE is *approximately equal* across α values. For further validation purposes, next we evaluate *Total DIL* versus *MSE*, which represents the amount of direct information leakage for given utility as depicted in Figure 4b. Since Total DIL is *approximately equal* across α, CoP achieves coordinated perturbation while retaining the utility-relevant information (i.e., direct information release).

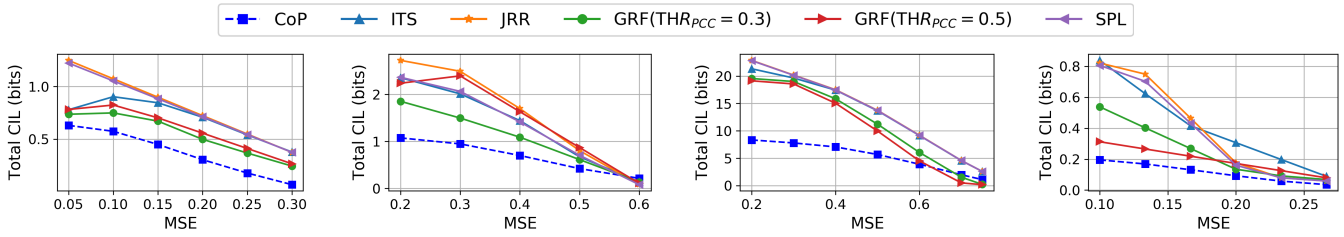
Next, we examine the *Total CIL* versus *MSE* across α values (Figure 4c). In this experiment, we vary the privacy budget, compute the resulting MSE and CIL, and then plot CIL against MSE. As α is reduced, the mechanism imposes stronger coordination among perturbed values. This increases the redundant information retained between the perturbed outputs, thereby making their distribution more closely aligned with the distribution of the original attributes. Therefore, Total CIL decreases as α decreases. Thus, these observations empirically validate the claim in Section 3—*calibrated coordination between perturbed outputs can mitigate CIL while retaining utility-relevant information.*

We further observe a non-monotonic relationship between Total CIL and MSE. Total CIL increases at intermediate MSE levels, but decreases when MSE becomes sufficiently large. At low MSE, the perturbed outputs are already highly informative about the true attributes, so leakage is dominated by direct disclosure, and there is limited additional CIL. At intermediate noise levels, direct



(a) Avg. MI = 0.3 bits **(b) Avg. MI = 0.6 bits** **(c) Avg. MI = 0.9 bits** **(d) Avg. MI = 1.1 bits**

Figure 5: Total CIL vs MSE for different dependency strengths (using the synthetic datasets).



(a) CelebA **(b) Adult** **(c) KuaiRec** **(d) CVD**

Figure 6: Total CIL vs MSE for real datasets.

information is partially reduced, while dependency information remains exploitable, resulting in higher CIL. In contrast, when the noise level is sufficiently high, both direct and dependency-based information are suppressed, which reduces Total CIL.

5.2.2 Effect of Dependency Strength. Next, we evaluate the performance of CoP under varying levels of inter-attribute dependency using synthetic datasets. Specifically, we generate four datasets with average mutual information values of 0.3 bits, 0.6 bits, 0.9 bits, and 1.1 bits between attributes, corresponding to $\sigma \in \{2, 1, 0.5, 0.1\}$, respectively (Figure 5).

The results show that CoP maintains low Total CIL across a broad range of dependency strengths. In particular, the *Total CIL* of the baseline mechanisms increases significantly as inter-attribute correlation strengthens. In contrast, the *Total CIL* of CoP increases only marginally with stronger attribute dependencies.

These results indicate that CoP becomes *significantly advantageous over existing mechanisms as the degree of inter-attribute dependency increases.*

5.2.3 Benchmark on Real Datasets. Next, we compare CoP against baseline mechanisms on real datasets, as shown in Figure 6. Overall, CoP consistently achieves lower CIL than the baselines across a wide range of utility levels.

We also observe that, on the CVD dataset, the CIL of CoP is closer to that of the baseline mechanisms (Figure 6d). This is expected because only a small number of attributes in CVD exhibit strong dependencies, which limits the extent to which coordinated perturbation can mitigate CIL (See heatmap of pairwise MI, Figure 14d in Appendix). *This is consistent with our synthetic experiments, which showed that the advantage of CoP grows as dependency strength increases* (Section 5.2.2).

These results suggest that CoP is particularly effective on real datasets with strong inter-attribute dependencies, while remaining competitive even when attribute dependencies are weak.

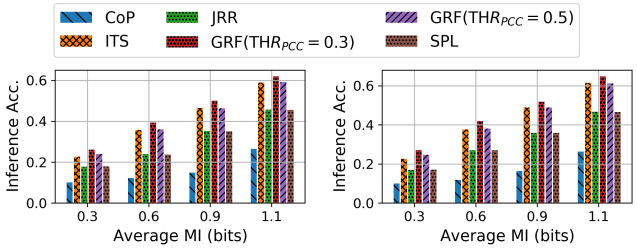
5.2.4 Attribute-Inference Attacks. So far, we have evaluated the reduction in CIL achieved by CoP through coordinated perturbation. To evaluate the practical implications of this reduction, we next conduct *attribute-inference attacks* on both synthetic and real datasets.

We first study the effect of dependency strength using synthetic datasets (Figure 7). The goal of this experiment is to examine how attack success varies as inter-attribute dependency increases. As expected, stronger dependencies lead to higher inference risk for all mechanisms. However, CoP remains more resistant to attribute-inference attacks, as the attacker achieves *consistently lower inference accuracy* than the baseline mechanisms.

Next, we evaluate inference accuracy vs MSE for the *Gender* attribute in the CelebA dataset. Here, we use only the perturbed data of correlated attributes as the attack input. *This experiment directly examines a natural concern for prior-knowledge-aware coordinated perturbation: because* Gender *is strongly correlated with other facial attributes, a mechanism that leverages dependency information could, in principle, make the hidden target attribute easier to infer from released correlated attributes.* As shown in Figure 8, CoP achieves the lowest attack success rate across a broad range of data utility levels ($0.1 \leq MSE \leq 0.4$). Therefore, even in this challenging high-correlation setting, CoP reduces the practical inferability of the hidden target attribute relative to the baseline mechanisms. *This attack-based result is consistent with our information-theoretic*

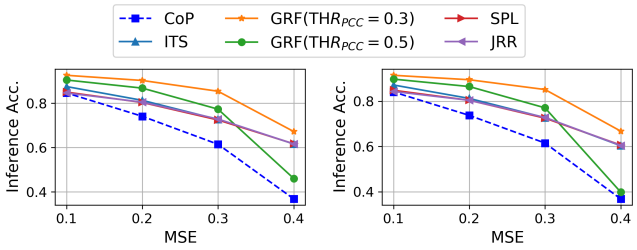
Table 1: Inference attack accuracy under $\mathcal{A}_{Trans}$ and $\mathcal{A}_{MAP}$ across datasets. Lower is better.

Mechanism	Adult		CelebA		KuaiRec		CVD	
	$\mathcal{A}_{Trans}$	$\mathcal{A}_{MAP}$	$\mathcal{A}_{Trans}$	$\mathcal{A}_{MAP}$	$\mathcal{A}_{Trans}$	$\mathcal{A}_{MAP}$	$\mathcal{A}_{Trans}$	$\mathcal{A}_{MAP}$
JRR	0.250	0.250	0.484	0.485	0.370	0.500	0.264	0.264
ITS	0.253	0.251	0.484	0.484	0.368	0.498	0.263	0.263
$\text{GRF}_{PCC=0.3}$	0.263	0.265	0.504	0.503	0.601	0.550	0.270	0.266
$\text{GRF}_{PCC=0.5}$	0.243	0.240	0.466	0.460	0.601	0.553	0.262	0.261
SPL	0.252	0.247	0.487	0.484	0.368	0.499	0.265	0.263
CoP (Ours)	**0.234**	**0.230**	**0.451**	**0.447**	**0.312**	**0.289**	**0.249**	**0.251**



(a) $\mathcal{A}_{Trans}$ Attacker **(b) $\mathcal{A}_{MAP}$ Attacker**

Figure 7: Attribute inference attacks on the synthetic datasets over different attribute dependency levels for $MSE = 0.1$.



(a) $\mathcal{A}_{Trans}$ Attacker **(b) $\mathcal{A}_{MAP}$ Attacker**

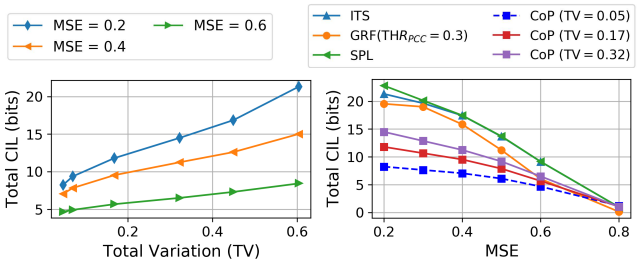
Figure 8: Attribute inference attack on learning gender attribute in CelebA. Here, inference accuracy corresponds to the *MacroF1* score for Gender.

analysis, which shows that CoP reduces CIL by increasing redundant information among perturbed attributes.

Finally, we conduct attribute-inference attacks on all real datasets at $MSE = 0.4$. Table 1 reports the inference accuracy for each mechanism. Across all settings, CoP achieves the lowest inference success, indicating reduced vulnerability to attribute-inference attacks. *Importantly, these attack-based results are consistent with the reductions in CIL observed in the earlier experiments.*

5.3 Robustness and Practical Configuration

5.3.1 Evaluation Under Imperfect Prior Knowledge. We next evaluate the robustness of CoP under imperfect prior distributional knowledge. To simulate imperfect priors, we estimate the distribution of the KuaiRec dataset using GRR [41] under different privacy



(a) Total CIL vs TV **(b) Total CIL vs MSE**

Figure 9: Evaluating CoP under imperfect prior knowledge for KuaiRec. (a) Total CIL vs TV. (b) Total CIL vs MSE of CoP under imperfect prior knowledge with baselines.

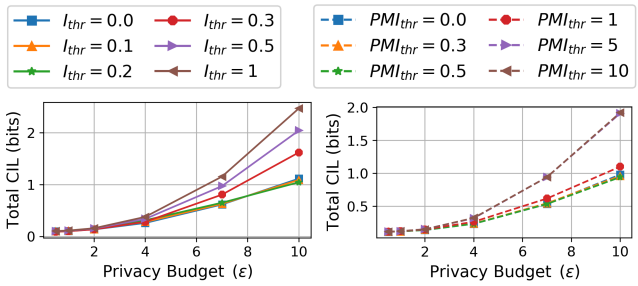
budgets. Smaller privacy budgets introduce stronger perturbation, resulting in larger deviations between the estimated and true distributions. We then use the estimated distributions to learn dependencies and optimize the atomic mechanisms (AMs) of CoP, and evaluate the resulting mechanism on the original dataset.

We quantify prior mismatch using total variation distance (TV) between the true and estimated distributions. TV ranges from 0 to 1, where values close to 0 indicate similar distributions and values close to 1 indicate substantial distributional mismatch.

Figure 9a shows Total CIL as a function of TV distance under three MSE levels. As TV distance increases, Total CIL also increases, since the prior distribution used for coordination becomes less representative of the true dependency structure. Nevertheless, this increase remains gradual, indicating that CoP is robust to moderate prior uncertainty.

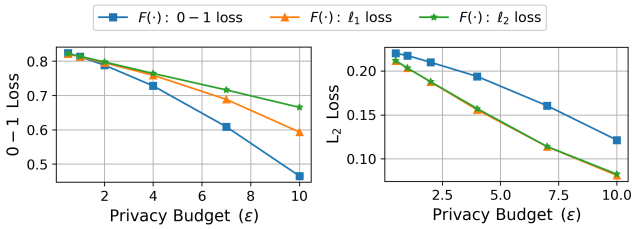
Figure 9b further compares CoP with the baseline mechanisms under imperfect prior knowledge. The results show that CoP continues to achieve lower Total CIL than the baselines across noisy-prior settings. This suggests that the proposed coordination strategy does not rely on exact prior knowledge and remains effective even when the available dependency information is imperfect. *Overall, these results support the practical applicability of CoP in service-oriented data collection scenarios where the prior distribution may be incomplete, noisy, or estimated from limited historical data.*

5.3.2 Sensitivity of CoP for hyperparameters I_{thr} and PMI_{thr}. Next, we analyze the sensitivity of CoP for hyperparameters (i.e., I_{thr} and PMI_{thr}). Increasing I_{thr} or PMI_{thr} systematically prunes weak and



(a) Varying I_{thr} and $PMI_{thr} = 0$ **(b) Varying PMI_{thr} and $I_{thr} = 0$**

Figure 10: Analyzing sensitivity of CoP for dependency graph (DG) construction by varying I_{thr} and PMI_{thr}.



(a) Categorical **(b) Discrete**

Figure 11: Evaluating CoP over different error functions $F(\cdot)$ for atomic mechanisms (AMs)—$0-1$ loss, ℓ_1 loss, and ℓ_2 loss—on the synthetic dataset with $\sigma = 0.5$. (a) Evaluate utility using $0-1$ Loss assuming dataset contains categorical fields, and (b) Evaluate utility using L_2 Loss assuming dataset contains discrete fields.

noisy dependencies from the dependency graph (DG), retaining only the most statistically significant edges. For example, on the Synthetic dataset with $\sigma = 0.5$ (average MI of 0.9 bits), increasing I_{thr} reduces the number of DG edges from 239 to 0.

To assess the mechanism's sensitivity to these thresholds, we examine how Total CIL varies with the privacy budget ε across different threshold settings, fixing one threshold at 0 while changing the other (Figure 10). We observe a range where performance **remains stable**: Total CIL remains tightly bounded and near-optimal over a broad low-threshold regime (e.g., $0 < I_{\text{thr}} < 0.2$ and $0 < PMI_{\text{thr}} < 1$). Total CIL begins to increase only when the thresholds become excessively large, at which point the DG loses the core informative dependencies required for effective coordinated perturbation.

5.3.3 Sensitivity of CoP for distortion matrices (E_U). As discussed in Section 4.3, the AM can be optimized for different utility criteria by changing the error function $F(\cdot)$ in (17). We evaluate CoP under three standard error functions—$0-1$ loss, ℓ_1 loss, and ℓ_2 loss—to examine how the choice of $F(\cdot)$ affects performance.

Figure 11 depicts loss versus privacy budget ε on the synthetic dataset with $\sigma = 0.5$. For categorical data, we evaluate utility using 0–1 loss, which is the most natural choice for categories (Figure 11a). For discrete data, we evaluate utility using ℓ_2 loss (Figure 11b).

We observe that using $F(\cdot) = 0$–1 loss yields the lowest error for categorical data, whereas mechanisms optimized under ℓ_1 and ℓ_2 loss perform better for discrete data. This is expected because the AM is explicitly optimized with respect to the distortion matrix E_U, which is constructed from the chosen error function $F(\cdot)$ (see (17) and (19)).

5.3.4 Recommended defaults for hyperparameters and distortion matrices. Based on our empirical sensitivity analysis, we recommend default settings for the hyperparameters and distortion matrices. Across the real datasets, the ranges $0 < I_{\text{thr}} < 0.1$ and $0 < PMI_{\text{thr}} < 1$ yield nearly identical *Total CIL*. This indicates that retaining all estimated dependency edges is unnecessary; filtering weak edges provides structural regularization, maintaining comparable CIL reduction while reducing the computational cost of AM optimization. Thus, we recommend using values within these ranges as a starting point and tuning them further to balance computational cost and performance. In our experiments, we set $I_{\text{thr}} = 0.05$ and $PMI_{\text{thr}} = 0.1$ by default. See Appendix M.3 for more recommended defaults for distortion matrices.

5.4 Scalability

Detailed runtime and memory results are provided in Appendix M.2. The results show that the main cost of CoP is the one-time offline computation of the policy parameters Θ, performed by the curator. This stage is parallelizable and benefits substantially from increasing the thread count. In contrast, the online user-side perturbation step is lightweight, requiring only milliseconds per record after Θ has been computed. Hence, CoP is suitable for practical deployment with offline calibration.

6 Discussion and Future Work

This section clarifies the modeling scope of CoP and discusses practical extensions to continuous domains, high-cardinality attributes, and richer dependency structures.

Continuous and high-cardinality domains. Although CoP is formulated for discrete attributes, it can be extended to continuous domains through discretization or binning. The binning granularity introduces a trade-off: finer bins preserve more information about the original values, but increase the effective domain size, perturbation-matrix size, optimization cost, and sample requirements for estimating conditional distributions. Thus, practical deployments should choose the binning strategy by balancing utility, statistical reliability, and computational cost.

Higher-order dependency modeling. CoP currently uses a single-parent dependency graph to capture dependency pathways for coordinated perturbation. In principle, CoP can be extended to multiple parents by replacing $Z_{k,l}$ in (15) with $Z_{k,L}$, where $Z_{k,L}$ denotes the conditional distribution of X_k given the joint realization L of its parent set $\text{pa}(k)$. The perturbation procedure in Algorithm 5 would also need to be modified so that the atomic mechanism for X_k is selected according to the perturbed outputs of all selected parent attributes. Such an extension could capture richer dependency structures and provide tighter control of CIL. However, the size of the conditional distribution grows with the product of the parent-domain sizes, increasing both estimation and optimization costs.

Therefore, the single-parent graph provides a practical trade-off between modeling accuracy, leakage control, and scalability.

Future directions. Extending CoP to better support continuous and high-cardinality domains, as well as more expressive dependency structures, is a promising direction for future work. In particular, adaptive discretization and sparse or bounded-order dependency graphs could improve applicability while preserving computational scalability.

7 Related Work

We review two relevant lines of work: (i) (L)DP and its information-theoretic interpretations, and (ii) LDP mechanisms for multidimensional correlated data.

7.1 (L)DP and Information-Theoretic Privacy

DP and LDP are standard privacy frameworks that overcome known limitations of heuristic anonymization methods such as k-anonymity and l-diversity [14, 42]. DP is typically used when a trusted curator can add noise centrally, whereas LDP is preferred when the data collector is untrusted and each user randomizes their data locally before reporting [20, 46].

A complementary line of work studies privacy from an information-theoretic perspective, connecting DP/LDP to quantities such as mutual information, maximal leakage, and related channel-capacity measures [9, 10, 26]. For instance, [10] derives information-leakage bounds implied by DP, while [9] relates the privacy budget ε to mutual information between a mechanism's input and output. However, most of this literature focuses on central DP and does not explicitly analyze the correlation-induced information leakage caused by inter-attribute dependencies under LDP [9, 10].

7.2 LDP Mechanisms in High-Dimensional Data

Multidimensional data often exhibit interdependencies, which can increase privacy leakage [28]. In the LDP literature, four high-level strategies are commonly used to handle multidimensional data while mitigating correlation-induced privacy leakage: (i) split the privacy budget among attributes, (ii) split the population, (iii) dimensionality reduction techniques, and (iv) joint perturbation.

Split the Privacy Budget Across Attributes. A common approach is to divide the total privacy budget across attributes and perturb each attribute separately. For example, SPL [41] often allocates the budget uniformly, which is simple but can be overly conservative in correlated data because it effectively assumes worst-case leakage. More recent work tightens privacy accounting by incorporating prior knowledge (e.g., universal norms [30], public datasets [3, 32], or information learned during data collection [43]) and analyzes leakage for multidimensional data under two-level perturbation mechanisms such as GRR and OUE [6, 12]. However, these methods focus on leakage accounting and budget allocation, rather than explicitly designing the perturbation channel to minimize CIL under utility constraints.

Split the Population. Instead of splitting the privacy budget, this method splits the population into disjoint groups to sample different attributes (e.g., random sampling (RS) [41], random sampling + fake data (RS+FD) [2], etc.). Recent work proposes sampling-based mechanisms for statistical learning in high-dimensional data, such as frequency estimation [41], and k-marginal estimation [36]. While effective for aggregate estimation, these techniques are not applicable to scenarios where we need valid unit-level user data, such as collecting user data to provide a service (e.g., contextual bandit recommender [47]).

Dimensionality Reduction. Another line of work reduces dimensionality before perturbation, for example, by learning dependency graphs using correlation metrics (e.g., LoPub [36], GRF [44]) or by transforming data into coefficient spaces (e.g., Fourier, Hadamard, or principal components) and adding calibrated DP noise to the transformed coefficients [25, 40]. These methods can improve scalability and utility for certain linear statistics, but they may struggle with complex nonlinear dependencies and generally do not explicitly optimize against CIL.

Joint Perturbation. Joint mechanisms perturb multiple attributes together to exploit structure and reduce noise. For example, PCKV [20] perturbs the key first (using ε_1) and then perturbs the value (using ε_2) conditioned on the perturbed key, while JRR [31] jointly perturbs multiple attributes and uses debiasing to improve aggregate estimation. Although effective in their target settings, these mechanisms are not readily generalizable to generic discrete multidimensional records and typically do not incorporate prior knowledge as an explicit component of mechanism design for reducing CIL.

Summary and Gap. Modern services often require collecting multidimensional user data for personalization and decision-making [39, 47]. Existing LDP approaches either use a partial user record (e.g., RS) or cannot mitigate CIL while preserving utility-relevant information. Moreover, prior knowledge is rarely incorporated as a principled component of mechanism design for mitigating CIL.

Motivated by these gaps, we propose a prior-knowledge-aware coordinated perturbation mechanism, CoP with information-theoretic grounding to mitigate CIL while preserving utility-relevant information.

8 Conclusion

Multidimensional user data is essential for many service-oriented applications, but dependencies between attributes can introduce *CIL*, leading to unintended disclosure beyond what is needed for utility. Motivated by data minimization, we proposed *CoP*, a coordinated perturbation mechanism that exploits prior dependency information to mitigate CIL while preserving utility-relevant *DIL*.

We formulated CoP as an optimization-based policy design problem, proved its ε-LDP guarantee, and showed that its atomic mechanisms can be computed offline and efficiently deployed on the user side. Through experiments on synthetic and real-world datasets, we show that CoP reduces CIL across multiple settings. We also show that CoP provides stronger resistance to MAP- and TabTransformer-based attribute-inference attacks. At the same time, it maintains a favorable privacy–utility trade-off compared with state-of-the-art LDP baselines. Overall, these findings demonstrate that coordinated perturbation with prior dependency awareness is a practical and effective way to improve the privacy–utility trade-off for correlated multidimensional data.

Acknowledgments

This research was partially supported by Meta Research Awards for Towards Trustworthy Products in AR, VR, and Smart Devices.

References

[1] Haldun Akoglu. 2018. User's Guide to Correlation Coefficients. *Turkish Journal of Emergency Medicine* 18, 3 (2018), 91–93. https://doi.org/10.1016/j.tjem.2018.08.001 Epub 2018 Aug 7.

[2] Héber H. Arcolezi, Jean-François Couchot, Bechara Al Bouna, and Xiaokui Xiao. 2021. Random Sampling Plus Fake Data: Multidimensional Frequency Estimates With Local Differential Privacy. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management* (Virtual Event, Queensland, Australia) *(CIKM '21)*. Association for Computing Machinery, New York, NY, USA, 47–57. https://doi.org/10.1145/3459637.3482467

[3] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. https://doi.org/10.24432/C5XW20 Accessed: 22 January 2026.

[4] S.P. Boyd and L. Vandenberghe. 2004. *Convex Optimization*. Number pt. 1 in Berichte über verteilte messysteme. Cambridge University Press, Cambridge, UK. https://books.google.com.au/books?id=mYm0bLd3fcoC

[5] California Civil Code Section 1798.100 Accessed 2024. California Civil Code Section 1798.100. Online; accessed 22 January 2026.

[6] Kah Meng Chong and Amizah Malip. 2024. Local Differential Privacy for correlated location data release in ITS. *Computer Networks* 255 (2024), 110830. https://doi.org/10.1016/j.comnet.2024.110830

[7] Thomas M. Cover and Joy A. Thomas. 2003. Entropy, relative entropy and mutual information. In *Wiley Series in Telecommunications*. John Wiley & Sons, Inc., New York, USA, 12–49.

[8] Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory* (2 ed.). John Wiley and Sons, Hoboken, NJ, USA. https://doi.org/10.1002/0471200611

[9] Paul Cuff and Lanqing Yu. 2016. Differential Privacy as a Mutual Information Constraint. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (Vienna, Austria) *(CCS '16)*. Association for Computing Machinery, New York, NY, USA, 43–54. https://doi.org/10.1145/2976749.2978308

[10] Anindya De. 2012. *Lower Bounds in Differential Privacy*. Springer Berlin Heidelberg, Berlin, Heidelberg, 321–338. https://doi.org/10.1007/978-3-642-28914-9_18

[11] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research* 17, 83 (2016), 1–5.

[12] Rong Du, Qingqing Ye, Yue Fu, and Haibo Hu. 2021. Collecting High-Dimensional and Correlation-Constrained Data with Local Differential Privacy. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, Online, 1–9. https://doi.org/10.1109/SECON52354.2021.9491591

[13] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. 2013. Local privacy and statistical minimax rates. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, Monticello, IL, USA, 1592–1592.

[14] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014).

[15] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. Association for Computing Machinery, New York, NY, USA, 1054–1067.

[16] Peter Foster. 2016. pyitlib: A library of information-theoretic methods for data analysis and machine learning. https://github.com/pafoster/pyitlib. Version 0.3.1, accessed 2026-03-01.

[17] Chongming Gao, Shijun Li, Wenqiang Lei, Jiawei Chen, Biao Li, Peng Jiang, Xiangnan He, Jiaxin Mao, and Tat-Seng Chua. 2022. KuaiRec: A Fully-Observed Dataset and Insights for Evaluating Recommender Systems. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. Association for Computing Machinery, New York, NY, USA, 540–550. https://doi.org/10.1145/3511808.3557220

[18] GDPR-Info. Accessed 2024. Article 5 GDPR - Principles relating to processing of personal data. https://gdpr-info.eu/art-5-gdpr/. Online; accessed 22 January 2025.

[19] Google. 2025. Protecting Your Google Assistant Privacy. https://safety.google/assistant/. Accessed: 2024-03-30.

[20] Xiaolan Gu, Ming Li, Yueqiang Cheng, Li Xiong, and Yang Cao. 2020. PCKV: Locally Differentially Private Correlated Key-Value Data Collection with Optimized Utility. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Online, 967–984.

[21] Julian Hall, Ivet Galabova, Qi Huangfu, Leona Gottwald, and Michael Feldmeier. 2026. HiGHS: Linear optimization software. https://github.com/ERGO-Code/HiGHS. GitHub repository, accessed 2026-01-06.

[22] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar S. Karnin. 2020. TabTransformer: Tabular Data Modeling Using Contextual Embeddings. *CoRR* abs/2012.06678 (2020), 1–7. arXiv:2012.06678 https://arxiv.org/abs/2012.06678

[23] Q. Huangfu and J. A. J. Hall. 2018. Parallelizing the dual revised simplex method. *Mathematical Programming Computation* 10, 1 (2018), 119–142. https://doi.org/10.1007/s12532-017-0130-5

[24] Jinyuan Jia and Neil Zhenqiang Gong. 2018. AttriGuard: A Practical Defense Against Attribute Inference Attacks via Adversarial Machine Learning. In *27th USENIX Security Symposium (USENIX Security 18)*. USENIX Association, Baltimore, MD, 513–529. https://www.usenix.org/conference/usenixsecurity18/presentation/jia-jinyuan

[25] Wuxuan Jiang, Cong Xie, and Zhihua Zhang. 2016. Wishart Mechanism for Differentially Private Principal Components Analysis. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, Phoenix, AZ, USA, 1730–1736. https://doi.org/10.1609/aaai.v30i1.10185

[26] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2016. Extremal mechanisms for local differential privacy. *J. Mach. Learn. Res.* 17, 1 (jan 2016), 492–542.

[27] Daniel Kifer and Ashwin Machanavajjhala. 2011. No free lunch in data privacy. In *Proceedings of ACM SIGMOD International Conference on Management of Data* (Athens, Greece) *(SIGMOD '11)*. Association for Computing Machinery, New York, USA, 12 pages.

[28] Daniel Kifer and Ashwin Machanavajjhala. 2011. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data* (Athens, Greece) *(SIGMOD '11)*. Association for Computing Machinery, New York, NY, USA, 193–204. https://doi.org/10.1145/1989323.1989345

[29] Robert Konig, Renato Renner, and Christian Schaffner. 2009. The Operational Meaning of Min- and Max-Entropy. *IEEE Transactions on Information Theory* 55, 9 (2009), 4337–4347.

[30] Yanan Li, Xuebin Ren, Shusen Yang, and Xinyu Yang. 2019. Impact of Prior Knowledge and Data Correlation on Privacy Leakage: A Unified Analysis. *IEEE Transactions on Information Forensics and Security* 14, 9 (2019), 2342–2357. https://doi.org/10.1109/TIFS.2019.2895970

[31] Ye-Zheng Liu, Shafizur Rahman Seeam, Yidan Hu, Rui Zhang, and Yanchao Zhang. 2025. Locally Differentially Private Frequency Estimation via Joint Randomized Response. *Proc. Priv. Enhancing Technol.* 2025 (2025), 242–260. https://api.semanticscholar.org/CorpusID:278636179

[32] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Santiago, Chile.

[33] Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, MA. Adaptive Computation and Machine Learning series.

[34] Juri Opitz and Sebastian Burst. 2021. Macro F1 and Macro F1. arXiv:1911.03347 [cs.LG] https://arxiv.org/abs/1911.03347

[35] Xuebin Ren, Liang Shi, Weiren Yu, Shusen Yang, Cong Zhao, and Zongben Xu. 2022. LDP-IDS: Local Differential Privacy for Infinite Data Streams. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 1064–1077. https://doi.org/10.1145/3514221.3526190

[36] Xuebin Ren, Chia-Mu Yu, Weiren Yu, Shusen Yang, Xinyu Yang, Julie A. McCann, and Philip S. Yu. 2018. LoPub : High-Dimensional Crowdsourced Data Publication With Local Differential Privacy. *IEEE Transactions on Information Forensics and Security* 13, 9 (2018), 2151–2166. https://doi.org/10.1109/TIFS.2018.2812146

[37] Svetlana Sulianova. 2021. Cardiovascular Disease Dataset. https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset.

[38] Latanya Sweeney. 2002. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 10, 5 (oct 2002), 557–570. https://doi.org/10.1142/S0218488502001648

[39] TikTok Newsroom. 2020. How TikTok recommends videos #ForYou. TikTok Newsroom. https://newsroom.tiktok.com/how-tiktok-recommends-videos-for-you?lang=en

[40] Guozhang Wang, Johannes Gehrke, and Xiaokui Xiao. 2011. Differential Privacy via Wavelet Transforms . *IEEE Transactions on Knowledge & Data Engineering* 23, 08 (Aug. 2011), 1200–1214. https://doi.org/10.1109/TKDE.2010.247

[41] Tianhao Wang, Jeremiah Blocki, Ninghui Li, and Somesh Jha. 2017. Locally Differentially Private Protocols for Frequency Estimation. In *26th USENIX Security Symposium (USENIX Security 17)*. USENIX Association, Vancouver, BC, 729–745.

[42] Teng Wang, Xuefeng Zhang, Jingyu Feng, and Xinyu Yang. 2020. A Comprehensive Survey on Local Differential Privacy toward Data Statistics and Analysis. *Sensors* 20, 24 (2020). https://doi.org/10.3390/s20247030

[43] Fei Wei, Ergute Bao, Xiaokui Xiao, Yin Yang, and Bolin Ding. 2024. AAA: An Adaptive Mechanism for Locally Differentially Private Mean Estimation. *Proc. VLDB Endow.* 17, 8 (May 2024), 1843–1855. https://doi.org/10.14778/3659437.3659442

[44] Jianhao Wei, Junyi Li, Yaping Lin, and Jin Zhang. 2021. LDP-Based Social Content Protection for Trending Topic Recommendation. *IEEE IoT Journal* 8, 6 (2021), 4353–4372. https://doi.org/10.1109/JIOT.2020.3026366

[45] Wikipedia contributors. 2024. F-divergence — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/F-divergence [Online; accessed 1-March-2025].

[46] Mengmeng Yang, Taolin Guo, Tianqing Zhu, Ivan Tjuawinata, Jun Zhao, and Kwok-Yan Lam. 2024. Local differential privacy and its applications: A comprehensive survey. *Computer Standards Interfaces* 89 (2024), 103827. https://doi.org/10.1016/j.csi.2023.103827

[47] Kai Zheng, Tianle Cai, Weiran Huang, Zhenguo Li, and Liwei Wang. 2020. Locally differentially private (contextual) bandits learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 1031, 11 pages.

[48] Xingyu Zhou and Wei Zhang. 2024. Locally private and robust multi-armed bandits. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '24)*. Curran Associates Inc., Red Hook, NY, USA, Article 236, 38 pages.

A Open Science

The artifacts are available in https://github.com/SandaruJayawardana/CoP-artifacts-PETS.

B Proof of Theorems

B.1 Proof for Theorem 3.1

Proof. Using the chain rule in two ways,

$$I(Y_{-k}; Y_k, X_k) = I(Y_k; Y_{-k}) + I(X_k; Y_{-k}|Y_k), \tag{27}$$
$$I(Y_{-k}; Y_k, X_k) = I(X_k; Y_{-k}) + I(Y_k; Y_{-k}|X_k). \tag{28}$$

Equating the right-hand sides and rearranging

$$I(X_k; Y_{-k}|Y_k) = I(X_k; Y_{-k}) - I(Y_k; Y_{-k}) + I(Y_k; Y_{-k}|X_k).$$

This completes the proof. □

B.2 Proof for Theorem 3.2

Proof. Assume the Markov chain $X_k \to X_{-k} \to Y_{-k}$ holds, i.e., $p(Y_{-k}|X_k, X_{-k}) = p(Y_{-k}|X_{-k})$. This follows because each Y_i is obtained by independently perturbing X_i for all $i \neq k$. Hence, by the Markov property,

$$I(X_k; Y_{-k}|X_{-k}) = 0.$$

Using the chain rule in two ways,

$$I(X_k; X_{-k}, Y_{-k}) = I(X_k; Y_{-k}) + I(X_k; X_{-k}|Y_{-k}),$$
$$I(X_k; X_{-k}, Y_{-k}) = I(X_k; X_{-k}) + I(X_k; Y_{-k}|X_{-k}) = I(X_k; X_{-k}).$$

Equating gives

$$I(X_k; Y_{-k}) = I(X_k; X_{-k}) - I(X_k; X_{-k}|Y_{-k}). \tag{29}$$

Since $I(X_k; X_{-k}|Y_{-k}) \leq H(X_{-k}|Y_{-k})$ and $H(X_{-k}|Y_{-k}) = H(X_{-k}) - I(X_{-k}; Y_{-k})$, we obtain

$$I(X_k; Y_{-k}) \geq I(X_k; X_{-k}) - H(X_{-k}) + I(X_{-k}; Y_{-k}). \tag{30}$$

Equivalently, using the nonnegativity of MI,

$$I(X_k; Y_{-k}) \geq \max\left\{0,\, I(X_k; X_{-k}) - H(X_{-k}) + I(X_{-k}; Y_{-k})\right\}. \tag{31}$$

For a fixed data distribution, the quantities $I(X_k; X_{-k})$ and $H(X_{-k})$ are fixed. Moreover, the utility requirement for X_{-k} imposes a lower bound on $I(X_{-k}; Y_{-k})$ (i.e., DIL). Therefore, the minimum value of $I(X_k; Y_{-k})$ is lower bounded by the utility requirements of X_{-k}.

This completes the proof. □

B.3 Proof of Theorem 4.1

Intermediate results. First, we establish intermediate results: Lemma B.1, Propositions B.2 and B.3 to prove Theorem 4.1.

Lemma B.1. *The function $f(x)$ defined as $f(x) = \frac{a_1x+b_1}{a_2x+b_2}$ is monotonic. Specifically, $f(x)$ is **increasing** with respect to x if $a_1b_2 - a_2b_1 \geq 0$, and $f(x)$ is **strictly decreasing** with respect to x if $a_1b_2 - a_2b_1 < 0$.*

Proof. We have $\frac{df(x)}{dx} = \frac{a_1b_2 - a_2b_1}{(a_2x+b_2)^2}$.

$$\text{Therefore,} \quad \frac{df(x)}{dx} \begin{cases} \geq 0 & ; \text{if } a_1b_2 - a_2b_1 \geq 0, \\ < 0 & ; \text{otherwise.} \end{cases}$$

This completes the proof of Lemma B.1. □

Proposition B.2. *Let $C = (c_1, \ldots, c_t)$ be a positive vector, and let $c_{\max} = \max_i c_i$ and $c_{\min} = \min_i c_i$. For fixed probability distributions $G = (g_1, \ldots, g_t)$ and $G' = (g'_1, \ldots, g'_t)$, define*

$$F(C, G, G') = \frac{\sum_{i\in[t]} c_i g_i}{\sum_{i\in[t]} c_i g'_i}.$$

Then there exists a vector $\tilde{C} \in \{c_{\min}, c_{\max}\}^t$ such that

$$F(\tilde{C}, G, G') \geq F(C, G, G').$$

Therefore, to compute the maximum value of $F(C, G, G')$ over all vectors C, it is sufficient to consider vectors whose entries take only the boundary values $c_{\min}$ and $c_{\max}$.

Proof. We know that $F(C, G, G') = \frac{\sum_{i\in[t]} c_i g_i}{\sum_{i\in[t]} c_i g'_i}$, where $c_i \in C, g_i \in G, g'_i \in G', 0 < c_i, g_i, g'_i \leq 1; \forall i \in [t]$. For any c_j, g_j and g'_j, we can rewrite $F(C, G, G')$ as

$$F(C, G, G') = \frac{c_j g_j + A}{c_j g'_j + B},$$

where $A = \sum_{i\in[t]\setminus\{j\}} c_i g_i$ and $B = \sum_{i\in[t]\setminus\{j\}} c_i g'_i$. From Lemma B.1, $F(\cdot)$ is a monotone function of c. The values of c which maximizes $F(C, G, G')$ can be determined as, $c = c_{\max}$ if $g_j B - g'_j A \geq 0$. Otherwise $c = c_{min}$.

Therefore, to compute the maximum value of $F(C, G, G')$ over C, it suffices to evaluate the function using a vector $\tilde{C}$ which consists only the boundary values c_{min}, c_{max} (i.e., $\tilde{C} \in \{c_{\min}, c_{\max}\}^t$).

This completes the proof of Proposition B.2. □

Proposition B.3. *Let $C = (c_1, \ldots, c_t)$ be a positive vector satisfying*

$$c_i \leq e^{\varepsilon} c_j, \quad \forall i, j \in [t],\ \varepsilon > 0,$$

and let $c_{\max} = \max_i c_i$ and $c_{\min} = \min_i c_i$. Let $G = (g_1, \ldots, g_t)$ and $G' = (g'_1, \ldots, g'_t)$ be probability distributions over the same alphabet, i.e.,

$$g_i, g'_i \geq 0, \quad \sum_{i\in[t]} g_i = \sum_{i\in[t]} g'_i = 1.$$

Then, for a given function

$$I(c_{min}, c_{max}, S, G, G') = \frac{c_{min} \sum_{i\in[t]\setminus S} g_i + c_{max} \sum_{j\in S} g_j}{c_{min} \sum_{i\in[t]\setminus S} g'_i + c_{max} \sum_{j\in S} g'_j},$$

c_{max} and c_{min} satisfy $c_{max} = e^{\varepsilon} c_{min}$ at the maximum of $I(\cdot)$ over all vectors C and subsets S.

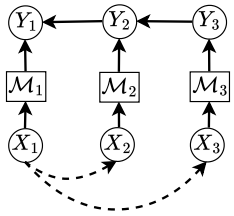


Figure 12: Markov chain relationship between variables for analyzing privacy leakage of X_1.

Proof. Since $c_i \leq e^{\varepsilon} c_j;\ \forall i, j \in [t]$, we have $c_{max} \leq e^{\varepsilon} c_{min}$. Therefore, we can write c_{max} as $c_{max} = k\ c_{min}$ using c_{min}, where $k \in (1, e^{\varepsilon}]$. Then $I(c_{min}, c_{max}, G, G')$ can be written in the form of $I(c_{min}, S, k, G, G')$ as

$$I(\cdot) = \frac{c_{min} \sum_{i\in[t]\setminus S} g_i + k c_{min} \sum_{j\in S} g_j}{c_{min} \sum_{i\in[t]\setminus S} g'_i + k c_{min} \sum_{j\in S} g'_j}. \tag{32}$$

Let us take $A = \sum_{j\in S} g_j$ and $B = \sum_{j\in S} g'_j$. We know that $\sum_{i\in[t]\setminus S} g_i + \sum_{j\in S} g_j = 1$ because it is a summation of a probability distribution. Then we have $\sum_{i\in[t]\setminus S} g_i = 1 - A$. In the same way we can write $\sum_{i\in[t]\setminus S} g'_i = 1 - B$. Then we can write (32) as

$$N(c_{min}, k, A, B) = \frac{c_{min}(1-A) + (kc_{min})A}{c_{min}(1-B) + (kc_{min})B}.$$

$N(c_{min}, k, A, B)$ is monotone over k. From Lemma B.1, we can maximize $N(\cdot)$ by setting k as, $k = e^{\varepsilon}$ if $A - B \geq 0$. Otherwise $k = 1$.

However, for any $a \in A$ and $b \in B$, $a < b \implies N(k, a, b) \leq 1$ and $a \geq b \implies N(\cdot) \geq 1$. Since we can change a and b by varying S, we can always set $a \geq b$ to maximize $N(\cdot)$. Therefore, $A - B \geq 0$ at the maximum $N(\cdot)$, and k should be $k = e^{\varepsilon}$.

Therefore, at the maximum $I(\cdot)$ the coefficients satisfy $c_{max} = e^{\varepsilon} c_{min}$. This completes the proof of Proposition B.3. □

Proof of Theorem 4.1.

Proof. Next, let us move to the proof of Theorem 4.1. First, we prove this theorem for a simple three-attribute scenario (i.e., X_1, X_2, X_3) and later generalize the results to n attributes.

We assume that X_1 is correlated with X_2 and X_3. Then, privacy leakage of X_1, L_{X_1} is given by,

$$\begin{aligned} L_{X_1} &= \ln \sup_{y_1\in\mathcal{Y}_1,\ y_2\in\mathcal{Y}_2,\ y_3\in\mathcal{Y}_3,\ x_1,x'_1\in\mathcal{X}_1} \frac{p(y_1,y_2,y_3|x_1)}{p(y_1,y_2,y_3|x'_1)}, \\ &= \sup_{y_1\in\mathcal{Y}_1,\ y_2\in\mathcal{Y}_2,\ y_3\in\mathcal{Y}_3,\ x_1,x'_1\in\mathcal{X}_1} \ln \frac{\sum_{x_2\in\mathcal{X}_2}\sum_{x_3\in\mathcal{X}_3} \frac{p(y_1,y_2,y_3,x_1,x_2,x_3)}{p(x_1)}}{\sum_{x_2\in\mathcal{X}_2}\sum_{x_3\in\mathcal{X}_3} \frac{p(y_1,y_2,y_3,x'_1,x_2,x_3)}{p(x'_1)}}, \end{aligned} \tag{33}$$

based on Definition 2.1.

Suppose the dependency graph of CoP is $X_3 \to X_2 \to X_1$. Then, we can build Markov chain dependencies between variables to analyze the leakage of X_1 as shown in Figure 12. Here, Y_1, Y_2, Y_3 and $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3$ denote the perturbed outputs and perturbation mechanisms (i.e., AMs) of X_1, X_2 and X_3 respectively. Since we want to analyze the privacy leakage of X_1 (i.e., L_{X_1}), we create a Markov relationship $X_1 \to X_2$ and $X_1 \to X_3$. Also, we have Markov relationships $Y_3 \to Y_2$ and $Y_2 \to Y_1$ as CoP performs coordinated perturbation by following the dependency graph.

Then, from Markov chain, we have $\sum_{x_2\in\mathcal{X}_2}\sum_{x_3\in\mathcal{X}_3} \frac{p(y_1,y_2,y_3,x_1,x_2,x_3)}{p(x_1)} = \sum_{x_2\in\mathcal{X}_2}\sum_{x_3\in\mathcal{X}_3} p(y_3|x_3)p(y_2|y_3,x_2)p(y_1|y_2,x_1)p(x_3|x_1)p(x_2|x_1) = p(y_1|y_2,x_1)\sum_{x_2\in\mathcal{X}_2} p(y_2|y_3,x_2)p(x_2|x_1)\sum_{x_3\in\mathcal{X}_3} p(y_3|x_3)p(x_3|x_1)$. Therefore, substituting this result into (33), we have

$$\begin{aligned} L_{X_1} = \sup_{x_1,x'_1\in\mathcal{X}_1} \Bigg(&\ln \sup_{y_1\in\mathcal{Y}_1,\ y_2\in\mathcal{Y}_2} \frac{p(y_1|y_2,x_1)}{p(y_1|y_2,x'_1)} \\ &+ \ln \sup_{y_2\in\mathcal{Y}_2,\ y_3\in\mathcal{Y}_3} \frac{\sum_{x_2\in\mathcal{X}_2} p(y_2|y_3,x_2)p(x_2|x_1)}{\sum_{x_2\in\mathcal{X}_2} p(y_2|y_3,x_2)p(x_2|x'_1)} \\ &+ \ln \sup_{y_3\in\mathcal{Y}_3} \frac{\sum_{x_3\in\mathcal{X}_3} p(y_3|x_3)p(x_3|x_1)}{\sum_{x_3\in\mathcal{X}_3} p(y_3|x_3)p(x_3|x'_1)} \Bigg). \end{aligned} \tag{34}$$

Next, let us consider each term on the RHS of (34).

- Term $\ln \sup_{y_1\in\mathcal{Y}_1,\ y_2\in\mathcal{Y}_2} \frac{p(y_1|y_2,x_1)}{p(y_1|y_2,x'_1)}$:

  Here,

$$\ln \sup_{y_1\in\mathcal{Y}_1,\ y_2\in\mathcal{Y}_2} \frac{p(y_1|y_2,x_1)}{p(y_1|y_2,x'_1)} \leq \varepsilon_1 \tag{35}$$

  as it denotes the privacy leakage of X_1's AM when perturbed output of X_2, $Y_2 = y_2$ (because each AM guarantees ε-LDP).

- Term $\ln \sup_{y_2\in\mathcal{Y}_2,\ y_3\in\mathcal{Y}_3} \frac{\sum_{x_2\in\mathcal{X}_2} p(y_2|y_3,x_2)p(x_2|x_1)}{\sum_{x_2\in\mathcal{X}_2} p(y_2|y_3,x_2)p(x_2|x'_1)}$:

  Next, let us consider the term

$$\ln \sup_{y_2\in\mathcal{Y}_2,\ y_3\in\mathcal{Y}_3} \frac{\sum_{x_2\in\mathcal{X}_2} p(y_2|y_3,x_2)p(x_2|x_1)}{\sum_{x_2\in\mathcal{X}_2} p(y_2|y_3,x_2)p(x_2|x'_1)}. \tag{36}$$

Suppose $\mathcal{X}_2 = \{x_{2,1}, \ldots, x_{2,t_2}\}$ be the alphabet of the attribute X_2 with $|\mathcal{X}_2| = t_2$. Then, we can define terms $p(y_2|y_3,x_2), p(x_2|x_1)$, $p(x_2|x'_1)$, $x_2 \in \mathcal{X}_2$, $y_2 \in \mathcal{Y}_2$, $x_1, x'_1 \in \mathcal{X}_1$ in vector forms as $C = (p(y_2|y_3,x_{2,1}), \ldots, p(y_2|y_3,x_{2,t_2}))$, $G_{2,1} = (p(x_{2,1}|x_1), \ldots, p(x_{2,t_2}|x_1))$, $G'_{2,1} = (p(x_{2,1}|x'_1), \ldots, p(x_{2,t_2}|x'_1))$. Then, we can write (36) as shown in (37) with removing ln.

$$F(C, G, G') = \sup_{y_2\in\mathcal{Y}_2,\ y_3\in\mathcal{Y}_3} \frac{C^T G_{2,1}}{C^T G'_{2,1}}. \tag{37}$$

Then, we can formulate (37) as a maximization problem as

$$\begin{aligned} \underset{C}{\textbf{maximize}} \quad & F(C, G_{2,1}, G'_{2,1}) = \frac{C^T G_{2,1}}{C^T G'_{2,1}} \\ \textbf{subject to} \quad & c \leq e^{\varepsilon_2}\tilde{c},\ \forall c, \tilde{c} \in C \qquad \textit{ε-LDP constraints,} \\ & 0 < c \leq 1,\ \forall c \in C \qquad \textit{probability constraints.} \end{aligned} \tag{38}$$

From Proposition B.2, the maximum of $F(\cdot)$ is given by $C \in \{c_{min}, c_{max}\}^{t_2}$. Here, $c_{min} = min(C)$ and $c_{max} = max(C)$. Then, we can be simplify $F(\cdot)$ as $I(c_{min}, c_{max}, G, G')$, where

$$I(c_{min}, c_{max}, S, G, G') = \frac{c_{min} \sum_{i\in[t_2]\setminus S} g_i + c_{max} \sum_{j\in S} g_j}{c_{min} \sum_{i\in[t_2]\setminus S} g'_i + c_{max} \sum_{j\in S} g'_j}. \tag{39}$$

Here, $S \subseteq [t_2]$ denotes the elements associate with c_{max}.

Next, we can show that the maximum of $I(\cdot)$ is given when $c_{max} = e^{\varepsilon_2} c_{min}$ using Proposition B.3. Therefore,

$$\begin{aligned} I(\cdot) &= \frac{c_{min} \sum_{i\in[t_2]\setminus S} g_i + c_{max} \sum_{j\in S} g_j}{c_{min} \sum_{i\in[t_2]\setminus S} g'_i + c_{max} \sum_{j\in S} g'_j} \\ &= \frac{\sum_{i\in[t_2]\setminus S} g_i + e^{\varepsilon_2} \sum_{j\in S} g_j}{\sum_{i\in[t_2]\setminus S} g'_i + e^{\varepsilon_2} \sum_{j\in S} g'_j} \end{aligned} \tag{40}$$

Let $C_{1,2} \in \{1, e^{\varepsilon_2}\}^{t_2}$. Then,

$$I(\cdot) = \frac{C_{1,2}^T G_{2,1}}{C_{1,2}^T G'_{2,1}}. \tag{41}$$

Therefore, $\ln \sup_{y_2 \in \mathcal{Y}_2,\ y_3 \in \mathcal{Y}_3} \frac{\sum_{x_2 \in \mathcal{X}_2} p(y_2|y_3,x_2)p(x_2|x_1)}{\sum_{x_2 \in \mathcal{X}_2} p(y_2|y_3,x_2)p(x_2|x'_1)}$ is bounded by

$$\leq \max_{C_{1,2}} \ln \frac{C_{1,2}^T G_{2,1}}{C_{1,2}^T G'_{2,1}}. \tag{42}$$

- Term $ln \sup_{y_3 \in \mathcal{Y}_3} \frac{\sum_{x_3 \in \mathcal{X}_3} p(y_3|x_3)p(x_3|x_1)}{\sum_{x_3 \in \mathcal{X}_3} p(y_3|x_3)p(x_3|x'_1)}$:

Similarly, we can simplify the term

$$\ln \sup_{y_3 \in \mathcal{Y}_3} \frac{\sum_{x_3 \in \mathcal{X}_3} p(y_3|x_3)p(x_3|x_1)}{\sum_{x_3 \in \mathcal{X}_3} p(y_3|x_3)p(x_3|x'_1)}$$

as it has a similar form compared to the term (36). Then,

$$\ln \sup_{y_3 \in \mathcal{Y}_3} \frac{\sum_{x_3 \in \mathcal{X}_3} p(y_3|x_3)p(x_3|x_1)}{\sum_{x_3 \in \mathcal{X}_3} p(y_3|x_3)p(x_3|x'_1)} \leq \max_{C_{1,3}} \ln \frac{C_{1,3}^T G_{3,1}}{C_{1,3}^T G'_{3,1}}. \tag{43}$$

Therefore, from (34), (35), (42) and (43), L_{X_1} can be written as

$$L_{X_1} \leq \varepsilon_1 + \max_{x_1, x'_1 \in \mathcal{X}_1,\ C_{1,j},\ j \in [3],\ j \neq 1} \sum_{j \in [3],\ j \neq 1} \ln \frac{C_{1,j}^T G_{j,1}}{C_{1,j}^T G'_{j,1}}. \tag{44}$$

Here, $C_{1,j} \in \{1, e^{\varepsilon_j}\}^{t_j}$, and $t_j = |\mathcal{X}_j|$ for all $j \in [3],\ j \neq 1$. Moreover, we can easily show that the longest Markov chain gives the maximum leakage. In the worst case, it includes all the attributes. Therefore, the generalized privacy leakage of any attribute $X_i \in X$ is given by

$$L_{X_i} \leq \varepsilon_i + \max_{x_i, x'_i \in \mathcal{X}_i,\ C_{i,j},\ j \in [n],\ j \neq i} \sum_{j \in [n],\ i \neq j} \ln \frac{C_{i,j}^T G_{j,i}}{C_{i,j}^T G'_{j,i}}. \tag{45}$$

Moreover, we know that $C_{i,j} \in \{1, e^{\varepsilon_j}\}^{t_j}$ and $\ln \frac{C_{i,j}^T G_{j,i}}{C_{i,j}^T G'_{j,i}} \leq \varepsilon_j$. Therefore, maximum privacy leakage is given by,

$$L_{X_i} \leq \varepsilon_i + \sum_{j \in [n],\ i \neq j} \varepsilon_j = \sum_{i \in [n]} \varepsilon_i \tag{46}$$

This completes the proof of Theorem 4.1. □

C Simplifying the LDP Constraints in (19)

We have simplified the LDP constraints by adopting the (utility-aligned) diagonal-dominance condition. Here, we assume that for each output symbol y, the maximum of $Q(y|x)$ over inputs is attained at $x = y$. This is a reasonable assumption as the utility of perturbed data is maximized when $x = y$ (i.e., lowest deviation). Therefore,

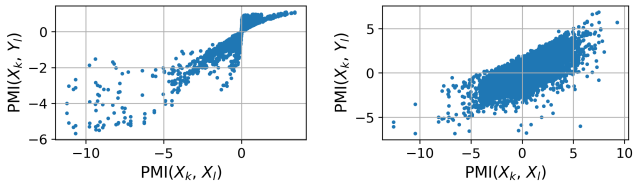
$$Q(Y_k = a|X_k = a) \geq Q(Y_k = a|X_k = a'), \tag{47}$$

for all $a, a' \in \mathcal{X}_k$[3]. Under (47), to guarantee $Q(\cdot|a) \preceq e^{\varepsilon} Q(\cdot|a')$ in (19), it suffices to bound the *max-to-min* ratio for each fixed $a \in \mathcal{X}_k$. Thus, the sufficient set of inequalities is

$$Q(a|a) \leq e^{\varepsilon} Q(a|a'), \quad \forall a, a' \in \mathcal{X}_k, \tag{48}$$

$$Q(a|a) \geq Q(a|a'), \quad \forall a, a' \in \mathcal{X}_k. \tag{49}$$

[3]We use (47) as a utility-aligned structural assumption (the truthful output is most likely).



(a) CelebA Dataset **(b) Adult Dataset**

Figure 13: Relationship between realization-level PMI before and after local perturbation on (a) CelebA and (b) Adult. For a fixed realization x_k and varying symbols $a \in \mathcal{A}_l$, each point compares $\text{pmi}(X_k = x_k; X_l = a)$ with $\text{pmi}(X_k = x_k; Y_l = a)$. The positive trend indicates that symbol-level dependency signals in the original data tend to persist after local perturbation. Here, X_l is perturbed with GRR mechanism at $\varepsilon = 5$.

C.1 Realization-Level Information Leakage

In our preliminary analysis, we study whether symbol-level dependency patterns persist after local perturbation. Specifically, we compare the PMI between original attribute values and the PMI between an original attribute value and the perturbed values of correlated attributes.

Let X_k and X_l be two correlated attributes, and let Y_k and Y_l denote their locally perturbed outputs, respectively. We assume a domain-preserving LDP mechanism, so that $X_k, Y_k \in \mathcal{A}_k$ and $X_l, Y_l \in \mathcal{A}_l$ share the same alphabets.

For a fixed realization $x_k \in \mathcal{A}_k$ and for each symbol $a \in \mathcal{A}_l$, we compare

$$\text{pmi}(X_k = x_k; X_l = a) \quad \text{and} \quad \text{pmi}(X_k = x_k; Y_l = a). \tag{45}$$

Figure 13 plots these quantities for the CelebA and Adult datasets. Each point corresponds to one symbol $a \in \mathcal{A}_l$. We observe a clear positive association between the two quantities: symbols that are strongly associated with $X_k = x_k$ before perturbation tend to remain relatively informative after perturbation as well. In particular, when $\text{pmi}(X_k = x_k; X_l = a)$ is large, $\text{pmi}(X_k = x_k; Y_l = a)$ also tends to be large.

This observation suggests that realizations with high PMI can reveal more information even after perturbation. On the other hand, if a realization has a low PMI, its potential information leakage is small. Therefore, we can identify realizations with high potential for CIL by comparing PMI against a threshold.

D Algorithm: Generating the Dependency Graph

Algorithm 1 summarizes the DG computational procedure.

Algorithm 1 Compute dependency graph (DG).

Input:
P_{X_i,X_j} $\forall i,j \in [n],\ i \neq j$. ▷ *Joint distributions.*
I_{thr}, PMI_{thr} ▷ *Hyperparameters (i.e., MI and PMI thresholds).*
1: E ▷ *A dictionary to keep edge information of attributes.*
2: **for all** X_i, X_j $i,j \in [n],\ i \neq j$ **do** ▷ *All possible attributes.*
3: $I \leftarrow$ Compute $I(X_i; X_j)$ using Eq. (3).
4: **for all** $l \in \mathcal{X}_i,\ k \in \mathcal{X}_j$ **do**
5: $i \leftarrow$ Compute $\text{pmi}(X_i = l; X_j = k)$ using Eq. (6).
6: **if** $i > PMI_{thr}$ and $I > I_{thr}$ **then**
7: $E[X_i, l] \leftarrow X_j$
8: **end if**
9: **end for**
10: **end for**
11: **return** E

E Algorithm: Computing Privacy Leakage of CoP

Next, we propose Algorithm 2 to compute a tight upper bound for the privacy leakage of CoP using Theorem 4.1. This algorithm first selects an attribute (i.e., X_k) at line 2 and traverses all possible realization pairs (x, x') over the alphabet of X_k at line 3. Next, we load the vectors G and G' and compute the element-wise ratio $G \oslash G'$. We then sort these ratios in descending order and store the resulting vector in Q (Lines 6–7). In line 8, we introduce two variables A and B to keep $\sum_{j\in S} g_j$ and $\sum_{j\in S} g'_j$, respectively. According to the Lemma. E.1, to maximize $\frac{C^T G}{C^T G'}$ for selected G, G', we should add g, g' to A, B, respectively, if $\frac{g}{g'} \geq \frac{1+A(exp(\varepsilon)-1)}{1+B(exp(\varepsilon)-1)}$. Therefore, we update A and B by checking the ratio $\frac{g}{g'}$ by traversing the Q vector as in lines 9-13. Next, the computed leakage is aggregated into l_{sum} at line 14, which maintains the total leakage for the selected $x, x' \in \mathcal{X}_k$. Next, we repeat the calculation for the next neighboring attribute. l_{max} keeps the maximum leakage over all attributes at line 16.

Time Complexity Analysis. Time complexity for computing privacy leakage is $O(n^2 t^3 \log t)$ since *for-loop* at line 2 have $O(t^2)$ time complexity and computing leakage for selected G, G' (lines 5-15) has time complexity of $O(t \log t)$ because of the sorting $G \oslash G'$ step (e.g., merge sort). Here, n and t denote the attribute count and the alphabet size, respectively.

Lemma E.1. *Let* $0 \leq A, B, g' \leq 1$, *and* $e^\varepsilon > 1$. *Then,*

$$\frac{1+(A+\sigma g')\lambda}{1+(B+g')\lambda} - \frac{1+A\lambda}{1+B\lambda} \geq 0 \iff \sigma \geq \frac{1+A\lambda}{1+B\lambda}.$$

Here, $\lambda = (e^\varepsilon - 1)$.

Proof. Let us take Δ as follows,

$$\Delta = \frac{1+(A+\sigma g')(e^\varepsilon-1)}{1+(B+g')(e^\varepsilon-1)} - \frac{1+A(e^\varepsilon-1)}{1+B(e^\varepsilon-1)}$$
$$\Delta = \frac{g'(e^\varepsilon-1)\left(\sigma+\sigma B(e^\varepsilon-1)-1-A(e^\varepsilon-1)\right)}{\left(1+(B+g')(e^\varepsilon-1)\right)\left(1+B(e^\varepsilon-1)\right)} \quad (50)$$

Since the denominator and $g'(e^\varepsilon - 1)$ are always positive, the sign of Δ only depends on $(\sigma + \sigma B(e^\varepsilon - 1) - 1 - A(e^\varepsilon - 1))$. Therefore,

$$\Delta \begin{cases} \geq 0 & ; \text{if } \sigma + \sigma B(e^\varepsilon - 1) - 1 - A(e^\varepsilon - 1) \geq 0, \\ < 0 & ; \text{otherwise.} \end{cases} \quad (51)$$

If $\sigma+\sigma B(e^\varepsilon-1)-1-A(e^\varepsilon-1) \geq 0$, we have $\sigma \geq \frac{1+A(e^\varepsilon-1)}{1+B(e^\varepsilon-1)}$. Therefore, we have, $\frac{1+(A+\sigma g')(e^\varepsilon-1)}{1+(B+g')(e^\varepsilon-1)} - \frac{1+A(e^\varepsilon-1)}{1+B(e^\varepsilon-1)} \geq 0 \iff \sigma \geq \frac{1+A(e^\varepsilon-1)}{1+B(e^\varepsilon-1)}$. This completes the proof. □

Algorithm 2 Compute privacy leakage of CoP.

Input:
P_{X_i,X_j} $\forall i,j \in [n],\ i \neq j$. ▷ *Joint distributions.*
ε ▷ *Per-attribute privacy budget.*
1: $l_{\max} \leftarrow -\infty$ ▷ *Variable to keep the maximum leakage.*
2: **for all** $k \in [n]$ **do**
3: **for all** $x, x' \in \mathcal{X}_k,\quad x \neq x'$ **do**
4: $l_{sum} \leftarrow 0$
5: **for all** $i \in [n] \setminus \{k\}$ **do**
6: $G \leftarrow P(X_i|X_k = x)$, $G' \leftarrow P(X_i|X_k = x')$
7: $Q \leftarrow \text{sort}(G \oslash G')$ ▷ *Descending order of element-wise ratios.*
8: $A, B \leftarrow 0$ ▷ *Variables to keep the summations.*
9: **for all** $q \in Q$ **do**
10: **if** $q \geq \frac{1+A(exp(\varepsilon)-1)}{1+B(exp(\varepsilon)-1)}$ **then**
11: $A \leftarrow A + g$ $B \leftarrow B + g'$ ▷ *Here, g and g' are the element of G and G' corresponds to q.*
12: **end if**
13: **end for**
14: $l_{sum} \leftarrow l_{sum} + \ln \frac{1+A(exp(\varepsilon)-1)}{1+B(exp(\varepsilon)-1)}$
15: **end for**
16: $l_{\max} \leftarrow \max\{l_{\max}, l_{sum}\}$
17: **end for**
18: **end for**
19: **return** $l_{\max}$

F Algorithm: Calibrating Privacy Budgets for CoP

Privacy budget calibration is a common approach for allocating a per-attribute privacy budget and maximizing utility under given privacy parameters in multidimensional spaces. Given a target per-attribute privacy leakage cap ε, the goal is to find the largest per-attribute budgets ε_i such that the privacy leakage of every attribute satisfies $\leq \varepsilon$. We develop Algorithm 3 to calibrate privacy budgets in CoP. Here, we assume that each attribute is perturbed with a single privacy budget.

From naive privacy budget splitting (SPL), we know that the privacy budget of each attribute should be $\geq \varepsilon/n$. Moreover, according to Theorem (4.1), privacy leakage of each attribute monotonically increases with its privacy budget. Therefore, we start the privacy budget of each attribute with ε/n (line 1) and increase $\tilde{e}$ iteratively until the privacy leakage of any attribute equals ε (lines 3-11).

Time Complexity Analysis - The privacy leakage analysis (Algorithm 2) repeats at maximum of $(\frac{\varepsilon-\varepsilon/n}{\tilde{e}})$ (*while-loop* at

line 3). Therefore, time complexity for calibrating per-attribute privacy budget is $O(\frac{n^2t^3\log t}{\tilde{e}})$ as each iteration has time complexity $O(n^2t^3\log t)$.

Algorithm 3 Compute per-attribute privacy budgets for CoP.

Input:
$P_{X_i,X_j}\ \forall i,j\in[n],\ i\neq j$. ▷ *Joint distributions.*
$\varepsilon,\quad \tilde{e}$ ▷ *Privacy budget and Step size.*

1: $\hat{\varepsilon}\leftarrow \varepsilon/n$ ▷ *Assign initial privacy budgets using naive privacy budget method SPL.*
2: $isOptimized \leftarrow False$ ▷ *To keep calibration status.*
3: **while** $not(isOptimized)$ **do**
4: $L\leftarrow$ Compute leakage using Algorithm 2 for $\hat{\varepsilon}$.
5: **if** $L<\varepsilon$ **then**
6: $\hat{\varepsilon}\leftarrow\hat{\varepsilon}+\tilde{e}$
7: **else** ▷ *i^{th} privacy budget is optimized.*
8: $isOptimized\leftarrow True$
9: Break
10: **end if**
11: **end while**
12: **return** $\hat{\varepsilon}$

G Algorithm: Computing Policy Parameters Θ

Algorithm 4 computes policy parameters Θ of CoP.

Algorithm 4 Compute CoP.

Input:
$P_{X_i,X_j}\ \forall i,j\in[n],\ i\neq j$ ▷ *Joint distributions.*
I_{thr}, PMI_{thr} ▷ *Hyperparameters for DG generation.*
ε ▷ *Privacy budget.*

1: $E\leftarrow$ Compute DG using Algorithm 1 for $P_{X_i,X_j}\ \forall i,j\in[n],\ i\neq j$ and I_{thr} and PMI_{thr}.
2: $(\varepsilon_1,\ldots,\varepsilon_n)\leftarrow$ Calibrate attributes' privacy budgets using Algorithm 3 for $P_{X_i,X_j}\ \forall i,j\in[n],\ i\neq j$ and ε.
3: M_e ▷ *To keep computed AMs corresponding to DG.*
4: M_r ▷ *To keep computed AMs for each attribute as starting node.*
5: **for all** $e\in E$ **do** ▷ *Visit every edges.*
6: $M_e[e]\leftarrow$ Compute AM for e by solving LP in Eq. (19).
7: **end for**
8: **for all** $X_i,\ i\in[n]$ **do** ▷ *Visit all attributes.*
9: $M_r[X_i]\leftarrow$ Compute AM for X_i by solving LP in Eq. (19). Here, Z is a uniform distribution.
10: **end for**
11: **return** $E,\ (\varepsilon_1,\cdots,\varepsilon_n),\ M_e,\ M_r$

H Algorithm: Perturbing Using CoP

Algorithm 5 summarizes the perturbation procedure of CoP.

Algorithm 5 Perturbation using CoP.

Input:
$X_1,\ldots,X_n$ ▷ *Actual attributes' values.*
M_e, M_r, E ▷ *AMs and dependency structure of CoP.*
$\varepsilon_1,\ldots,\varepsilon_n$ ▷ *Privacy budgets of each attribute.*

1: $S_A\leftarrow\{X_1,\ldots,X_n\}$ ▷ *Keep unvisited attributes.*
2: $S_P\leftarrow\{\}$ ▷ *Keep perturbed attributes.*
3: $Y_P\leftarrow\{\}$ ▷ *A map variable to keep perturbed values.*
4: **while** $|S_A|>0$ **do** ▷ *Check whether all attributes are visited.*
5: $\tilde{X}\leftarrow Randomly\ select\ attribute\ from\ S_A$.
6: $Y_p[\tilde{X}]\leftarrow$ Perturbed using $M_r[\tilde{X}]$.
7: $S_p.add(\tilde{X})$. ▷ *Add $\tilde{X}$ to the perturbed set.*
8: **for all** $X'\in S_p\cap S_A$ **do** ▷ *Visit unvisited perturbed attributes.*
9: $l\leftarrow Y_p[X']$. ▷ *Get perturbed value of X'.*
10: **for all** $\hat{X}\in neighbours(X',l)\setminus S_p$ **do** ▷ *Visit all unperturbed child nodes of $\tilde{X}$.*
11: $Y_p[\hat{X}]\leftarrow$ Perturbed using $M_e[\hat{X},l]$.
12: $S_p.add(\hat{X})$. ▷ *Add $\hat{X}$ to perturbed set.*
13: **end for**
14: $S_A.remove(X')$. ▷ *Remove X' from unvisited set.*
15: **end for**
16: **end while**
17: $Y\leftarrow order(\text{values of } Y_p)$. ▷ *Order the perturbed value w.r.t. X.*
18: **return** Y

I Why Stochastic Traversal is Important in the Dependency Graph

A fixed traversal perturbs attributes in the same order for every record, which introduces a systematic *order bias*: attributes perturbed earlier can consistently influence those perturbed later, while later attributes never influence earlier ones. As a result, the privacy and utility impacts may become uneven across attributes, with some receiving a persistent advantage or disadvantage solely because of their position in the traversal order.

Stochastic traversal mitigates this issue by randomizing the dependency-consistent perturbation order across records, ensuring that no single attribute is always processed in the same position. This reduces sensitivity to an arbitrary fixed ordering, distributes coordination effects more evenly across attributes, and makes the resulting leakage structure less deterministic.

J Summary of Datasets

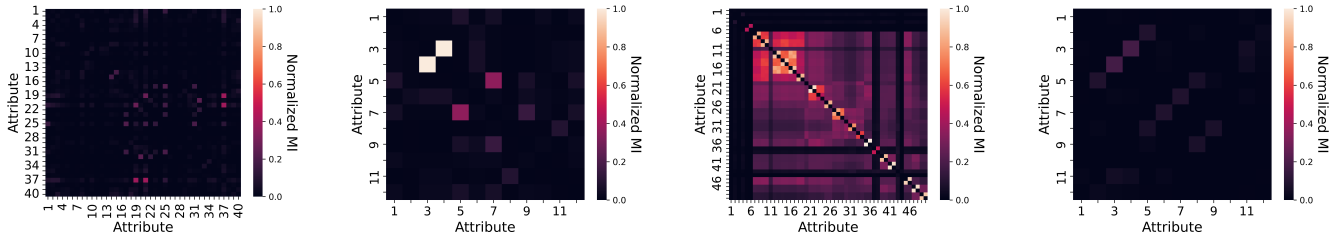
Table 2 provides a summary of the datasets used in our experiments.

Table 2: Summary of datasets used in experiments.

Dataset	#Samples	#Attr.	Alphabet	Notes
CelebA [32]	$>200K$	40	2	facial attributes
Adult [3]	$>45K$	12	2–41	income survey
KuaiRec [17]	$>10K$	50	2–40	Kuaishou logs
CVD [37]	$>68K$	12	2–22	health survey
Synthetic	$100K$	5	10	–

K Attribute Inference Attackers

TabTransformer attacker. We model a strong learned adversary using TabTransformer, which embeds each released categorical attribute as a token and applies a Transformer encoder to capture



(a) CelebA Dataset (b) Adult Dataset (c) KuaiRec Dataset (d) CVD Dataset

Figure 14: Pairwise mutual information in real datasets.

cross-attribute interactions, followed by a multi-class softmax head for predicting X_j [22]. We train a separate attacker g_j for each target attribute using auxiliary pairs $(\tilde{\mathbf{Y}}_{-j}, X_j)$ and report the attribute-averaged Macro-F1. We train for 10 epochs and select a batch size of 16384.

MAP attacker. We also consider a maximum-a-posteriori (MAP) attacker that learns a posterior model directly from auxiliary data, without requiring an explicit analytic model of the perturbation channel. Given auxiliary pairs, we estimate $\widehat{P}(X_j|\tilde{\mathbf{Y}}_{-j})$ and predict

$$\hat{x}_j \;=\; \arg\max_a \widehat{P}(X_j = a|\tilde{\mathbf{Y}}_{-j}).$$

In practice, we realize this rule using a smoothed categorical naive Bayes model trained on perturbed reports and true labels [33], following the standard auxiliary-data threat model for attribute inference [24].

Macro-F1. Macro-F1 is the unweighted mean of the per-class F_1 scores (one-vs-rest), giving equal weight to each class

$$\text{MacroF1} = \frac{1}{|C|}\sum_{c\in C}\frac{2P_cR_c}{P_c+R_c}.$$

Here, P_C and R_C are the precision and recall for class C.

L Pairwise Mutual Information in Real Datasets

Figure 14 depicts pairwise normalized mutual information between attributes of the real datasets.

M Additional Experimental Results

M.1 Utility–Privacy Trade-off

Figure 15a reports the MSE of CoP and the baseline mechanisms on the CVD dataset under different privacy budgets. We observe that CoP achieves lower MSE than ITS, SPL, and JRR across the considered privacy-budget range. However, GRF achieves lower MSE than CoP at the same privacy budget ε. This result should **not** be interpreted from MSE alone, because GRF also incurs *substantially larger measured privacy leakage*, as shown in Figure 15b.

Here, *measured privacy leakage* refers to the worst-case log-likelihood ratio between output distributions induced by different input values, analogous to the LDP definition. The measured privacy leakage of the baselines is computed using the theoretical privacy-leakage analysis in [6], while the privacy leakage of CoP is computed using Algorithm 2. Unlike the privacy budget, which is usually a design parameter, measured privacy leakage reflects the resulting privacy exposure after accounting for attribute dependencies.

To compare the mechanisms more directly, Figure 15c reports MSE as a function of measured privacy leakage. This view shows that *the lower MSE of GRF comes at the cost of substantially higher privacy leakage.* This is expected because GRF uses dependency measures such as PCC or MI to approximate correlated attributes and improve utility, but it does not explicitly characterize the privacy leakage induced by such correlations. As a result, GRF can incur larger measured privacy leakage for a given privacy budget. In contrast, CoP may have a higher MSE than GRF at a fixed privacy budget. However, for a given level of measured privacy leakage, CoP achieves lower MSE, indicating a better privacy-utility trade-off.

We also observe that the privacy leakage of CoP and ITS closely follows the allocated privacy budget, as shown in Figure 15b. In contrast, SPL and JRR show lower measured leakage because their perturbation strategies are more conservative in this setting.

Overall, CoP achieves a more favorable privacy-utility trade-off by maintaining lower MSE levels for a given level of privacy leakage.

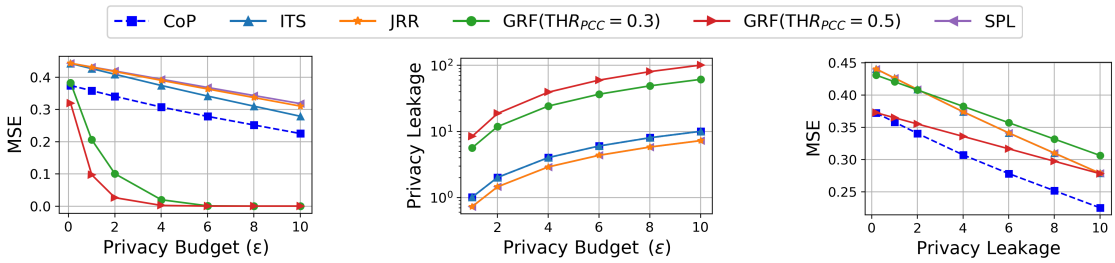
M.2 Empirical Time and Space Analysis

Table 3 summarizes the empirical run time and memory (Python object size) for each dataset. $t_{generation}$ denotes the run time of computing policy parameters Θ of CoP and this is multi-threadable computation. Increasing the thread count (TC) can significantly reduce run time. Specifically, CoP is computed one-time offline by the curator (server). In terms of storage requirements, the memory needed to store the AMs is small for all datasets except KuaiRec. The larger memory allocation for KuaiRec is due to its higher dimensionality and larger per-attribute alphabet sizes, which increase the dataset's size and the number of optimized AMs.

After computing policy parameters Θ of CoP perturbation $t_{perturb}$ (at user-level) takes only *milliseconds* to generate the perturbed output. These results indicate the feasibility of deploying CoP in real-world applications.

M.3 Recommended Defaults for Distortion Matrices

For the distortion matrices, we select loss functions according to the attribute type. For categorical attributes, we use 0–1 loss, since categories generally have no natural ordering and all incorrect



(a) MSE vs Privacy budget ε

(b) Privacy leakage vs Privacy budget ε

(c) MSE vs Privacy leakage

Figure 15: Utility-privacy analysis for CVD dataset.

Table 3: Runtime and Memory Analysis. Here, TC denotes the number of threads.

Dataset	# of Attributes	Per-attribute alphabet size	$t_{generation}$ (min) $TC = 1$	$t_{generation}$ (min) $TC = 10$	$t_{perturb}$ (sec) $TC = 1$	*Space* (MB)
CelebA	40	2	0.26	0.042	0.00021	1.33
Adult	12	11.16	3.53	1.04	0.0012	1.91
KuaiRec	50	13.74	82.5	14.63	0.0074	208.8
CVD	12	8.16	0.11	0.093	0.00010	0.69

reports can be treated equally. For discrete numerical attributes, we use ℓ_2 loss, since the magnitude of the error is meaningful and larger deviations should incur higher loss. Thus, the distortion matrix captures the task-relevant notion of distortion: exact matching for categorical attributes and distance-aware accuracy for numerical attributes. Other task-specific distortion matrices can also be used when the application requires a different distortion model, such as ordinal loss for ordered categories or weighted loss for errors with different service-level impacts.