DIFFERENTIALLY PRIVATE LEARNED INDEXES

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we study the problem of efficiently answering predicate queries for encrypted databases—those powered by Trusted Execution Environments (TEEs), allowing untrusted providers to process encrypted user data all without revealing sensitive details. A common strategy in conventional databases to accelerate query processing is the use of indexes, which map attribute values to their corresponding record locations within a sorted data array. This allows for fast lookup and retrieval of data subsets that satisfy specific predicates. Unfortunately, these traditional indexing methods cannot be directly applied to encrypted databases due to strong data-dependent leakages. Recent approaches use differential privacy (DP) to construct noisy indexes that enable faster access to encrypted data while maintaining provable privacy guarantees. However, these methods often suffer from significant data loss and high overhead. To address these challenges, we propose to explore learned indexes—a trending technique that repurposes machine learning models as indexing structures—to build more efficient DP indexes. Our contributions are threefold: (i) We propose a flat learned index structure that seamlessly integrates with differentially private stochastic gradient descent (DPSGD) algorithms for efficient and private index training. (ii) We introduce a novel noisy-max based private index lookup technique that ensures lossless indexing while maintaining provable privacy. (iii) We benchmark our DP learned indexes against state-of-the-art (SOTA) DP indexing methods. Results show that our method outperform the existing DP indexes by up to 925.6$\times$ in performance.

1 INTRODUCTION

Over the past decade, there has been a significant increase in the use of cloud computing for data storage and analysis. Its low cost, high availability, scalability, and ease of use make it an appealing option for businesses and scientific research. However, organizations that handle sensitive data, such as hospitals, banks, government agencies, and energy companies, may hesitate to use cloud services due to privacy concerns. The shared nature of cloud resources, coupled with potential vulnerabilities in the privileged software stack, has already led to various privacy breaches (Security, 2024). As a result, there is a critical need for robust measures to safeguard data-in-use privacy in the cloud environment. This is essential not only for policy compliance (HIPAA, 2003; GDPR, 2017), but also for maintaining public trust and advancing national priorities (The White House, 2022)

This need has given rise to a long line of research in an area known as *Encrypted Databases* (EDBs)(Eskandarian & Zaharia, 2017; Wang et al., 2021; Qiu et al., 2023), which enable untrusted cloud providers to manage and process encrypted user data. To achieve this, EDBs leverage Trusted Execution Environments (TEEs)(Costan & Devadas, 2016) to establish secure hardware enclaves on cloud machines, ensuring that any execution within these enclaves remains strongly isolated from the rest of the software stack, including the privileged OS and hypervisors. Users' data is only decrypted and processed inside these enclaves, and remains encrypted and integrity-protected whenever it leaves the enclave. Despite the strong encryption and isolation provided by TEEs, researchers have identified various side-channel threats associated with TEEs, leading to significant real-world data breaches (Kocher et al., 2020). For example, different query processing can result in distinguishable memory access patterns and read/write volumes, which attackers can exploit to reconstruct substantial portions of the data (Kellaris et al., 2016), even if it is placed inside a TEE or encrypted elsewhere. As a result, modern EDBs combine TEEs with oblivious algorithms, which implement branchless processing methods and pad the complexity to the worst-case maximum to ensure complete data independence.

While oblivious algorithms provide strong and provable privacy guarantees for today's EDBs, they clash with modern database optimization techniques, which often rely on leveraging data-dependent patterns for fine-grained performance improvements. A prime example is the use of indexes, which map attribute values (or keys) to their positions in a sorted array. Indexes enable rapid access to specific data subsets, reducing the need for frequent full table scans and minimizing excessive I/Os. Unfortunately, this promising technique is not directly compatible with EDB's privacy guarantees as they can leak exact information about the data distribution. As a result, EDBs must sequentially load all encrypted data into the TEE for every query processing, even when only a small portion is needed. To bridge this gap, Roy et al. (2020) introduced the concept of DP indexes, which distort exact key-position mappings with DP noise and and use these noisy mappings to privately index encrypted data. For instance, if the true index range for an attribute value is $[v_0, v_1]$, the DP index would produce $[\tilde{v}_0, \tilde{v}_1]$, where the endpoints $\tilde{v}_0$ and $\tilde{v}_1$ are distorted by DP noise. While this approach provides strong privacy guarantees, it can lead to significant data loss. Since the DP noise is symmetric, it is possible that $[v_0, v_1] \setminus [\tilde{v}_0, \tilde{v}_1] \neq \emptyset$, and when noise is large, the ranges may not overlap at all, such as $[v_0, v_1] \cap [\tilde{v}_0, \tilde{v}_1] = \emptyset$. A recent approach (Wang et al., 2024) improves this by using one-sided DP noise to enforce that $\tilde{v}_1 > v_1$ and $\tilde{v}_0 < v_0$, which achieves lossless indexing. However, the large noise scale and repeated injections (distorting each value-position pair) can result in excessive extra data being fetched, and in certain cases where the selectivity is relatively high, it may lead to only marginally better performance than a full table scan.

Kraska et al. (2018) argue that indexes are inherently models, sparking a growing research area known as learned indexes (Wu et al., 2024), which repurpose machine learning (ML) models as database indexes. While most work on learned indexes focuses on performance, we see their unique potential for building private indexes. With private training techniques (Goodfellow et al., 2016), learned indexes could allow us to introduce a unified DP noise into the ML model, instead of adding multiple noises to distort each key-position pair and potentially lowering noise scales significantly. For example, privacy induced noise of DP training can be bounded by $O((\sqrt{N})^{-1})$ (Bassily et al., 2014), where N is the sample size (unique key-position pairs for learned indexes). In contrast, existing DP indexes have noise scales proportional to $O(\sqrt{N})$ (Roy et al., 2020). This leads to the fundmental research question of this work:

Can we leverage learned indexe techniques to build new DP indexes that are both efficient and lossless, overcoming the their traditional limitations?

To address this question, we initiate the first study on designing DP learned indexes. Our major contributions are as follows: (i) We identify a key challenge: classical learned index structures are difficult to integrate with private training techniques. To overcome this, we propose a new Flat Model Index (FMI) structure that seamlessly combines with DP gradient descent algorithms (DPSGD). (ii) To achieve lossless indexing, we introduce a novel *Report Noisy Max Error* mechanism that privately determines an upper bound on indexing error, which guides a pessimistic indexing to prevent data loss. (iii) We implement the entire flow and benchmark our method against existing DP index methods. Our evaluation shows that the proposed DP learned indexes provide strong lossless guarantees while reducing the overhead of the state-of-the-art (SOTA) lossless DP indexes (Wang et al., 2024) by up to $494\times$; (iv) Finally, we conclude our work with a forward-looking discussion on potential improvements and future directions, aiming to spark broader interest and inspire new ideas in the emerging field of DP learned indexes.

2 Background and Related Work

General notations for relational databases. We define *database instances* D as relational tables with attributes $\texttt{attr}(D)$, where each attribute $A \in \texttt{attr}(D)$ has a domain $\texttt{dom}(A)$. For a subset of attributes $\mathbf{A} = \{A_1, A_2, \ldots, A_k\}$, the combined domains are $\texttt{dom}(\mathbf{A}) = \prod_{i=1}^{k} \texttt{dom}(A_i)$. Given D, $A_i \subseteq \texttt{attr}(D)$, and a value $x \in \texttt{dom}(A_i)$, we define the *frequency (count)* of attributed value x in D as $F(x, D, A_i) = \sum_{t \in D \wedge t.A_i = x} \mathbb{1}$. The sorted set of frequencies for all attributed values of A_i, is defined as the histogram of A_i in D, denoted as $H(D, A_i) = \{c_k = F(x_k, D, A_i)\}_{\forall x_k \in \texttt{dom}(A_i)}$. In this work, we focus on linear predicate queries, denoted as $q_\phi(D)$, which retrieve tuples from D satisfying a predicate ϕ then compute aggregated statistics based on the fetched data. A predicate ϕ is a logical expression with conditions on attributes, formed using conjunctions ($\wedge$) or disjunctions ($\vee$). Each condition is a logical comparison such as $A_i = a$ or $A_j > b$.

EDB system model. We consider a standard EDB model (Zheng et al., 2017; Eskandarian & Zaharia, 2017) in a cloud environment with two entities (as shown in Figure 1): the service provider (SP), managing the cloud infrastructure including the TEE, and the data owner (DO), who securely outsources storage and processing of private data to the SP. In the standard EDB model, the SP is considered honest-but-curious (Paverd et al., 2014), meaning they follow the pre-defined EDB protocol without deviation but may attempt to learn sensitive information about the owner's data by observing execution transcripts.

To initiate the EDB, the TEE first creates an enclave on SP's cloud machine, generates encryption keys $(\mathsf{sk}, \mathsf{pk})$, keeps sk inside the enclave, and sends pk to the owner. The DO encrypts their data tuple-wisely using pk along with a result key Kr, then uploads the encrypted data and Kr to the SP. When the DO issues a query, the TEE loads, decrypts, and processes the data, reencrypts the result using decrypted $\mathsf{K_r}$, and sends it back to the DO for decryption. Moreover, we assume the EDB employs oblivious algorithms for data processing to prevent side-channel leakages. For predicate queries, an example of the oblivious processing algorithm is as follows: (i) Upon receiving the query, the TEE sequentially reads the entire dataset into the enclave and performs a linear scan, labeling each tuple as a "match" or "non-match" based on the predicate. To maintain obliviousness, the label is updated for every tuple, regardless of whether it matches the predicate; (ii) The TEE then makes a second pass over the labeled data to compute the desired statistics as specified by the query. Similarly, during the statistics computation, every tuple is accessed, including non-matching ones, for which a dummy read is applied to maintain obliviousness.

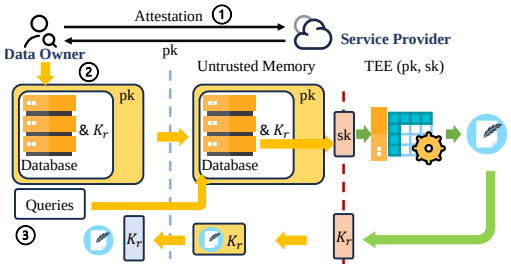


Figure 1: EDB system overview

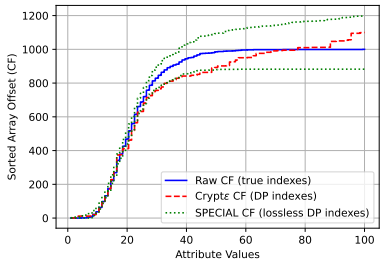


Figure 2: Indexes in the CF model

Index models. Given D sorted by A_i, and an attributed value $x_j \in \mathtt{dom}(A_i) = \{x_1, x_2, \ldots, x_n\}$, a index structure maps x_j to an interval $[v_0, v_1)$ such that $D[v_0, v_1]$ contains all tuples $t \in D$ that $t.A_i = x_j$. Kraska et al. (2018) suggest that indexes can essentially be abstracted as a cumulative frequency (CF) model. So that the lower and upper bound of $[v_0, v_1)$ mentioned above can actually computed by the CFs of $v_0 = \sum_{j=1}^{k} F(x_{j-1}, D, A_i)$, and $v_1 = \sum_{j=1}^{k} F(x_j, D, A_i)$. To better show this, we provide a visual example in Figure 2. As noted by Kraska et al. (2018), existing indexes (e.g., B^+ trees, etc.) are inherently data structures that compute or approximate such a CF curve.

Differential privacy, Gaussian mechanism, and DPSGD. DP is a well-established privacy framework that is rooted from the property of algorithm stability. Specifically, a randomnized mechanism $\mathcal{M}$ is said to satisfies (ϵ, δ)-DP if for any pair of neighboring databases D and D', differing by at most one tuple, and for all $\forall O \subset \mathcal{O}$, where $\mathcal{O}$ denotes all possible outputs, the following holds

$$\Pr[\mathcal{M}(D) \in O] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in O] + \delta$$

This definition ensures that the probability of producing a specific output does not change significantly (up to a multiplicative factor of e^{ϵ}) when any single tuple in the dataset is modified. The slack δ introduces a practical relaxation, allowing the privacy guarantee to fail with probability at most δ. Gaussian mechanism (Dwork et al., 2014) is a common method used to achieve (ϵ, δ)-DP. Specifically, given a function f that maps datasets to real numbers, the mechanism perturbs the output by adding noise sampled from $\mathcal{N}(0, \sigma^2)$, where σ is determined by both the desired privacy parameters (ϵ, δ) and the sensitivity of f. The sensitivity of f measures how much a single tuple can affect the output. DPSGD is an application of the Gaussian mechanism in machine learning training, where noise is added to the gradients during each iteration (Goodfellow et al., 2016) to protect the privacy of individual data points while still allowing model optimization.

DP indexes. To allow EDBs to use indexing techniques, Roy et al. (2020) introduced the first DP index design. Specifically, for each attributed value x_k, they consider to add independent

Laplace noises to generate a DP frequency count $\tilde{F}(v_k, D, A_i) = \min(|D|, \max(0, F(v_k, D, A_i) + \texttt{Lap}(\frac{1}{\epsilon})))$. The DP frequencies across all attribute values are then used to build a noisy CF curve, which inherently serves as an index structure (see example in Figure 2). While this design provides strong privacy guarantees, it can introduce significant errors due to symmetric Laplace noise—the noisy index range may be smaller than the true range, potentially excluding a large portion of matching tuples. To address this issue, Wang et al. (2024) propose using one-sided Laplace noise (strictly positive or negative) to build DP indexes. Specifically, they create two noisy CF curves: one overestimates the true CF with positive noise, while the other underestimates it with negative noise. To answer index lookup, the lower end point is taken from the underestimated curve, and the upper end point from the overestimated curve, ensuring all matching data is included. However, this approach can result in significant overhead, as the noisy index range can be much larger than the true range, leading to substantial I/O costs. We shown an example of this in Figure 2.

Learned indexes. Learned indexes are inherently ML models that fits the CF curves of attributed values. To gain high accuracy, the well-adopted method is to use a staged Recursive Model Index (RMI) (Kraska et al., 2018), where each stage's model takes a key as input and selects the next model in the hierarchy. The final stage then fits the key to a CF curve—estimating its position within a sorted array. As mentioned before, learned indexes have demonstrated various advantages including strong accuracy, efficient storage, and faster lookup times, making them a promising alternative for building DP indexes, which can potentially overcome the traditional limitations of noisy CF based DP indexes. An example of RMI structure can be found in § 3.2, Figure 3.a.

3 DP LEARNED INDEXES

In this section, we present the technical details of our proposed DP learned indexes. Before delving into the specifics, we first formulate the concrete problem to be addressed.

Problem formulation. In general we consider the problem of private training and inference of learned indexes on static data. Formally, given D sorted by $A_i \in \texttt{attr}(D)$, and $CF(D, A_i) = \{(x_i, y_i)\}_{i=1}^{N}$ to be the discrete CF of A_i over D and $\texttt{dom}(A_i) = \{x_1, x_2, ..., x_N\}$. We consider an idealized index is $\text{IDX}(x_i) = [y_{i-1}, y_i)$ for all x_i, and our goal is to build a private learned index model, PIDX, such that $\text{PIDX}(x_i) = [\tilde{y}(x_{i-1}), \tilde{y}(x_i))$, where $\tilde{y}(x_i)$ denotes the predicted position of x_i (vs. the true position of x_i to be y_i). In addition, we consider PIDX should satisfies the following:

- (ϵ, δ)**-DP at the tuple level.** Given $\epsilon > 0$, and $0 < \delta < 1$. For any neighboring databases D and D', differing by a single tuple and both sorted by the same attribute A_i, where $\texttt{dom}(A_i) \in D = \texttt{dom}(A_i) \in D'$, it holds for all outputs $O \subset \mathcal{O}$ that

$$\Pr[\text{PIDX}^D \in O] \leq e^{\epsilon} \cdot \Pr[\text{PIDX}^{D'} \in O] + \delta.$$

  This notion describes privacy at the tuple level, for instance the information related to the presence of each individual tuple by observing the index outcome, is bounded by DP. We do not consider key privacy, for example, an attacker might know whether a specific key x_j is included in $\texttt{dom}(A_i) \in D$. This aligns with the privacy guarantees of existing DP index approaches (Roy et al., 2020; Wang et al., 2024) and the logic of traditional indexes, where a lookup should abort if a key is not present. Note that tuple-level privacy can also be extended to user-level privacy. If a user owns multiple tuples in the dataset and we wish to preserve privacy at the user level, we can achieve this by applying the group privacy (Dwork & Rothblum, 2016) mechanism with appropriately adjusted privacy parameters.
- **Lossless indexing.** $\forall x_i$ it holds that $\text{IDX}(x_i) \setminus \text{PIDX}(x_i) = \emptyset$. This means that for records that is selected by the idealized index will also be included in the private index.
- **Small overhead.** If PIDX is lossless, the overhead for each lookup of x_i is given by $|\text{PIDX}(x_i)| - |\text{IDX}(x_i)|$, which represents the number of extra records selected by PIDX compared to the ideal index. Our objective is to minimize this overhead.

In the following sections, we present our approach to designing the desired DP learned index structure. We begin by discussing the potential challenges in Section 3.1, followed by proposed building block solutions 3.2 3.3 to address these challenges. Finally, we provide an integrated view 3.4 of these building blocks within the context of the EDB's workflow.

3.1 CHALLENGES

Challenge 1. DP training of classical RMIs is hard. A key challenge in designing DP learned indexes is that the hierarchical training procedure required by classical RMIs can be hard to integrate with private training algorithms like DP-SGD. Specifically, let $CF = \{(x_i, y_i)\}_{i=1}^N$ to be the ground truth CF for building indexes, and consider an RMI with ℓ stages, one root model, and M_ℓ sub-models at stage ℓ. The root model is denoted as $f_0(x; \theta_0)$, where θ_0 are the model parameters. A sub-model at stage ℓ, indexed by k, is denoted as $f_\ell^{(k)}(x; \theta_\ell^{(k)})$, with parameters $\theta_\ell^{(k)}$. The root model will be trained first by minimizing the following loss function:

$$L_0(\theta_0) = \frac{1}{N} \sum_{i=1}^{N} \left(f_0(x_i; \theta_0) - y_i \right)^2 .$$

Subsequently, for stages $\ell \geq 1$, each sub-model $f_\ell^{(k)}(x; \theta_\ell^{(k)})$ is trained independently, only after all models from the previous stage $\ell - 1$ have been trained. The loss function for the sub-model is

$$L_\ell^{(k)}(\theta_\ell^{(k)}) = \frac{1}{|CF_\ell^{(k)}|} \sum_{(x_i, y_i) \in CF_\ell^{(k)}} \left(f_\ell^{(k)}(x_i; \theta_\ell^{(k)}) - y_i \right)^2 ,$$

where $CF_\ell^{(k)}$ represents the data routed to the k-th sub-model at stage ℓ, and is defined as:

$$CF_\ell^{(k)} = \left\{ (x_i, y_i) \in CF \;\middle|\; \left\lfloor \frac{M_\ell f_{\ell-1}(x_i; \theta_{\ell-1})}{N} \right\rfloor = k \right\} .$$

To incorporate DP-SGD into the aforementioned training process, independent DP noise must be added at each stage, however, this can potentially lead to significant error accumulation in the final-stage models. Moreover, tracking and managing the privacy budget across different stages becomes challenging. As such, we will need to *redesign a learned index architecture that seamlessly integrates with DP-SGD techniques.*

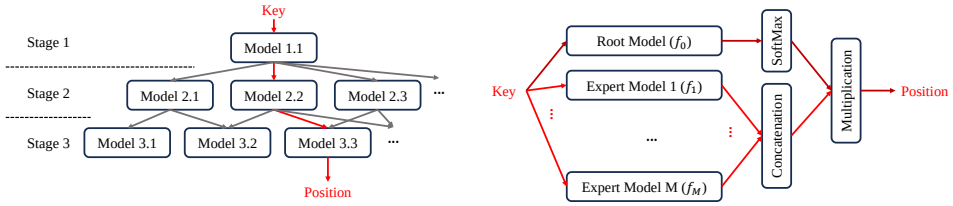
Challenge 2. Missing data can still occurs with learned indexes. Even after addressing the aforementioned challenge and creating a DP learned index that can be seamlessly trained with DP-SGD, the learned index can still suffer from the potential issue of missing data. Predicted indexing ranges could be smaller than the true range, causing some data to be missed. Therefore, we must also *explore how to ensure lossless indexing without compromising privacy.*

3.2 FLAT MODEL INDEX (FMI) AND DP TRAINING

To address the challenge that traditional RMI structures can be hard to compatible with existing DP training techniques, such as DP-SGD, we draw inspiration from the Mixture of Experts framework to introduce FMIs. Unlike the hierarchical structure of classical RMI, which necessitates independent training of each sub-model, the FMI consists of multiple models that can be trained at the same time, which significantly simplifies integration with DP-SGD—as one can derive a unified gradient during training to which DP noise can be applied. Specifically, we consider a FMI consists of a root model $f_0(x; \theta_0)$ and M expert-models $\{f_k(x; \theta_k)\}_{k=1}^M$ operating at the same level (Figure 3.b). Given a key $x_i \in CF = \{(x_i, y_i)\}_{i=1}^N$, the index prediction of FMI is then given by

$$\hat{y}(x_i) = \sum_{k=1}^{M} w_k(x_i; \theta_0) \cdot f_k(x_i; \theta_k),$$

where $\mathbf{w}(x_i; \theta_0) = [w_1(x_i; \theta_0), \ldots, w_M(x_i; \theta_0)]$ denotes the weight vector generated by the root model. We say that FMI can be logically similar to RMI: In RMI, the the backbone logic is to distribute each key (attribute value) to a final stage model, which then individually predicts the range for that specific key. FMI can be logically formed as a two staged RMI, but intead of picking one model for prediction, multiple last stage models collaboratively predict the output for every key, with a root model $f_0(x; \theta_0)$ adjusting the contributions of these predictions through a weight vector $\mathbf{w}(x_i; \theta_0)$. Notably, if we generalize the root model to output a one-hot-encoded weight vector, the FMI effectively reduces to a two-stage RMI model.



(a) RMI architecture (b) FMI (our approach) architecture

Figure 3: Comparison of architectures between RMI and FMI.

Private training of FMI with DP-SGD. To enable simultaneous training of all components, the model parameters are concatenated into a unified parameter vector $\Theta = [\theta_0; \theta_1; \ldots; \theta_M]$, and the loss function is defined as $L(\Theta) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}(x_i) - y_i)^2$. During each training iteration, we sample a mini-batch $B \subset D$, where the per-example unified gradient is computed as

$$g_i = \nabla_\Theta \ell_i(\Theta) = [\nabla_{\theta_0}\ell_i(\Theta), \nabla_{\theta_1}\ell_i(\Theta), \ldots \nabla_{\theta_M}\ell_i(\Theta)]^\top,$$

such that the gradients of the root and the k-th expert model parameters are given, respectively, by

$$\nabla_{\theta_0}\ell_i(\Theta) = 2\left(\sum_{k=1}^{M} w_k(x_i;\theta_0)\cdot f_k(x_i;\theta_k) - y_i\right)\cdot\sum_{k=1}^{M}\left(\nabla_{\theta_0} w_k(x_i;\theta_0)\cdot f_k(x_i;\theta_k)\right),$$

$$\nabla_{\theta_k}\ell_i(\Theta) = 2\left(\sum_{k=1}^{M} w_k(x_i;\theta_0)\cdot f_k(x_i;\theta_k) - y_i\right)\cdot w_k(x_i;\theta_0)\cdot\nabla_{\theta_k} f_k(x_i;\theta_k).$$

In this way, a unified per-example gradient vector g_i can be obtained. Next, we apply DP-SGD to update the overall model parameters, and using the obtained unified gradients, shown as follows:

$$\Theta \leftarrow \Theta - \eta\left(\frac{1}{|B|}\left(\sum_{i\in B}\frac{g_i}{\max\left(1, \frac{\|g_i\|_2}{C}\right)} + \mathcal{N}\left(0, \sigma^2 C^2 I\right)\right)\right),$$

The above process can be summarized as follows: At each iteration, we sample a mini-batch $B \subset \{(x_i, y_i)\}_{i=1}^{N}$ for training. For each example in the mini-batch, the per-example gradient g_i is first clipped to a maximum ℓ_2-norm of C to bound the sensitivity. After that, the clipped gradients are averaged and perturbed with Gaussian noise$\mathcal{N}(0, \sigma^2 C^2 I)$, where I is the identity matrix corresponding to the dimensionality of Θ, and $\sigma \geq 2\varepsilon^{-1}C\sqrt{2\ln(1.25/\delta)}$ is the noise multiplier. Finally, the DP model update is applied to the overall model parameters with a learning rate η. Note that the above process is a direct application of DP-SGD, and thus it follows the same privacy guarantees as by DP-SGD. For completeness, we have also provided a derived privacy proof in A.1.

3.3 Noisy max based private index overestimation

We now address the second challenge of ensuring lossless indexing atop the DP-FMIs derived in the previous step. The key idea is to leverage the maximum inference error as a pessimistic upper bound to guarantee the correctness of every index search. Specifically, we first compute the inference errors, $e_i = |\hat{y}(x_i) - y_i|$, for all x_i, and subsequently determine the maximum error, $e_{\max} = \arg\max_{\{1,\ldots,N\}}(e_i)$. For any index lookup (e.g., assuming query with key x_j), we can then compute a pessimistic overestimated indexing range as $[\hat{y}(x_j) - e_{\max}, \hat{y}(x_{j+1}) + e_{\max})$. Since the true indexing range for x_j is $[y_j, y_{j+1})$, it holds that $\hat{y}(x_j) - e_{\max} \leq y_j$ and $y_{j+1} \leq \hat{y}(x_{j+1}) + e_{\max}$. As a result, the true range $[y_j, y_{j+1})$ is guaranteed to be completed contained within the predicted range $[\hat{y}(x_j) - e_{\max}, \hat{y}(x_{j+1}) + e_{\max})$, and thus no tuples are missing (lossless).

Although this method ensures lossless, it may introduce additional privacy concerns—while the predicted results on DP-FMI satisfy DP as they post-processing on DP models, the computation of $e_{\max}$ involves direct access to raw data, which violates DP. To resolve this issue, we propose an enhancement to the max error method by employing the *Report Noisy Max* (Ding et al., 2021) mechanism. This generates a DP-distorted max error, $\tilde{e}_{\max}$, which to be used as a private bound for overestimating the indexing ranges. The details are shwon in Algorithm 1.

Algorithm 1 Lossless private FMI inference

Input: (i) Given D sorted by A, and let $\{(x_i, y_i)\}_{i=1}^{N}$ to be the CF of attribute A. (ii) Privacy parameters: $\epsilon > 0$, Δ; (iii) A look up request (key), x_j.

if not exists $\tilde{e}_{\max}$ **then**
  Compute the lookup errors for all attribute values, $\forall i \in [1, N]$, $e_i = |\hat{y}(x_i) - y_i|$.
  Sample $z_1, z_2, ..., z_N \sim \text{Exp}(z, \frac{\epsilon}{2\Delta})$ i.i.d, where $\text{Exp}(z, \lambda) \leftarrow \lambda e^{-\lambda x}$ for all $z \geq 0$,
    and $\Delta = \max_{i, D \sim D'} |e_i^D - e_i^{D'}| \sim O(\frac{\sqrt{\log 1/\delta}}{N\epsilon})$ is the sensitivity of prediction errors.
  One-time release $\tilde{e}_{\max} \leftarrow \arg\max_{e_1, ..., e_N} (e_i + z_i)$.
for each lookup $k_i \in k_1, ..., k_m$ **do**
  output indexing range $[\hat{y}(x_j) - \tilde{e}_{\max}, \hat{y}(x_{j+1}) + \tilde{e}_{\max})$.

In summary, this method introduces non-negative exponential noise to each indexing error and selects the noisy maximum from these distorted errors. This ensures that $\tilde{e}_{\max} \geq e_{\max}$, thereby guaranteeing that using $\tilde{e}_{\max}$ to overestimate the indexing range preserves the lossless property. Moreover, since modifying a single tuple results in $|y_i^D - y_i^{D'}| \leq 1$, the sensitivity is bounded by $\Delta \leq \max_i |\hat{y}(x_i)^D - \hat{y}(x_i)^{D'}| + 1$. By applying the error bounds from the DP-SGD algorithm (Das et al., 2023; Bassily et al., 2014), the sensitivity Δ is actually asymptotically bounded by $O(\sqrt{\log \delta^{-1}/N\epsilon^2})$. For space concerns, the privacy proof of Algorithm 1 is deferred to A.2.

3.4 Putting it together

We now show how our PD learned index building blocks can be integrated into the EDB's workflow.

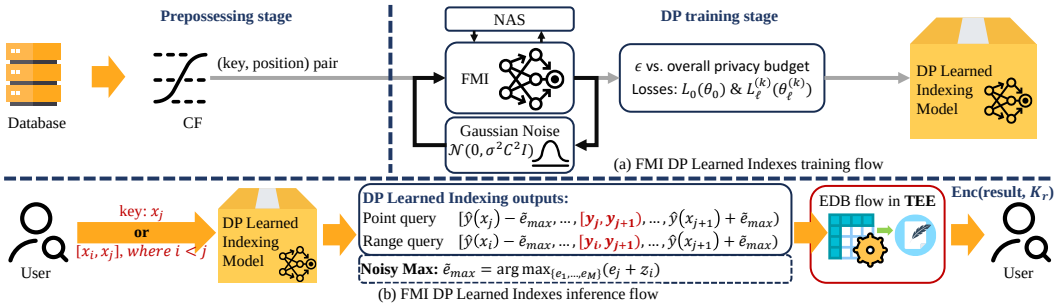


Figure 4: Our approach enables end-to-end training and inference for DP Learned Indexes.

Phase 1. On-premises training (Figure 4.a). Initially, the DO will train a DP learned index on-premises using their private data. The process begins by sorting the data based on an attribute of interest (e.g., frequently queried ones) and computing the corresponding CF curve. The DO then trains a DP-FMI using the CF as training data and derives the noisy error bound $\tilde{e}_{\max}$. For better outcomes, we say that one may adopt a Neural Architecture Search (NAS) (Liu et al., 2018a;b; 2022) process to identify an optimized FMI architecture. The search space may include parameters such as (i) the number of expert models, (ii) the number of hidden layers per model, and (iii) the number of neurons per layer. Finally, the DO uploads the trained DP-FMI, the private error bound $\tilde{e}_{max}$, and their encrypted data to the SP. As all uploaded objects are either DP-distorted or encrypted, they can be safely managed by the SP directly.

Phase 2. Runtime private indexing (Figure 4.b). We now discuss how DP-FMI is used to accelerate EDB predicate queries. Upon receiving a plaintext query, the TEE analyzes the predicate and performs an index lookup on the DP-FMI, which returns private intervals. The TEE then loads the corresponding encrypted data, decrypts it, and processes as specified by the query to compute the required statistics. Note that both the DP-FMI and $\tilde{e}_{\max}$ can be stored and used outside the TEE, as they have already been injected with DP noise. Thus, DP-FMI inference can be offloaded to the untrusted SP's software stack or external accelerators like GPUs, without compromising EDB's privacy guarantees. Moreover, since all operations on DP-FMI and $\tilde{e}_{\max}$ are post-processing of DP results, the private indexing incurs no additional privacy loss.

4 EXPERIMENTS

In this section, we benchmark our proposed DP-FMI against the SOTA DP index, SPECIAL (Wang et al., 2024). Since both indexes are lossless, our focus will be on comparing their performance overheads. Additional evaluations, including privacy tradeoffs of DP-FMI and accuracy comparisons with lossy DP indexes (Roy et al., 2020), are conducted but deferred to A.3 for space concerns.

4.1 EXPERIMENT SETUP

The baseline method. We select the SPECIAL index (Wang et al., 2024) as our comparison baseline because, to our knowledge, it is the only DP index that ensures (deterministic) lossless indexing. In contrast, the classical Cryptϵ (and its variant for growing data (Zhang et al., 2023)) is a lossy scheme and can suffer significant data loss, particularly with scaled data or high privacy levels (see our additional experiments in A.3). We also exclude ORAM-based index proposals like Oblix (Mishra et al., 2018) and GraphOS (Chamani et al., 2024), as they focus on fully oblivious settings with no leakage, while DP indexes consider to trade some bounded leakage for efficiency. As such, ORAM indexes typically do not offer performance comparable to DP indexes. For fair comparison reasons, we exclude both lossy DP indexes and fully oblivious ones from our main evaluation.

Datasets and Queries. Our experiments are conducted using 11 datasets, consisting of both synthetic and real-world data. The first eight datasets are generated from mathematical distributions, including uniform and lognormal, with each dataset scaled to four different sizes from 1K to 1M records. In addition, we use three real-world datasets: (i) The bank transaction records from the *Czech Financial Dataset* (Petrocelli, 2020), which consists of 1M entries; (ii) A sampled set of bureau credit records from Home Credit Data (Kaggle, 2018), for which we create two versions: one with 250K unique index keys, and another with 1M records allowing duplicate index keys. For the synthetic datasets, the generated numbers serve as the index column. In the bank transaction dataset, the index column is `account_id`, and in the credit card dataset, it is `SKU_ID`. These index columns are used to construct the CDF for training and evaluation. In our evaluations, we consider two types of index lookup: point query lookups and range query lookups. A point query retrieves the index of a single attribute value, while a range query looks up the indexes for a continuous set of attribute values. For example, a query like "get indexes for tuples where attribute $A_i = x_j$" is a point query lookup, whereas "get indexes for tuples where attribute $A_i \in [x_j, x_k]$" is a range query lookup.

Implementation and privacy settings. The FMI is implemented and trained using PyTorch (v2.4.0) with CUDA 12.6. The private DP-SGD training is implemented using the Privacy Engine from Opacus package (Yousefpour et al., 2021). For all experiments, including our baselines, we set a maximum privacy budget of $\epsilon = 1$. For DP-FMI, we allocate 0.8 of the budget to DP-SGD and the remainder to the report noisy max error mechanism. The default privacy multiplier in DP-SGD is set to 4.0 (adjusted to 3.0 for the transaction dataset due to better model outcomes), and we use Opacus's Privacy Engine with the default moments accountant method to track cumulative privacy loss, stopping training when it reaches 0.8. For the report noisy max error method, we set a conservative sensitivity bound of $10^{-3}|D|$, where $|D|$ is the size of the dataset. All implementations and benchmark data are *anonymously* open-sourced at https://github.com/uu60/learned-index-fmi.

System Configuration. The experiments were performed on a machine running Ubuntu 22.04.5 LTS (Jammy Jellyfish). The system is equipped with an Intel(R) Xeon(R) w3-2423 processor (6 cores, 12 threads), 64 GB of RAM, and an NVIDIA RTX A6000 GPU.

4.2 RESULTS

In the benchmark, we train a DP-FMI and create a SPECIAL index for each dataset. We then generate 50 random point and range queries to be processed by both indexing structures. The overhead for each method is measured, which refers to the additional data fetched beyond the true indexed data during lookup operations. Formally, let D represent a sorted dataset, and $V = D[v_0, v_1]$ denote the true indexing range, while $\tilde{V} = D[\tilde{v}_0, \tilde{v}_1]$ represents the range produced by DP-FMI. The overhead is then computed as $|\tilde{V}| - |V|$. We also calculate the relative overhead as $(|\tilde{V}| - |V|)/|V|$. Since the lookup cost is minimal compared to the cost of fetching extra data, overhead serves as the primary performance metric. We report and compare the average and maximum overheads of DP-FMI and SPECIAL indexes in Figure 5.

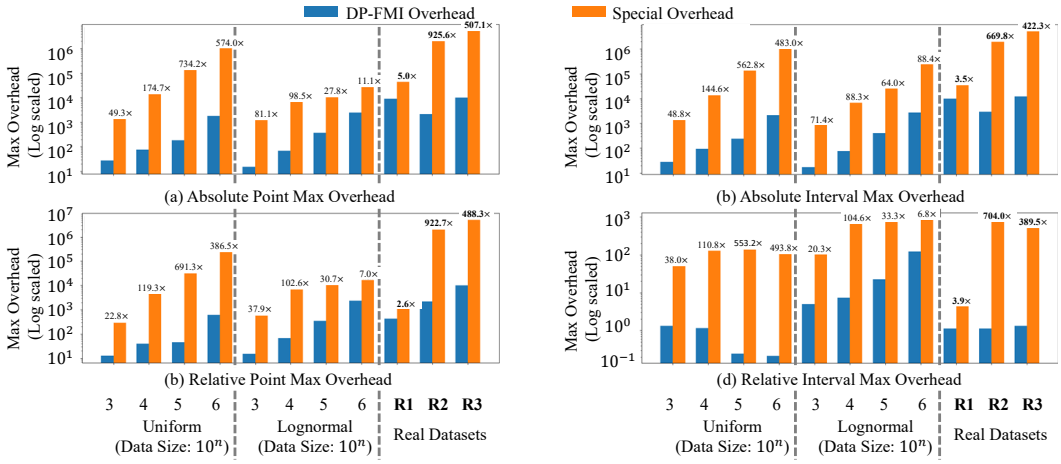


Figure 5: Approach evaluation based on absolute & relative maximum (Max) overhead includes 2 synthetic datasets with varying distributions and data sizes, and 3 real sampled datasets: **R1** with 1M records from the Czech Financial dataset, **R2** with 250K records from the bureau credit dataset, and **R3** with 1M records from the bureau credit dataset.

As shown in Figure 5, the DP-FMI model consistently exhibits significantly lower overhead than SPECIAL indexes, especially in larger, uniformly distributed datasets. For instance, in real-world datasets, DP-FMI achieves up to $925.6\times$ improvement in point query lookups. Even in the worst case, such as the point query lookups on the transaction data, DP-FMI still provides a $2.6\times$ improvement over SPECIAL. The reduced improvement in the transaction data is due to its small number of unique keys—only 4,500. Since SPECIAL's noise scale, which is the primary source of its overhead, is proportional to $O(\sqrt{N}/\epsilon)$, where N is the number of unique keys, it performs better in cases with fewer unique keys. However, as the unique key size grows, such as in the credit data with 250K unique keys, DP-FMI demonstrates dramatic improvements, with a minimum of $400\times$. A similar trend is also observed in the 1M credit data group, which contains 605,429 unique keys, where DP-FMI achieves at least a $389.5\times$ improvement over SPECIAL.

In addition to the significant performance improvements of DP-FMI over SOTA DP indexes, we observe several interesting trends: First, for uniform datasets, DP-FMI shows a steady increase in performance improvement over SPECIAL as the data size grows. This is because the number of unique keys increases proportionally with the dataset size (from 100 to 99,995 as the size grows from 1K to 1M), leading to a substantial increase in SPECIAL's CDF error. In contrast, lognormal datasets experience slower growth in unique keys (from 75 to 1,808) when expanding data sizes, thus resulting in more modest performance gains for DP-FMI compared to SPECIAL. Moreover, in chart (d) for the uniform dataset, the relative overhead of range queries decreases as the dataset size grows. This is because FMI maintains a fixed maximum error for each key, keeping the predicted overhead ranges $\tilde{V}_o$ consistent. As the real range V increases linearly with dataset size, the relative overhead, computed as $|\tilde{V}_o|/|V|$, decreases accordingly.

5 Conclusion

In this work, we propose DP-FMI, the first DP learned index that is both lossless and efficient. Benchmark comparisons show that DP-FMI achieves up to $925.6\times$ improvement over the SOTA DP indexes. Despite this, further optimization is possible. For example, the sensitivity bound in the report noisy max error mechanism is set conservatively large, which can also inflate overhead. A tighter sensitivity estimate could improve DP-FMI's performance. Moreover, to achieve lossless, another approach could involve mimicking SPECIAL, learning two noisy indexes (one overestimating and the other underestimating the cumulative frequency) by adjusting the loss function to favor positive or negative errors. This may eliminate the need for max error based range overestimation and further reducing overhead. Finally, while this work focuses on static data, extending DP-FMI to support private, efficient, and lossless updates for dynamic data is an intriguing future direction.

REFERENCES

Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th annual symposium on foundations of computer science*, pp. 464–473. IEEE, 2014.

Javad Ghareh Chamani, Ioannis Demertzis, Dimitrios Papadopoulos, Charalampos Papamanthou, and Rasool Jalili. Graphos: Towards oblivious graph processing. *Cryptology ePrint Archive*, 2024.

Victor Costan and Srinivas Devadas. Intel sgx explained. *Cryptology ePrint Archive*, 2016.

Rudrajit Das, Satyen Kale, Zheng Xu, Tong Zhang, and Sujay Sanghavi. Beyond uniform lipschitz condition in differentially private optimization. In *International Conference on Machine Learning*, pp. 7066–7101. PMLR, 2023.

Zeyu Ding, Daniel Kifer, Thomas Steinke, Yuxin Wang, Yingtai Xiao, Danfeng Zhang, et al. The permute-and-flip mechanism is identical to report-noisy-max with exponential noise. *arXiv preprint arXiv:2105.07260*, 2021.

Cynthia Dwork and Guy N Rothblum. Concentrated differential privacy. *arXiv preprint arXiv:1603.01887*, 2016.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Saba Eskandarian and Matei Zaharia. Oblidb: Oblivious query processing for secure databases. *arXiv preprint arXiv:1710.00458*, 2017.

GDPR. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, 10(3152676):10–5555, 2017.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

HIPAA. Summary of the hipaa privacy rule. *Office for Civil Rights*, 2003.

Kaggle. Home credit default risk. `https://www.kaggle.com/competitions/home-credit-default-risk`, 2018. Accessed: 2024-09-30.

Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O'neill. Generic attacks on secure outsourced databases. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1329–1340, 2016.

Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, 63(7):93–101, 2020.

Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 international conference on management of data*, pp. 489–504, 2018.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 19–34, 2018a.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018b.

Jihao Liu, Xin Huang, Guanglu Song, Hongsheng Li, and Yu Liu. Uninet: Unified architecture search with convolution, transformer, and mlp. In *European Conference on Computer Vision*, pp. 33–49. Springer, 2022.

Pratyush Mishra, Rishabh Poddar, Jerry Chen, Alessandro Chiesa, and Raluca Ada Popa. Oblix: An efficient oblivious search index. In *2018 IEEE symposium on security and privacy (SP)*, pp. 279–296. IEEE, 2018.

Andrew Paverd, Andrew Martin, and Ian Brown. Modelling and automatically analysing privacy properties for honest-but-curious adversaries. *Tech. Rep*, 2014.

L. Petrocelli. Czech financial dataset: Real anonymized transactions. https://data.world/lpetrocelli/czech-financial-dataset-real-anonymized-transactions, 2020. Accessed: 2024-09-30.

Lina Qiu, Georgios Kellaris, Nikos Mamoulis, Kobbi Nissim, and George Kollios. Doquet: Differentially oblivious range and join queries with private data structures. *Proceedings of the VLDB Endowment*, 16(13):4160–4173, 2023.

Amrita Roy, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. Crypt: Crypto-assisted differential privacy on untrusted servers. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pp. 603–619, 2020.

IBM Security. Cost of a data breach report 2024. https://www.ibm.com/reports/data-breach, 2024. Accessed: 2024-09-30.

The White House. National strategy to advance privacy-preserving data sharing and analytics, 2022. URL https://www.whitehouse.gov/wp-content/uploads/2022/03/National-Strategy-to-Advance-Privacy-Preserving-Data-Sharing-and-Analytics-1.pdf.

Chenghong Wang, Johes Bater, Kartik Nayak, and Ashwin Machanavajjhala. Dp-sync: Hiding update patterns in secure outsourced databases with differential privacy. In *Proceedings of the 2021 International Conference on Management of Data*, pp. 1892–1905, 2021.

Chenghong Wang, Lina Qiu, Johes Bater, and Yukui Luo. Special: Synopsis assisted secure collaborative analytics. *arXiv preprint arXiv:2404.18388*, 2024.

Yang Wu, Xuanhe Zhou, Yong Zhang, and Guoliang Li. Automatic database index tuning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Amr Yousefpour, Amirhossein Shamsabadi, Adrià Gascón, Alex Kurakin, Abhradeep Thakurta, James Jordon, and Alperen Yalçın. Opacus: User-friendly differential privacy library in pytorch. https://github.com/pytorch/opacus/blob/main/opacus/privacy_engine.py, 2021. Accessed: 2024-09-30.

Yanping Zhang, Johes Bater, Kartik Nayak, and Ashwin Machanavajjhala. Longshot: Indexing growing databases using mpc and differential privacy. *Proceedings of the VLDB Endowment*, 16 (8):2005–2018, 2023.

Wenting Zheng, Ankur Dave, Jethro G Beekman, Raluca Ada Popa, Joseph E Gonzalez, and Ion Stoica. Opaque: An oblivious and encrypted distributed analytics platform. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pp. 283–298, 2017.

A Appendix

A.1 Proof of FMI training satisfies DP

We say that the private training on our FMI is inherently the directly adoption of DP-SGD, so that it does not affect the privacy guarantees from those provided by DP-SGD. For completeness, we provide a formal privacy analysis to here, which is in herently derived from the proof technique of DP-SGD Goodfellow et al. (2016). Specifically, we will consider two neighboring datasets D and D', and we abstract the training process as a probabilistic mechanism $\mathcal{M}$. We will set the constraint

$\sigma \geq 2\epsilon^{-1}C\sqrt{2\ln(1.25/\delta)}$, and then compute the ratio of the probabilities for the output of the mechanism $\mathcal{M}(D)$ and $\mathcal{M}(D')$ as:

$$\ln\left(\frac{\Pr[\mathcal{M}(D)=\Theta]}{\Pr[\mathcal{M}(D\prime)=\Theta]}\right) = \ln\left(\frac{\frac{1}{(2\pi\sigma^2C^2)^{d/2}}\exp\left(-\frac{\|\Theta-\tilde{g}(D)\|_2^2}{2\sigma^2C^2}\right)}{\frac{1}{(2\pi\sigma^2C^2)^{d/2}}\exp\left(-\frac{\|\Theta-\tilde{g}(D\prime)\|_2^2}{2\sigma^2C^2}\right)}\right)$$
$$= \ln\left(\exp\left(\frac{\|\Theta-\tilde{g}(D\prime)\|_2^2-\|\Theta-\tilde{g}(D)\|_2^2}{2\sigma^2C^2}\right)\right)$$
$$= \frac{\|\Theta-\tilde{g}(D\prime)\|_2^2-\|\Theta-\tilde{g}(D)\|_2^2}{2\sigma^2C^2}$$
$$= \frac{\|\Theta-\tilde{g}(D)+\tilde{g}(D)-\tilde{g}(D\prime)\|_2^2-\|\Theta-\tilde{g}(D)\|_2^2}{2\sigma^2C^2}$$
$$= \frac{\|\Theta-\tilde{g}(D)\|_2^2+2(\tilde{g}(D)-\tilde{g}(D\prime))^\top(\Theta-\tilde{g}(D))+\|\tilde{g}(D)-\tilde{g}(D\prime)\|_2^2-\|\Theta-\tilde{g}(D)\|_2^2}{2\sigma^2C^2}$$
$$= \frac{2(\tilde{g}(D)-\tilde{g}(D\prime))^\top(\Theta-\tilde{g}(D))+\|\tilde{g}(D)-\tilde{g}(D\prime)\|_2^2}{2\sigma^2C^2} = *$$

Here, we apply Cauchy-Schwarz inequality, and which then gives us

$$* \leq \frac{2\|\tilde{g}(D)-\tilde{g}(D\prime)\|_2\cdot\|\Theta-\tilde{g}(D)\|_2+\|\tilde{g}(D)-\tilde{g}(D\prime)\|_2^2}{2\sigma^2C^2} = **$$

Since the gradients are clipped to have max ℓ_2 norm of C, so that $\|\tilde{g}(D)-\tilde{g}(D\prime)\|_2 \leq 2C$, and thus we can derive that the following

$$** \leq \frac{2C\cdot\|\Theta-\tilde{g}(D)\|_2+4C^2}{2\sigma^2C^2} = \frac{C\cdot\|\Theta-\tilde{g}(D)\|_2}{\sigma^2C^2}+\frac{2C^2}{\sigma^2C^2} = \frac{\|\Theta-\tilde{g}(D)\|_2}{\sigma^2C}+\frac{2}{\sigma^2}$$

We substitute $\sigma = \frac{2C\sqrt{2\ln(1.25/\delta)}}{\varepsilon}$, to the above equation, then we obtain

$$\frac{\|\Theta-\tilde{g}(D)\|_2}{\sigma^2C}+\frac{2}{\sigma^2} \leq \varepsilon$$

A.2 Proof of the Report Noisy Max Error is DP

We say that to prove that Algorithm 1 satisfies DP is equivalent to demonstrate the following.

Given two neighboring datasets x and x', which differ by at most one record, and a score function $q(y;x)$ for each possible output $y \in \mathcal{Y} = \{1,2,\ldots,d\}$, the score function $q(y;x)$ has sensitivity $|q(y;x)-q(y;x')| \leq \Delta$ for all y. We then add to each score $q(y;x)$ a random variable drawn from an exponential distribution $Z_y \sim \text{Exp}\left(\frac{\epsilon}{2\Delta}\right)$.

Let the mechanism $\mathsf{RNM}(x) = \arg\max_y\left(q(y;x)+Z_y\right)$. We prove that for any y, and any neighboring datasets x and x', the following holds:

$$\Pr[\mathsf{RNM}(x)=y] \leq e^\epsilon\cdot\Pr[\mathsf{RNM}(x')=y]+\delta.$$

So in fact if we set $q(y;x)$ to be $|\hat{y}(x)-y|$ then the aforementioned scenario is equivalent to Algorithm 1. Hence, in what follows, we focus on proving the above scenario.

1. RNM **follows exponential distribution.** To prove DP, we will first need to compute the probability distribution of RNM. The probability that y is selected is:

$$\Pr[\mathsf{RNM}(x)=y] = \Pr\left(q(y;x)+Z_y > q(u;x)+Z_u, \quad \forall u \neq y\right).$$

Because the exponential distribution is memoryless and the noise variables Z_y are i.i.d., so we can then describe the probability of selecting y (under x and x'), using the exponential distribution

$$\Pr[\mathsf{RNM}(x)=y] = \frac{\exp\left(\frac{\epsilon\cdot q(y;x)}{2\Delta}\right)}{\sum_{u\in\mathcal{Y}}\exp\left(\frac{\epsilon\cdot q(u;x)}{2\Delta}\right)}; \quad \Pr[\mathsf{RNM}(x')=y] = \frac{\exp\left(\frac{\epsilon\cdot q(y;x')}{2\Delta}\right)}{\sum_{u\in\mathcal{Y}}\exp\left(\frac{\epsilon\cdot q(u;x')}{2\Delta}\right)}.$$

2. Probability ratio between x and x' is bounded by e^{ϵ}. In what follows, without loss of generity, we will assume $x' \leq x$, and we only prove one direction of the probability ratio, while by symmetric, the other side can be trivially implied. Now, we compute:

$$\frac{\Pr[\mathsf{RNM}(x) = y]}{\Pr[\mathsf{RNM}(x') = y]} = \frac{\exp\left(\frac{\epsilon \cdot q(y;x)}{2\Delta}\right)}{\exp\left(\frac{\epsilon \cdot q(y;x')}{2\Delta}\right)} \cdot \frac{\sum_u \exp\left(\frac{\epsilon \cdot q(u;x')}{2\Delta}\right)}{\sum_u \exp\left(\frac{\epsilon \cdot q(u;x)}{2\Delta}\right)}.$$

Note that $\forall y |q(y;x) - q(y;x')| \leq \Delta. \Rightarrow q(y;x) \leq q(y;x') + \Delta, \quad q(y;x') \leq q(y;x) + \Delta.$, thus:

$$\exp\left(\frac{\epsilon \cdot q(y;x)}{2\Delta}\right) \leq \exp\left(\frac{\epsilon \cdot (q(y;x') + \Delta)}{2\Delta}\right) = e^{\frac{\epsilon}{2}} \cdot \exp\left(\frac{\epsilon \cdot q(y;x')}{2\Delta}\right).$$

For the denominator, since the exponential function is monotonic (and $x > x'$), so that:

$$\sum_u \exp\left(\frac{\epsilon \cdot q(u;x)}{2\Delta}\right) \geq \exp\left(\frac{\epsilon \cdot q(y;x)}{2\Delta}\right)$$
$$\Rightarrow \sum_u \exp\left(\frac{\epsilon \cdot q(u;x')}{2\Delta}\right) \leq e^{\frac{\epsilon}{2}} \cdot \exp\left(\frac{\epsilon \cdot q(u;x)}{2\Delta}\right).$$

By applying the above bounds, we re-compute the probability ratio as

$$\frac{\Pr[\mathsf{RNM}(x) = y]}{\Pr[\mathsf{RNM}(x') = y]} \leq e^{\frac{\epsilon}{2}} \cdot e^{\frac{\epsilon}{2}} = e^{\epsilon}.$$

Additional discussions. We conduct additional discussions on the case where we set a smaller the sensitivity bound Δ in the purist of better performance (e.g., smaller scale of noise to be added to $e_{\max}$). However we may now assume Δ might be violated, for instance with small probability $\delta > 0$. In this case, the probability ratio bound might no longer hold. However, by definition, this failure happens with probability at most δ. This then translate into the (ϵ, δ)-DP guarantee.

A.3 Additional Experiments

Privacy tradeoff experiment. In this experiment, the trend of the maximum error of the FMI model during each training epoch, along with the corresponding epsilon values, is presented in Figure 6. We observe that as epsilon increases, the FMI model shows a reduction in maximum error, reflecting improved performance. This relationship allows us to select the most suitable model for specific privacy requirements.

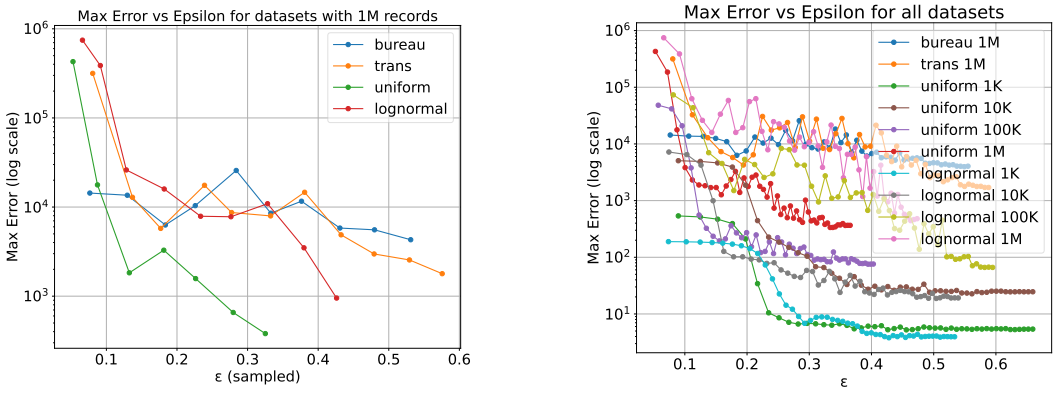


Figure 6: Max Error vs Epsilon: The left chart shows the maximum error as epsilon increases for four datasets, each containing 1 million records. The right chart includes all datasets of various record sizes.

Accuracy comparison results. In this experiment, we compare the rate of missing data (i.e., the proportion of data that is not successfully retrieved) for three indexing methods: Cryptϵ, SPECIAL,

and FMI. The results, as shown in Table 1, highlight significant differences in the performance of these methods across various datasets for both point queries and range queries. Formally, let D represent a sorted dataset, and $V = D[v_0, v_1]$ denote the true indexing range, while $\tilde{V} = D[\tilde{v_0}, \tilde{v_1}]$ represents the range produced by indexing mechanism. The missing data rate is computed as $\frac{\max(0,\min(v_1,\tilde{v_1})-\max(v_0,\tilde{v_0}))}{|V|}$.

Both SPECIAL and FMI maintain a 0.0 missing data rate across all datasets and query types, meaning that they never fail to retrieve the correct data, ensuring complete accuracy for both point and range queries. However, Cryptϵ performs substantially worse, with consistently high missing data rates, particularly in larger datasets, rendering it unsuitable for practical use. For instance, in the uniform 1K dataset, Cryptϵ has a missing data rate of 0.4899 for point queries and 0.1618 for range queries, meaning that nearly half of the data for point queries and over 16% of the data for range queries are not retrieved correctly.

As the dataset size increases, Cryptϵ's performance even gets worse, remaining significantly inferior to SPECIAL and FMI. Expect uniform distribution dataset with less than 1M records, for example, Cryptϵ's missing data rate for point queries increases to nearly 100%. This level of performance is unacceptable in real-world applications, especially when SPECIAL and FMI continue to show a 0.0 missing data rate, regardless of the dataset size.

In summary, while SPECIAL and FMI maintain flawless performance with 0.0 missing data across all datasets and query types, Cryptϵ's performance is consistently poor, particularly in smaller datasets, lognormal distributions and real-world datasets. Its high rate of missing data close to over 90% makes it impractical for real-world use. These results underscore the superiority of SPECIAL and FMI in providing reliable, accurate indexing, while Cryptϵ proves inadequate for tasks requiring high precision.

Table 1: Accuracy comparison between Cryptϵ, SPECIAL, and FMI

Model	**Dataset**	**Point (Avg)**	**Point (Max)**	**Range (Avg)**	**Range (Max)**
Cryptϵ	`uniform 1K`	0.4899	1.0	0.1618	0.3333
	`uniform 10K`	0.6365	1.0	0.0688	0.2034
	`uniform 100K`	0.9970	1.0	0.1528	0.3353
	`uniform 1M`	1.0	1.0	1.0	1.0
	`lognormal 1K`	0.9406	1.0	0.8542	1.0
	`lognormal 10K`	1.0	1.0	1.0	1.0
	`lognormal 100K`	1.0	1.0	0.9986	1.0
	`lognormal 1M`	1.0	1.0	0.9940	1.0
	`trans 1M`	1.0	1.0	0.9904	1.0
	`bureau 250k`	1.0	1.0	0.9423	1.0
	`bureau 1M`	1.0	1.0	0.9264	1.0
SPECIAL	All datasets	0.0	0.0	0.0	0.0
FMI	All datasets	0.0	0.0	0.0	0.0