A Differentially Private Index for Range Query Processing in Clouds

Cetin Sahin, Tristan Allard, Reza Akbarinia, Amr El Abbadi, Esther Pacitti

A Differentially Private Index for Range Query Processing in Clouds

Cetin Sahin[1], Tristan Allard[2,3,4], Reza Akbarinia[5,6,7], Amr El Abbadi[1], Esther Pacitti[4,5,7]

[1] *UC Santa Barbara,* [2] *Univ. Rennes,* [3] *Irisa,* [4] *CNRS,* [5] *INRIA,* [6] *LIRMM,* [7] *Univ. Montpellier*

{cetin, amr}@ucsb.edu, tristan.allard@irisa.fr,
reza.akbarinia@inria.fr, esther.pacitti@lirmm.fr

Abstract—**Performing non-aggregate range queries on cloud stored data, while achieving both privacy and efficiency is a challenging problem. This paper proposes constructing a differentially private index to an outsourced encrypted dataset. Efficiency is enabled by using a cleartext index structure to perform range queries. Security relies on both differential privacy (of the index) and semantic security (of the encrypted dataset). Our solution, PINED-RQ develops algorithms for building and updating the differentially private index. Compared to state-of-the-art secure index based range query processing approaches, PINED-RQ executes queries in the order of at least one magnitude faster. The security of PINED-RQ is proved and its efficiency is assessed by an extensive experimental validation.**

I. Introduction

Substantial advances in cloud technologies have made outsourcing data to the cloud highly beneficial today. However, strong concerns from private companies and public institutions about the security of the outsourced data still hamper the adoption of cloud solutions. This reluctance is fed by frequent massive data breaches either caused by external attacks against cloud service providers or by negligent or opaque practices from the service provider.

During the last decade, a large body of academic work has tackled the problem of outsourcing databases to an untrusted cloud while maintaining both confidentiality and SQL-like querying functionality (at least partially). However, to the best of our knowledge, *performing range queries efficiently* in this context has not been addressed in a satisfactory manner. Range queries express a bounded restriction over the retrieved records. They are fundamental database operations. For example, assume that a university has outsourced its student database to the cloud. The following SQL query can retrieve the records of students with a grade between A and B: `SELECT * FROM` students `WHERE` grade $\geq$ 3.0 `AND` grade $\leq$ 4.0. Most related work has essentially focused on trading efficiency with security. In particular, they either allow unacceptable security leakage or employ costly cryptographic computations. For example, bucketization techniques [12] do not provide formal privacy guarantees and order preserving encryption based schemes (OPE) [2] are vulnerable to statistical attacks, while searchable symmetric encryption schemes [14], [5] suffer from execution times that are incompatible with real-world efficiency requirements.

In this paper, we advocate a novel efficient approach without sacrificing sound formal privacy guarantees. We build on the most recent advances in the field of privacy-preserving data publishing by *using cryptography with differential privacy for performing selection range queries*. Our solution proposes to send two complementary data structures to the cloud: an encrypted version of the database, *e.g.,* AES encryption scheme, indexed by a hierarchy of histograms, such that both are perturbed to satisfy differential privacy. Efficiency comes from the disclosure of the index, in the clear, to the cloud, for guiding the query execution strategy. No computation is ever performed on encrypted data. Privacy comes from the differential privacy guarantees of the function that computes the encrypted database and the index. Indeed, the differential privacy model is today's *de facto* standard for protecting personal information that needs to be partially disclosed. It applies to the functions computed on personal data, and defines privacy as a limit on the impact that any possible record may have on their outputs. This new efficiency/privacy trade-off, however, comes at a cost: the differentially private perturbation makes the index inaccurate. There may be some records that are relevant to the query that will not be retrieved (false negatives), and there may be some irrelevant records that will actually be returned (false positives). After receiving the result set, clients perform post-processing to filter out false positive records. This lower precision is the inevitable cost of increased privacy. A specific query processing strategy must be designed to cope with such inaccuracies, as well as an update management system to maintain the data structures when the original database is updated (inserts, deletes, or modifications of records) without jeopardizing the privacy guarantees.

We propose PINED-RQ (**P**rivacy-preserving **IN**dex for **E**ncrypted **D**atabases dedicated to **R**ange **Q**ueries) which makes the following contributions:

1) A differentially private function computing the encrypted dataset and its hierarchical index.
2) An update strategy for managing the insertion, deletion, and modification of records on the cloud.
3) Formal proofs showing the security of PINED-RQ.
4) A thorough empirical evaluation demonstrating the efficiency and quality of the query processing strategy.

To the best of our knowledge, this is the first work that builds, uses, and maintains a differentially private index for performing selection range queries.

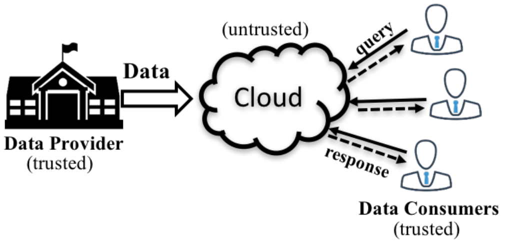


Fig. 1: System Model

II. Problem Definition

A. Basic Components

Trusted Component. Raw data is produced by the *data provider* and queried by *data consumers*. The data provider encrypts the data and creates a differentially private index, which are sent to the cloud. Both the data provider and data consumers are trusted (see Figure 1), they are considered to be *honest*. For example, in a *university* setting, the data provider would be the university information system and the data consumers are the teachers and administrators.

Untrusted Component. The cloud is untrusted. It stores the data outsourced by the data provider and processes the queries posed by data consumers. We assume the cloud is *honest-but-curious*: it records all information resulting from its exchanges with the trusted components and may infer anything that can be inferred in a computationally-feasible way.

B. Basic Data Structures

The dataset stored by the data provider is a relation $\mathcal{D}(A_1, \ldots, A_d)$ where each A_i is an attribute. Queries are non-aggregate single-dimensional *range queries*, over a single attribute A_q. A query $\mathcal{Q}$ consists of a set of disjunctions of non-overlapping ranges over A_q : $\mathcal{Q} \leftarrow \phi_1 \vee \ldots \vee \phi_l$ where each ϕ_i is a range defined by a minimum and a maximum value, $\phi_i.min$ and $\phi_i.max$, such that $\cap_{\forall i} \phi_i = \emptyset$. Without loss of generality, we assume a query $\mathcal{Q}$ consists of a single range. Data Consumers submit queries *in the clear* to the cloud, without leading to any security breach. The attributes of $\mathcal{D}$ can be of any type, except A_q which must be a totally ordered data type to allow range queries. The set of records in $\mathcal{D}$ that satisfy $\mathcal{Q}$ exactly is called the set of *relevant records* and is denoted $\mathcal{R}$.

In order for the cloud to process queries, the data provider provides the following two data structures to the cloud:

- An encrypted version of the dataset denoted $\overline{\mathcal{D}}$. The encryption $\overline{r}$ of a record $r \in \mathcal{D}$ is performed by concatenating the attribute bit values of r and encrypting the resulting bitstring by a *semantically-secure* encryption scheme [11], which means that no probabilistic polynomial-time algorithm is able to gain additional knowledge on a record given its encrypted version and (possibly) auxiliary information (*e.g.,* AES in CBC mode). Loosely speaking, semantic security implies that no information leaks about a cleartext bitstring, given its encrypted value.
- An *index*, denoted $\mathcal{I}(A_q)$, over the queriable attribute of $\mathcal{D}$, A_q, computed from $\mathcal{D}$ but pointing to the encrypted records $\overline{r} \in \overline{\mathcal{D}}$. The index is sent in the clear to the cloud. The information in the nodes of the index is randomly perturbed so that *differential privacy* is satisfied.

The differentially private perturbation of the index results in an inherent approximation in the set of records that is returned: false positives may be returned while false negatives may be omitted. The recall and precision of an approximate set of records returned by the cloud are defined as follows.

Definition 1 (Recall and Precision): Given a query $\mathcal{Q}$, with an exact set of relevant records $\mathcal{R}$ in $\mathcal{D}$, while the set of records returned by the cloud is $\widetilde{\mathcal{R}}$, then the recall $\mathtt{r}$ and precision $\mathtt{p}$ of $\widetilde{\mathcal{R}}$ are: $\mathtt{r} = |\mathcal{R} \cap \widetilde{\mathcal{R}}|/|\mathcal{R}|$ and $\mathtt{p} = |\mathcal{R} \cap \widetilde{\mathcal{R}}|/|\widetilde{\mathcal{R}}|$.

C. End-to-End Privacy Model

PINED-RQ combines a differentially-private perturbation scheme with a semantically-secure encryption scheme. Differential privacy and semantic security are two well-established models but each comes with is own formalism and assumptions. We first describe these two models in their usual settings, and then show how PINED-RQ integrates them consistently to come up with an overall unified end-to-end privacy model.

1) Traditional Differential Privacy: The ϵ-differential privacy model [6] requires that whatever the output of an ϵ-differentially private function, the probability that any given individual record $r \in \mathcal{D}(A_1, \ldots, A_n)$ is present in the dataset is close to the probability that r is absent by an e^ϵ factor. This model considers a very strong adversary that is not computationally-bounded (information-theoretic guarantees). Definition 2 gives a formal definition.

Definition 2 (ϵ-differential privacy [6], [8]): Randomized function $\mathtt{f}$ satisfies ϵ-differential privacy if:

$$\mathtt{Pr}[\mathtt{f}(\mathcal{D}_1) = \mathcal{O}] \leq e^\epsilon \cdot \mathtt{Pr}[\mathtt{f}(\mathcal{D}_2) = \mathcal{O}]$$

for any set $\mathcal{O} \in Range(\mathtt{f})$ and any dataset $\mathcal{D}_1$ and $\mathcal{D}_2$ that differ in at most one record.

PINED-RQ perturbs the index $\mathcal{I}(A_q)$ based on the Laplace mechanism (Definition 3) and on the composability properties of the ϵ-differential privacy model.

Definition 3 (Laplace mechanism [7]): Let D_1 and D_2 be two datasets such that D_2 is obtained from D_1 by changing the value of one record. Let $\mathtt{f}$ be a real-valued function. Let $\mathcal{L}(\lambda)$ denote a random variable which has a Laplace distribution with probability density function $pdf(x, \lambda) = \frac{1}{2\lambda} \cdot e^{-|x|/\lambda}$. The Laplace mechanism consists of adding $\mathcal{L}(\max \|\mathtt{f}(D_1) - \mathtt{f}(D_2)\|_1/\epsilon)$ to the output of $\mathtt{f}$, where $\epsilon > 0$.

Theorem 1 (Compositions [15]): Let $(\mathtt{f}_1, \ldots, \mathtt{f}_n)$ be a sequence of real-valued functions each satisfying ϵ_i-differential privacy. This sequence of functions satisfies (1) $(\sum_{i=1}^n \epsilon_i)$-differential privacy when applied to the same dataset (*sequential*), and (2) $(\max(\epsilon_i))$-differential privacy when applied to disjoint datasets (*parallel*).

2) Traditional Semantic Security: Efficient private key encryption schemes today satisfy *semantic security* (*e.g.,* AES in CBC mode). Loosely speaking, semantic security considers a computationally-bounded adversary, *i.e.,* a probabilistic polynomial time algorithm, and requires that anything computable from the encrypted message and any auxiliary information, *e.g.,* obtained from another external data source, can also be computed efficiently from the auxiliary information only. Definition 4 is a simplification of Definition 5.2.1 in [10][1].

Definition 4 (Semantic security [10]): A private key encryption algorithm $\mathtt{E}_\chi$, where χ is the secret key, is *semantically secure* if for every probabilistic polynomial time algorithm $\mathtt{A}$ there exists a probabilistic polynomial time algorithm $\mathtt{A}'$ such that for every input dataset $\mathcal{D}$, every auxiliary background knowledge $\zeta \in \{0,1\}^*$, every polynomially bounded function $\mathtt{g} : \{0,1\}^* \rightarrow \{0,1\}^*$, every polynomial $p(\cdot)$, every sufficiently large $n \in N$, it holds that :

$$\mathtt{Pr}[A_n(\mathtt{E}_\chi(\mathcal{D}), |\mathcal{D}|, \zeta) = \mathtt{g}(\mathcal{D})] < \mathtt{Pr}[A'_n(|\mathcal{D}|, \zeta) = \mathtt{g}(\mathcal{D})] + \frac{1}{p(n)}$$

3) Unified Privacy Model for PINED-RQ: The unified privacy model of PINED-RQ is a probabilistic relaxation of a variant of differential privacy that considers computationally-bounded adversaries [16] and that consequently takes into account the cryptographically-negligible leaks due to the use of efficient real-world encryption schemes. Definition 5 is a simplification of ϵ_n-SIM-CDP, the simulation-based computational differential privacy model proposed in [16].

Definition 5 (ϵ_n-SIM-CDP privacy[16]): Randomized function $\mathtt{f}_n$ provides ϵ_n-SIM-CDP if there exists a function $\mathtt{F}_n$ that satisfies ϵ_n-differential-privacy and a polynomial $p(\cdot)$, such that for every input dataset $\mathcal{D}$, every probabilistic polynomial time adversary $\mathtt{A}$, every auxiliary background knowledge $\zeta \in \{0,1\}^*$, and every sufficiently large $n \in N$, it holds that :

$$|\mathtt{Pr}[\mathtt{A}_n(\mathtt{f}_n(\mathcal{D}, \zeta)) = 1] - \mathtt{Pr}[\mathtt{A}_n(\mathtt{F}_n(\mathcal{D}, \zeta)) = 1]| \leq \frac{1}{p(n)}$$

Finally, the overall end-to-end privacy model of PINED-RQ is stated in Definition 6.

Definition 6 ($(\epsilon, \delta)_n$-Probabilistic-SIM-CDP): A randomized function $\mathtt{f}_n$ is said to provide $(\epsilon, \delta)_n$-Probabilistic-SIM-CDP, if it provides ϵ_n-SIM-CDP to each individual with probability greater than or equal to δ, where $\delta \in]0, 1]$.

D. Problem in a Nutshell

We address the problem of designing the functions (1) `CREATE`, in charge of computing the two complementary data structures $\mathcal{I}(A_q)$ and $\overline{\mathcal{D}}$, (2) `INSERT` and `MOD/DEL`, in charge of handling the updates over $\mathcal{D}$, and (3) the query processing strategies of the cloud that answer cleartext range queries $\mathcal{Q}$ while ensuring high recall and precision, (4) all that while satisfying $(\epsilon, \delta)_n$-Probabilistic-SIM-CDP.

III. Static Context: the Create Function

In this section, we discuss the basic design of the PINED-RQ `CREATE` function and the corresponding query processing strategy in a read-only context. Throughout the section, we refer to the sample example depicted in Figure 2. The aim is to publish a student GPA dataset to a cloud server privately. The original dataset has 7 student records with GPAs from 0 to 4. Initially, a clear index is constructed for the dataset where leaf nodes point to the tuples with the corresponding GPA value. Then, the index entries are perturbed with differentially private noise which results in adding dummy records or deleting some records. In the example, 1 dummy record is inserted to the node with range $[2, 3)$ and 1 dummy record is removed from the node with range $[0, 1)$. After the perturbation, the records are encrypted with a semantically secure encryption scheme. Finally, the cloud is provided the output of the CREATE function : namely, the encrypted dataset $\overline{\mathcal{D}}$ and the cleartext differentially private index $\mathcal{I}(A_q)$ over $\overline{\mathcal{D}}$. In this section, we first describe the CREATE function, then prove that it satisfies the $(\epsilon, \delta)_n$-Probabilistic-SIM-CDP privacy model, and finally present a simple query execution strategy.

A. Data Structures Output by CREATE

We describe the two data structures provided to the cloud, and demonstrate the end-to-end security of the proposed system.

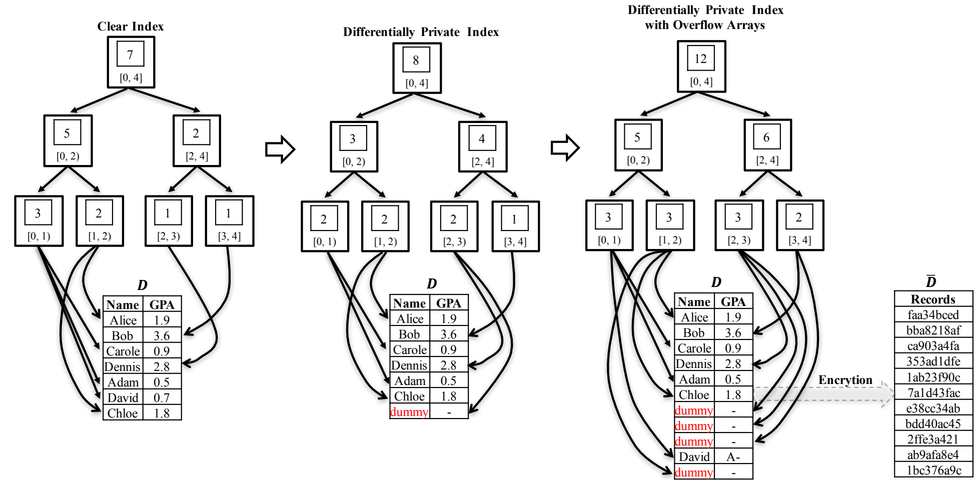
The first data structure computed by the data provider is the differentially private index. The design of this index must address the conflicting goals of allowing the retrieval of the records in a given range query while satisfying differential privacy. Both false negatives and positives are inherent, thus the challenge lies in reducing them to realistic numbers. To this end, we propose to benefit from two fruitful research tracks: on the one hand, from the B+-Tree family of indices, widely used in traditional databases for supporting range queries, and on the other hand, from the more recent differentially private hierarchies of histograms, that have been shown to answer *aggregate* range queries accurately while satisfying differential privacy [18]. Index $\mathcal{I}(A_q)$ is a balanced tree over the encrypted dataset $\overline{\mathcal{D}}$ in which each *intermediate node* contains a fixed-size set of pointers and each *leaf node* contains a set of pointers to the data records in $\overline{\mathcal{D}}$.

1) Basics : Nodes and Histograms: In a B+-Tree, a key is a single value, *e.g.,* an integer, that indicates the range within which fall the records that can be accessed while following the associated pointers. In our context, rather than using single values as indications for ranges, we propose to use *histograms*. Indeed, histograms can be used as accurate estimators for ranges through the distributions they disclose. Histograms are defined in Definition 7. Histograms are also well known for integrating well with differential privacy [18]: the Laplace mechanism and the parallel composition theorem together allow adding a random variable sampled independently in $\mathcal{L}(1/\epsilon)$ to each bin in order to satisfy ϵ-differential privacy.

Definition 7 (Histogram and Histogram Bins): Let $\Phi \leftarrow (\phi_1, \ldots, \phi_k)$ be a set of non-overlapping ranges that partition the domain of queriable attribute A_q. Each range $\phi_i \in \Phi$ is associated to its corresponding *bin* b_i where b_i stores the number of data records within the range of ϕ_i. Each bin belongs to a unique node (as in Definition 8) in the index. A histogram h is a complete set of bins over the entire domain.

Definition 8 (Node): A node is a histogram bin/pointer pair, where the pointer references either a child node or encrypted data records. Each node represents a bin of a histogram.

[1] For simplicity, we use the single-message setting formalization but emphasize that it is extended easily to the multiple-messages setting (see Definition 5.2.8 in [10]).



D

Name	GPA
Alice	1.9
Bob	3.6
Carole	0.9
Dennis	2.8
Adam	0.5
David	0.7
Chloe	1.8

D

Name	GPA
Alice	1.9
Bob	3.6
Carole	0.9
Dennis	2.8
Adam	0.5
Chloe	1.8
dummy	-

D

Name	GPA
Alice	1.9
Bob	3.6
Carole	0.9
Dennis	2.8
Adam	0.5
Chloe	1.8
dummy	-
dummy	-
dummy	-
David	A-
dummy	-

$\overline{D}$

Records
faa34bced
bba8218af
ca903a4fa
353ad1dfe
1ab23f90c
7a1d43fac
e38cc34ab
bdd40ac45
2ffe3a421
ab9afa8e4
1bc376a9c

Fig. 2: Sample Publication

Considering the example from Figure 2, the index structure has 7 nodes where 4 of them are leaf nodes. The leaf nodes have histogram bins $\{[0,1),\ [1,2),\ [2,3),\ [3,4]\}$. 4 leaf nodes together construct a complete histogram at the leaf level.

2) Building $\mathcal{I}\left(A_q\right)$: The hierarchy of nodes in the index structure is built following a three-step process. The first step computes the clear index, *i.e.,* the nodes and the pointers. It does require a single pass over the clear dataset to associate each data record with the corresponding leaf node. Later, the hierarchy of the index is constructed considering the non-private *branching factor*, a system parameter of the structure. The second step perturbs the nodes of the index based on the Laplace mechanism to satisfy differential privacy and post-processes them in order to increase its utility. The third step encrypts the data records with a semantically secure encryption scheme after constructing the differentially private index. The dataset is scanned twice: 1) during the first step to construct the leaf nodes, a scan that is both necessary and sufficient for computing the histograms, and 2) during the encryption of data records. We explain now each step.

The first step computes each level of the hierarchy iteratively, starting from the *leaf nodes* at the bottom and ending with the single *root node* at the top. First, the leaf nodes (level 0) are instantiated. The number of leaf nodes is computed from the domain of the queriable attribute A_q and a unit-length interval that defines the length of each histogram bin. For example, if the attribute domain is from 0 to 100 and the unit-length interval is 1, then there are 100 leaf nodes with ranges $\{[0,1),[1,2),..,[99,100]\}$. Once the leaf nodes are created, then a scan of the cleartext dataset creates a pointer from the corresponding node to the data records and incrementing each bin's counter by 1. When the scan is completed, the leaf nodes have the correct numbers with associated pointers to the actual data records. Considering the example in Figure 2, the leaf nodes of the clear index have counts $3, 2, 1, 1$ for the corresponding ranges of $\{[0,1),\ [1,2),\ [2,3),\ [3,4]\}$. This means there are 3 data records whose values are within the range of $[0,1)$ and the first leaf node has pointers to these records. All nodes together at the same level construct a complete histogram over the domain $[0,4]$. The nodes of the upper levels are then computed such that each upper node points to a set of children nodes. The number of pointers that map to child nodes is set to the branching factor. The range of the bin of a node is a union of the children nodes' ranges, *i.e.,* given Φ^0 is the range partitioning of the leaf nodes $\Phi^0 \leftarrow (\phi_1^0, \ldots, \phi_k^0)$, the upper level range partitioning of the domain is $\Phi^1 \leftarrow \cup_{l=1}^{m}\left(\cup_{l}^{l+bf-1}\phi_l^0\right)$. The count of the bin is computed by summing the counts of the child nodes. In our toy example, the upper node for the range $[0,2)$ has a count of 5, which is the summation of two child nodes $[0,1)$ and $[1,2)$. As can be seen from the figure, the branching factor for the example is 2. This process goes on iteratively until a single node remains, which is the root node (highest level).

The second step perturbs the clear index to satisfy differential privacy. The differentially private computation of hierarchies of histograms has been extensively studied in the context of analytical queries. Although our context is different, PINED-RQ can benefit from the strategies proposed in [13], [4], [18] for distributing the privacy budget over the levels. Cormode et al. [4] compare a uniform budget allocation to a geometric budget allocation approach. The uniform budget allocation strategy allocates budgets to each level equally such that if the index has h levels, each level is allocated a budget of ϵ/h. In the geometric budget allocation, the allocated budget increases geometrically from the root to the leaves. The root receives the lowest budget, whereas the leaf nodes receive the highest budget. Cormode et al. demonstrate that geometric budgeting outperforms the uniform strategy as the height of an index increases. However, for shallower indexes ($h \leq 5$), both strategies are competitive. Qardaji et al. [18] also explore the effect of privacy budget allocation and do not recommend optimizing the privacy budget allocation as long as the optimal branching factor is selected. Our approach also uses a uniform budget allocation strategy. The straightforward approach to apply differential privacy is to sample a random noise from the Laplace distribution for each node bin and add the sampled noise to the original count. The total privacy budget is denoted by ϵ_{total}. Consider the example discussed before (Figure 2). The root node of the clear index has a count of 7 (range = $[0, 4]$). After the perturbation, the same node has a bin count of 8. This means the sampled noise for the bin is 1. Clearly, differential privacy causes a loss of utility in the index. Qardaji et al. [18] increase the utility of a histogram by generalizing the constrained inference approach proposed in [13]. The constrained inference provides consistency constraints between parent/children histograms, which has also been adopted by [4]. PINED-RQ also adopts the constrained inference approach to increase utility.

Unlike earlier approaches, we tackle a new problem while satisfying the differential privacy of the index. Earlier approaches target answering analytical range queries where the utility loss occurs due to the sampling of noises. However, the noise itself does not cause any problem since it only changes the count in the bins. This is also the case for the inner nodes in our index and our approach also updates the count of the node bins. However, in our context, the leaf nodes point to the data records which brings a novel challenge regarding noise sampling. We use the Laplace mechanism to sample noises and the sampled noises can be negative or positive as well. Therefore, our index construction handles two cases: 1) sampling of a positive noise, and 2) sampling of a negative noise, which are discussed next.

a) Positive noise sampling at leaf nodes: If the noise v sampled from the Laplace mechanism is positive, v dummy records are inserted at uniformly-random positions in the dataset and v additional pointers from the leaf node to the dummy records are created. Assuming bin b_i has a count of c_i, the updated count will be $c_i + v$. Note that each node in the index is perturbed separately and such a rule applies for each node that is perturbed with positive noise. Considering the example in Figure 2, the actual count of the node with bin range $[2, 3)$ is 1 in the clear index. This means there is only one real record in the real dataset that falls into this range. During perturbation, the sampled noise is $v = 1$ and the bin count of the same node in the differentially private index is 2. The differentially private index is outsourced to the cloud in the clear which means an adversary sitting in the cloud is able to see the counts of the nodes. In case a dummy record is not inserted into the actual dataset, the adversary would be able to see that there is only 1 pointer to the dataset from the node with the range $[2, 3)$ even if the bin count is 2. This would allow an adversary to infer the actual number of records within this range regardless of the node's count information. To hide such information from the adversary, inserting dummy records is necessary. This allows us to ensure that the number of pointers pointing to the real dataset matches the count inside the node. The encryption of data records with semantically secure encryption scheme, as will be explained later, prevents the adversary from distinguishing real records from the dummy records.

b) Negative noise sampling at leaf nodes: The handling of negative noise during the perturbation is a bit more complicated. When the sampled noise v is negative, this requires removing some data records from the dataset in order to ensure differential privacy (while still guaranteeing that they can be retrieved). Considering the toy example, the node with a range $[0, 1)$ is perturbed with noise -1 and a uniform-randomly selected single record -$(David, 0.7)$ in this example- in this range is removed from the actual dataset. Completely removing records from the dataset results in missing actual data during query processing. This is not reasonable as removing records might cause a decrease in recall (low utility). Although removing these records from the actual dataset is enough to ensure the differential privacy of the index, PINED-RQ introduces a new approach to handle removed records without violating differential privacy to achieve higher performance.

To handle removed records, PINED-RQ creates a fixed-sized *overflow array* for each leaf node. Basically, each leaf node is expanded by the size of an overflow array. Recall that the updated count of bin b_i is $p_i = c_i + v$ after the perturbation. Once an actual record is removed from a node, it is removed from the actual dataset and inserted into the corresponding overflow array. Then, each leaf node is expanded by the size of an overflow array. Assuming the size of an overflow array is o, the finalized b_i becomes $p_i + o$. If there are any records for a node that is removed as a result of sampling negative noise, it is inserted back to the dataset, and a pointer to this record is placed into the empty space in the leaf node. The dataset size is also expanded for each overflow array, given there are n leaf nodes, the dataset size is increased by $o * n$. This allows removed records to stay in the dataset. The pointer to the record is placed in the leaf node's overflow array. Considering our example, the removed record from the leaf node with range $[0, 1)$ is inserted into the overflow array for the same node. During perturbation, the other two leaf nodes are perturbed with noise 0 and one leaf node is perturbed with positive noise 1. Therefore, their overflow arrays do not contain any actual records. However, leaving them empty would reveal to an adversary that the data records in the overflow arrays are real data records. To ensure privacy, the number of real records in each array should be indistinguishable. Hence, the empty spaces in the arrays are padded with dummy data, so each overflow array consists of the same number of records. The size can be selected probabilistically large enough to store any removed records. Since the noise is sampled from the Laplace distribution, it can be selected based on the inverse of the cumulative distribution function (CDF) of the Laplace distribution with a very high probability. Note that it is possible for a sampled noise to be out of the selected bound. In this case, it is possible to append data records to their associated overflow arrays. This prevents PINED-RQ from missing the records. Later, we show that appending records to the associated overflow arrays does not violate PINED-RQ's end-to-end privacy guarantee. Note that although overflow records from the overflow arrays are theoretically possible, it is very unlikely to happen if the size of the overflow array is selected large enough.

Setting the parameters. Our histogram-based index structure shares similarities with the hierarchy of histograms proposed in [18]. Two important parameters, *branching factor* and *number of bins*, have a significant impact on the accuracy of the index. Qardaji et al. [18] discuss computing the branching factor and point out that when the number of bins is high, the optimal branching factor is around 16. The empirical analysis in [18] also highlights that the best performance is delivered when the branching factor is selected between 8 and 16. Therefore, we set the branching factor in PINED-RQ to 16.

B. Privacy Guarantees of the CREATE Function

We now show that the CREATE function of PINED-RQ satisfies $(\epsilon, \delta)_n$-Probabilistic-SIM-CDP as in Definition 6.

Theorem 2: Index $\mathcal{I}(A_q)$ satisfies ϵ_{total}-differential privacy (as defined in Definition 2) where ϵ_{total} is the budget dedicated to index perturbation, *i.e,* $\epsilon_{total} = \sum_{i=0}^{h} \epsilon_i$.

Proof: We consider neighboring datasets as two datasets, $\mathcal{D}_1$ and $\mathcal{D}_2$, differing in at most one record. Each node represents a bin of a histogram and a difference of a single record affects a maximum of one node: either increment or decrement count by 1. The maximum change 1 is used as a *global sensitivity* to sample noises from the Laplace distribution. Each bin is perturbed with a noise sampled from the Laplace mechanism, i.e., $\mathcal{L}(1/\epsilon_i)$. At each given level l_i, the set of bins (of histogram) satisfy ϵ_i-differential privacy (parallel composition theorem). Second, any given root-to-leaf path satisfies $\sum_i \epsilon_i$-differential privacy. As a result, thanks to the distribution of the privacy budget, $\sum_i \epsilon_i = \epsilon_{total}$. Moreover, since the composition of any function with a differentially-private function also satisfies differential privacy, the post-processing of the histograms in $\mathcal{I}(A_q)$ also satisfies ϵ_{total}-differential privacy. ■

Once PINED-RQ constructs the differentially private index, it encrypts each data record in $\mathcal{D}$ and overflow arrays with a semantically secure symmetric encryption scheme parameterized with the secret key χ while preserving the pointer relation. The output of the encryption is the encrypted dataset $\overline{\mathcal{D}}$ and encrypted overflow arrays. Note that semantic security guarantees that our computationally-bounded adversary has no way to distinguish encrypted dummy records from actual ones, neither in $\overline{\mathcal{D}}$ nor in encrypted overflow arrays. We demonstrate in Theorem 3 the overall guarantees of the CREATE function.

Theorem 3: The CREATE function, in charge of computing $\mathcal{I}(A_q)$, $\overline{\mathcal{D}}$, and the overflow arrays satisfies $(\epsilon, \delta)_n$-Probabilistic-SIM-CDP as defined in Definition 6.

Proof: We have shown above (Theorem 2) that computing $\mathcal{I}(A_q)$ satisfies ϵ_{total}-differential privacy. We start with the computation of $\overline{\mathcal{D}}$ and then consider the overflow arrays.

Computation of $\overline{\mathcal{D}}$. First, the records in $\overline{\mathcal{D}}$ are all encrypted with a semantically secure encryption scheme whether they are real records or dummy records. Loosely speaking, this means that a computationally-bounded adversary has negligible probability to distinguish encrypted real/dummy records from random bitstrings[2]. Due to this negligible leak, (information-theoretic) ϵ-differential privacy is not satisfied anymore. We thus switch to the ϵ_n-SIM-CDP computational analogue. Consider the pair made of the following two randomized functions : $\mathtt{f}^1$ that computes $\mathcal{I}(A_q)$, and $\mathtt{f}^2$ that encrypts the real/dummy records and outputs $\overline{\mathcal{D}}$. Consider the following equation : $|\mathtt{Pr}[\mathtt{A}_n(\{\mathtt{f}_n^1(\mathcal{D},\zeta), \mathtt{f}_n^2(\mathcal{D},\zeta)\}) = 1] - \mathtt{Pr}[\mathtt{A}_n(\mathtt{F}_n(\mathcal{D},\zeta)) = 1]|$. First, it is easy to see that $\mathtt{Pr}[\mathtt{A}_n(\{\mathtt{f}_n^1(\mathcal{D},\zeta), \mathtt{f}_n^2(\mathcal{D},\zeta)\}) = 1] = \mathtt{Pr}[\mathtt{A}_n(\mathtt{f}_n^1(\mathcal{D},\zeta)) = 1] + \mathtt{Pr}[\mathtt{A}_n(\mathtt{f}_n^2(\mathcal{D},\zeta)) = 1]$. Second, for $\mathtt{f}_n^2$, the definition of semantic security (Definition 4) yields : $\mathtt{Pr}[\mathtt{A}_n(\mathtt{f}_n^2(\mathcal{D},\zeta)) = 1] < \mathtt{Pr}[\mathtt{A}_n(\emptyset) = 1] + \frac{1}{p(n)} = \frac{1}{p(n)}$. Third, for $\mathtt{f}_n^1$, we have $\mathtt{Pr}[\mathtt{A}_n(\mathtt{f}_n^1(\mathcal{D},\zeta)) = 1] - \mathtt{Pr}[\mathtt{A}_n(\mathtt{F}_n(\mathcal{D},\zeta)) = 1] = 0$ because both $\mathtt{F}_n$ and $\mathtt{f}_n^1$ are ϵ-differentially private functions. As a result, $|\mathtt{Pr}[\mathtt{A}_n(\{\mathtt{f}_n^1(\mathcal{D},\zeta), \mathtt{f}_n^2(\mathcal{D},\zeta)\}) = 1] - \mathtt{Pr}[\mathtt{A}_n(\mathtt{F}_n(\mathcal{D},\zeta)) = 1]| = |\mathtt{Pr}[\mathtt{A}_n(\mathtt{f}_n^1(\mathcal{D},\zeta)) = 1] + \mathtt{Pr}[\mathtt{A}_n(\mathtt{f}_n^2(\mathcal{D},\zeta)) = 1] - \mathtt{Pr}[\mathtt{A}_n(\mathtt{F}_n(\mathcal{D},\zeta)) = 1]| < \frac{1}{p(n)}$. This is precisely the definition of ϵ_n-SIM-CDP (Definition 5).

Overflow arrays. Finally, let introduce the overflow arrays. First, we consider the case where the number of removed records does not exceed the size of any overflow array. In that case, ϵ_n-SIM-CDP is still satisfied because (1) there is exactly one overflow array per leaf, (2) each overflow array has the same size, and (3) all the records contained in an overflow array are encrypted by a semantically secure encryption scheme. Second, we consider the case where the number of removed records exceeds at least one overflow array. Note that this may happen only to the leaves that store a number of encrypted records higher than the capacity of an overflow array. Since the exceeding records are kept, appended to their corresponding overflow arrays, ϵ_n-SIM-CDP is not satisfied anymore for the corresponding leaves. Fortunately, we are able to decide on the probability that this case occurs, say δ, by computing from the Laplace cumulative distribution the minimum size that an overflow array should have to absorb negative noise. This results in satisfying ϵ_n-SIM-CDP with a probability of δ, which is precisely the definition of $(\epsilon, \delta)_n$-Probabilistic-SIM-CDP (Definition 6).

■

C. Query Processing Strategy

Static PINED-RQ deploys a simple query processing strategy to answer client requests. Given a range query, the query execution starts from the root of the index, and traverses the child of any node that has a non-negative intersection with the provided range. This is repeated recursively until the leaves of the index. In the leaf nodes, if a node has a positive count with the overlapping range query, then PINED-RQ returns the records pointed to by the corresponding node. If a leaf node is reached, independent of the node count, PINED-RQ returns the records in the overflow array for the corresponding node since PINED-RQ prioritizes high recall. Recall that PINED-RQ returns false positive and negative records in the result set. When data consumers receive the result set, they post-process to data to filter out the records that do not belong to the queried range. The remaining records are the set of correct results for the query.

[2] Note that semantic security leaks the size of the encrypted dataset, *i.e.,* the total number of encrypted records per leaf (the encrypted records are grouped by leaf). But this does not endanger ϵ-differential privacy because the number of encrypted records is equal to the perturbed count of their corresponding bin, which was perturbed to satisfy ϵ-differential privacy.

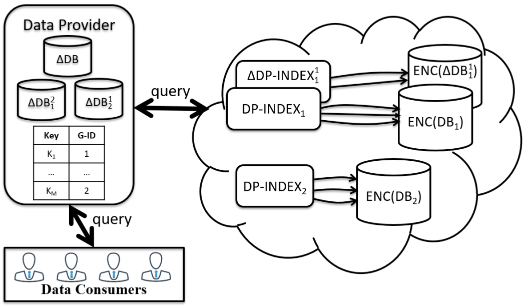


Fig. 3: Updates in PINED-RQ

IV. Dynamic Context : Insert & Mod/Del Functions

This section extends our description of PINED-RQ to support updates, namely, *insert*, *delete* or *modify* operations after the initial publication to the cloud server. We propose two new functions that satisfy our privacy guarantees: `INSERT` for handling insert operations, and `MOD/DEL` for handling modify and delete operations.

Supporting updates with a differentially private index raises several important issues: 1) How to manage a privacy budget over multiple updates? 2) How are updates reflected on the published differentially private index and the encrypted storage? 3) How does query processing work after the updates? PINED-RQ supports updates for append-only applications as well as applications with any number of inserts but a finite number of modifications and deletes to the existing records. Append-only data repositories are very popular and have widespread applications. Likewise, many real-world applications have finite updates to existing records while supporting insert operations, for example, university and medical databases. In the university case, students in a university have a finite number of updates to their grades, and it is unlikely that their grades changes once a student graduates. On the other hand, each year new students register to the university.

Data is published in the cloud with an initial index structure. A simple approach would process updates one by one, but given that PINED-RQ is differentially private and each publication consumes a fraction of the privacy budget, such an approach would exhaust the budget quickly. Therefore, PINED-RQ handles updates in batches. Queries are sent to the data provider rather than to the storage server. Hence, the data provider becomes a proxy and mediates the client-storage server communication. Proxy-based secure storage systems have been shown to deliver reasonable performance [17], [19], therefore, utilizing the data provider as a proxy does not cause drastic performance degradation in PINED-RQ.

A. General Approach

Assume a delete of record r_i is requested, PINED-RQ must handle this delete without compromising privacy. PINED-RQ does not remove the record from the published index and database - doing so would violate differential privacy (*e.g.,* tagging or removing a record directly from the database would reveal that r_i is not a dummy record). Instead, the data provider maintains small databases, called ΔDB and ΔDB_i^j where i denotes the publication index and j denotes the j^{th} batched modification/delete updates corresponding to the i^{th} publication (see Figure 3). PINED-RQ stores new insert operations in ΔDB. ΔDB_i^j are used to store modifications and deletes corresponding to the i^{th} publication, *i.e.,* ΔDB_1^1 and ΔDB_1^2 store updates related to the first publication. When the system is initialized and the initial dataset is published, the data provider stores modifications and deletes in ΔDB_1^1. With incoming modifications and deletes, ΔDB_1^1 grows and the data provider decides to publish ΔDB_1^1 to the cloud at some point. After ΔDB_1^1 is published to the cloud, the data provider stores modifications and deletes related to the first publication set in ΔDB_1^2. Inserts, on the other hand, are batched in ΔDB, which also grows until the data provider decides to publish to the cloud as DB_2 with its associated index structure.

In PINED-RQ, each *update* operation is a new insert to either ΔDB or one of the ΔDB_i^j databases. When a new insert operation is requested, the data provider inserts it to ΔDB. Although such action is obvious in the case of an *insert* operation, *deletes* and *modifies* result in inserting new records to the ΔDB_i^j corresponding to the modified/deleted record. Each record has an additional attribute indicating the type of operation, i.e., insert, delete or modify. ΔDB and ΔDB_1^1 are initially empty and after some updates, differentially private indexes for them are created and then published to the server.

B. INSERT Function for handling inserts

An interesting feature of inserts that PINED-RQ exploits is that newly inserted records can be grouped together and each new group is associated with a new privacy budget. Considering the university example discussed before, each year new students register to the university with no records in the published dataset. Instead of integrating new students to the previous publication, which would require using the partially consumed privacy budget, PINED-RQ associates these new students with a new publication and hence with a *full* new privacy budget.

1) Data Structures and Algorithms: Assume that the initially published dataset contains records with specific key values in attribute A_q, denoted by $\{K_1, ..., K_m\}$. Note that a key might have multiple records in the database, *e.g.,* records r_i and r_j might belong to K_1. Now consider an insert operation related to a new key K_{m+1}. The data provider will note that there is no intersection with earlier publications regarding K_{m+1}. Therefore, this new record can be associated with a new publication set. If there is an intersection with an earlier publication, the data provider would process K_{m+1} as a modify/delete. All keys $K_{1..m}$ are related with the first publication; however, K_{m+1} is mapped to the second publication. The data provider stores new inserts in ΔDB, and, after processing some number of updates, it publishes ΔDB to the cloud by constructing a secure index. Now, the cloud stores two encrypted databases, denoted by DB_1 and DB_2, and two differentially private indexes, DP-INDEX$_1$ and DP-INDEX$_2$, which point to DB_1 and DB_2, respectively. After this point, further insert operations will be associated with the third publication. In this way, PINED-RQ can use separate privacy budgets $(\epsilon_{total}, \delta_{total})$ for each set of publications using parallel composition, which ensures differential privacy.

If the initial publication of a set of inserts consumes the full privacy budget $(\epsilon_{total}, \delta_{total})$, it is not possible to perform further publications associated with that publication set. To allow the system to continue publishing, the initial publication should use an initial ϵ privacy budget denoted by ϵ_{init} such that $\epsilon_{init} < \epsilon_{total}$. The remaining budget is used to perform future publications.

2) Privacy Proof:

Theorem 4: The `INSERT` function satisfies $(\epsilon_{total}, \delta)_n$-Probabilistic-SIM-CDP while processing inserts.

Proof: PINED-RQ stores new inserts in ΔDB. After some time, it creates a differentially private index DP-INDEX$_i$ on ΔDB by dedicating some privacy budget denoted by ϵ_i. The created index DP-INDEX$_i$ is ϵ_i differentially private. Given $\epsilon_i \leq \epsilon_{total}$, DP-INDEX$_i$ satisfies ϵ_{total}-differential privacy. PINED-RQ creates each DP-INDEX$_i$ on a disjoint dataset, hence from parallel composition (Theorem 1), PINED-RQ with the `INSERT` function is $(\max(\epsilon_i), \delta)$-Probabilistic-SIM-CDP. Since $\epsilon_i \leq \epsilon_{total}$, PINED-RQ with `INSERT` also satisfies $(\epsilon_{total}, \delta)$-Probabilistic-SIM-CDP with new inserts. ■

C. MOD/DEL Function for Handling Modifies and Deletes

Assume a student objects to a grade after the grades have been submitted. As a consequence, this grade needs to be changed. As discussed earlier, a direct modification of a student's grade in the published database violates privacy guarantees. Therefore, PINED-RQ stores updates related to previous publications in ΔDB$_i^j$ where i corresponds to the publication DB$_i$ that contains the record corresponding to this student, and j refers to the j^{th} update batch of this publication.

Later, the data provider publishes these ΔDB$_i^j$ to the cloud. The main question is when to perform publications regarding modifications/deletes and how to allocate the privacy budget among multiple publications (publications to the same dataset need to share the same budget). Recall that the initial publication uses a privacy budget of ϵ_{init}. Hence, all future publications have to use the remaining privacy budget, denoted by ϵ_{rem} where $\epsilon_{rem} = \epsilon_{total} - \epsilon_{init}$, to create a differentially private index over ΔDB$_i^j$ and publish to the cloud. As stated earlier, it is possible to perform multiple ΔDB$_i^j$ publications. If this is the case, the remaining budget should also be shared among multiple ΔDB$_i^j$ publications. But the question is *how*?

1) When to Publish: The straightforward solution would be to perform periodic publications, *i.e.,* after some fixed time or some fixed number of updates. The challenge with this approach is deciding on the parameters. It is obvious there is a trade-off between executing queries through the data provider and the server. Therefore, we consider the cost in the decision process for performing publications. The cost might depend on many external factors, *e.g.,* location of servers, bandwidth, etc., but specifying these factors is beyond the scope of this paper. We assume there is a constant cost ratio between storing data at the data provider compared to the server, denoted by α. The decision is based on the following heuristic approach:

$$\alpha \times Relative\ Size \times \left(1 + \frac{\epsilon_{rem}}{\epsilon_{total}}\right) \geq 2\mu \qquad (1)$$

where *Relative Size* is the ratio of the *perturbed* data size stored in the data provider compared to the server and μ is a constant threshold parameter. ϵ_{rem} is always less than or equal to ϵ_{total}, *i.e.,* $\epsilon_{rem}/\epsilon_{total} \leq 1$. Thus, the ratio on the left hand side $(1 + \epsilon_{rem}/\epsilon_{total}) \leq 2$. In the best case, this ratio is 2, therefore, μ is multiplied by a constant factor of 2 to match the ratio $(1 + \epsilon_{rem}/\epsilon_{total})$. The right hand side is the minimum threshold that needs to be reached to trigger publication. Whenever the current ratio at the data provider (the left hand side of the inequality) exceeds the minimum threshold, the data provider constructs an index DP-INDEX$_i^j$ for ΔDB$_i^j$ and publishes the index along with the encrypted ΔDB$_i^j$ to the cloud. A higher μ results in less frequent publications in bigger batches. In our analysis, we found 2 to be reasonable for μ. In addition, higher α triggers more frequent publications. When the system is out of budget, no further publications can be performed, but the system can continue serving modifications/deletes using the data provider.

2) Allocating the Privacy Budget: PINED-RQ allocates a privacy budget for the publication in a novel way. Previous works on differentially private updates were in the context of data streams, which have different characteristics. A notable exception is [20], which requires a system administrator to decide on the total number of publications k upfront, where each publication consists of updates in batches. The budget is split equally among publications (i.e., each publication receives ϵ/k). However, this deterministic approach does not consider any external factors like dataset sizes. Thus, PINED-RQ uses the perturbed database size as a basis for allocation which considers the relative data storage at the data provider and ϵ_{rem}. Equations 2 and 3 compute the allocated privacy budget for ΔDB_i^j together.

$$\epsilon_{\Delta DB_i^j} = \epsilon_{rem} \times \frac{Size(\Delta DB_i^j)}{Size(DB_i + \sum_{l=1}^{j} \Delta DB_i^l)} \qquad (2)$$

As ϵ_{rem} decreases, the system requires a higher ratio between the size of the storage at the data provider versus that in the cloud to perform publication. However, increasing this ratio results in storing more data in the data provider. Note that it is possible for Equation 2 to allocate very small budgets. Therefore, PINED-RQ allows a system administrator to set a minimum budget threshold for the allocated privacy budget denoted by ϵ_{min}. If $\epsilon_{\Delta DB_i^j}$ is less than ϵ_{min}, the budget for publication $\epsilon_{\Delta DB_i^j}$ is set to ϵ_{min}. To achieve higher utility, PINED-RQ uses a simple motivation, namely, less budget for bigger datasets, and more budget for smaller datasets.

$$\epsilon_{\Delta DB_i^j} = \max(\epsilon_{\Delta DB_i^j}, \epsilon_{min}) \qquad (3)$$

As each publication has its own budget, the data provider performs computations for each publication independently to decide on the publication of ΔDB$_i^j$ and its budget $\epsilon_{\Delta DB_i^j}$.

3) Privacy Proof:

Theorem 5: PINED-RQ with the `MOD/DEL` function satisfies $(\epsilon_{total}, \delta^k)$-Probabilistic-SIM-CDP with modifications and deletes, where k is the number of non-disjoint publications.

Proof: The modifications and deletes on the published dataset are also new inserts into ΔDB_i^j, which is initially stored at the data provider. PINED-RQ applies a heuristic budget allocation mechanism for each publication, i.e., $\epsilon_{total} \geq \sum_{j=1}^{k} \epsilon_{\Delta DB_i^j}$ for any i where k is the number of publications on the set i. Hence from sequential composition (Theorem 1), PINED-RQ satisfies $(\epsilon_{total}, \delta^k)$-Probabilistic-SIM-CDP while processing modifications and deletes. ■

D. Executing Queries

In the steady state, the server might have multiple indexes, *e.g.,* DP-INDEX$_i$, ΔDP-INDEX$_i^j$, ΔDP-INDEX$_i^{j+1}$, DP-INDEX$_{i+1}$. Clients send queries to the data provider and the data provider partially answers the query based on its local information stored in ΔDB and ΔDB_i^j. In addition, the data provider redirects the query to the server. To retrieve the latest state, the server might need to process a query over all indexes as the data might be deleted or modified. Since each index is independent, parallel execution is possible and existing parallel query execution strategies can be directly applied.

V. Performance Evaluation

This section presents experimental results that demonstrate the efficiency and practicality of PINED-RQ by examining the effects of varying system configuration parameters. We implemented PINED-RQ in Java. All experiments are conducted on a machine running Windows 7 with i5-2320 3 GHz CPU and 8 GB memory. The branching factor (bf) is set to 16 and the total privacy budget ϵ_{total} to 1. The domain of A_q is normalized to $[0, 100]$. The size of overflow arrays is selected using the inverse of the CDF of the Laplace distribution for a given ϵ with 99.99% confidence interval.

Datasets. The experiments were performed with both synthetic and real datasets. To emulate real-world scenarios, the synthetic datasets follow two distributions: *uniform* or *Zipfian* with a skewness of 1, and contain 0.5 million records. For real datasets, we chose Gowalla [3], a social networking website where users share their locations by checking-in, and the US Postal Employees [1], USPS, dataset. The Gowalla dataset consists of $6,442,890$ check-in records. For query attribute, we use the 32-bit integer representation of check-in times. The experiments are performed on different sized datasets starting from 0.5 to 5 million records, created by choosing records uniformly from the complete Gowalla dataset. The USPS dataset compromises $624,414$ employees. We use *annual salary* as a query attribute and filtered out employee records with an hourly payment rate. After filtering, the dataset consists of $394,763$ records. The USPS dataset is highly skewed, whereas Gowalla is relatively uniform.

Query Set. In the experiments, we create various query sets of ranges corresponding to $1\%, 5\%, 10\%,\ 25\%,\ 50\%$, and 75% of the entire domain. For each set of query ranges, we sample 1000 queries uniformly over the domain. Unless stated, all other experiments are conducted using a uniform workload.

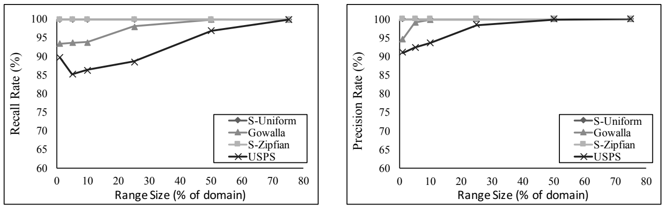
Evaluation metrics. The main metrics used are recall and precision. Note that PINED-RQ constructs a differentially private index; therefore, it does not always return the complete set of true records in a given range. It is also possible to have false records in the returned set. Therefore, the aim of PINED-RQ is to maintain privacy while achieving high recall and keeping the precision as high as possible. In addition, we measure the elapsed time for query execution.

A. Effect of Overflow Arrays

In this set of experiments, we analyze how the index construction mechanism performs in the presence or absence of overflow arrays. As discussed earlier, depending on the sampled noise, the recall of a query might drastically decrease. Our empirical findings validate this claim. These experiments are run on different datasets. The synthetic uniform and skewed datasets are denoted by *S-Uniform* and *S-Zipfian*, respectively. We use a variant of the Gowalla dataset with 0.5 million records, similar to the synthetic data sets and approximately equal to the USPS dataset of size $394,763$ records.

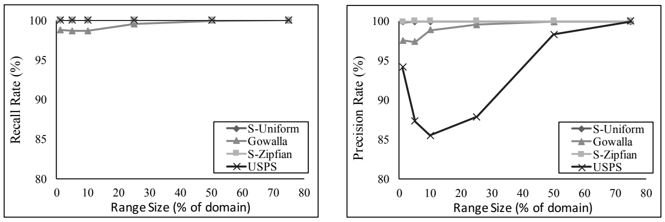
Without Overflow Arrays. Figure 4 shows the recall and precision results of PINED-RQ over different datasets by varying the query range size without deploying overflow arrays. In this setting, negatively sampled noise might cause the removal of a significant number of data records from the publication. Although PINED-RQ delivers high recall and precision for most of the cases, the performance with lower ranges sizes might not be satisfactory for both real datasets. For example, PINED-RQ delivers 85.15% recall and 92.26% precision for the USPS dataset when the range size is 5%. This is not the case for the synthetic datasets. During query execution, most of the decisions are made at the leaf level where each node covers 1 unit-length interval. The real datasets do not provide a perfect distribution so some of the nodes might cover very few records. If such nodes are perturbed with relatively high positive or low negative noises, the returned results include a number of false positive records or miss actual records. Therefore, the leaf nodes are more error-prone due to the added differentially private noise if a covered bin has a very low count. This is observed in Gowalla and USPS datasets. Some ranges have low counts and data removal causes query execution to suffer from low recall (e.g., as low as 85% for 5% range queries on the USPS dataset). This affects the overall recall performance for the USPS and Gowalla datasets. The removal of data records decreases recall for every dataset but is less observable for the synthetic datasets. The reason is that they follow almost a perfect distribution and leaf node bins have high enough counts (even the skewed one) which make the impact of removed records negligible. Although one can argue that PINED-RQ delivers good performance for the synthetic datasets even with no overflow array, real world application/datasets usually do not follow perfect data distribution.

With Overflow Arrays. When PINED-RQ deploys overflow arrays, it achieves almost 100% recall for all cases except small ranges executed over Gowalla (see Figure 5). Recalls for small ranges are relatively high $(98.6 - 98.8\%)$. PINED-RQ misses some records since query execution stops traversing



(a) Recall (b) Precision

Fig. 4: PINED-RQ without overflow arrays

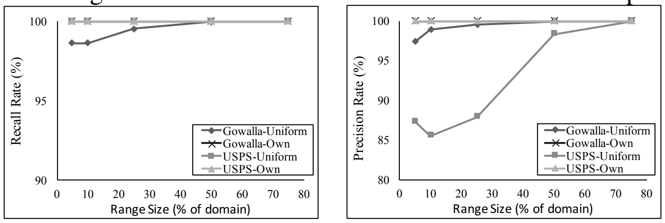


(a) Recall (b) Precision

Fig. 5: PINED-RQ with overflow arrays

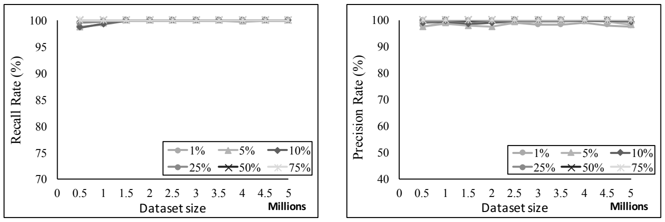
the index when overlapped ranges have a negative count. Some nodes have low counts as discussed before and the added differentially private noise misleads the query execution algorithm. Regardless of the dataset and the query range, the deployment of overflow arrays improves recall. This is also true for precision except in the case of USPS. While achieving higher recall, the precision drops to 85.52% from 93.53 when query size is 10%. Even in the worst case, achieving an 85.52% precision is decent. Note that both recall and precision are important in evaluating system performance. In this context, recall is more crucial and higher recall is preferable at the cost of lower precision most of the time. Note that the rest of the experiments use overflow arrays unless otherwise stated.

B. Skewness & Workload



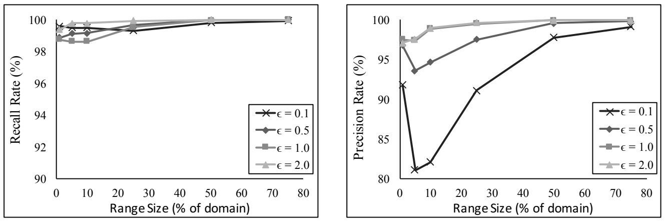
(a) Recall (b) Precision

Fig. 6: Effect of workloads



(a) Recall (b) Precision

Fig. 7: Scalability



(a) Recall (b) Precision

Fig. 8: Effect of privacy budget

When the data distribution is skewed, the precision performance of PINED-RQ is affected by the workload type (*e.g.,* uniform, skewed). If a query range covers the skewed area, PINED-RQ delivers high recall and precision. We analyzed this case in more detail and Figure 6 presents our findings. We use datasets *Gowalla* and *USPS*, and two different workloads: *uniform* or *own*. The workload tag "own" means the query set that is executed over the dataset follows the same distribution as the dataset, and hence is skewed too. This is realistic, as it would be expected that the more dense areas of the dataset will be more often queried. The first part of the tag in the legend describes the dataset itself and the second part describes the type of workload used in the experiments. The executions of uniform randomly selected queries on Gowalla and USPS were presented in Figure 5 and discussed in the previous subsection. When the workload follows a dataset's distribution, the queries are generated randomly following this distribution. Independent of the underlying dataset and range size, PINED-RQ delivers high recall and precision when the workload follows the dataset's own distribution. In all cases, both recall and precision are very close to 100%. This is not the case with uniform-randomly generated workloads, where performance depends on the queries. From our observations, precision and recall are quite high when the query range covers the skewed area and most of the queries in the skewed workloads cover the skewed area.

C. Scalability

We use variants of the Gowalla dataset with a scaling factor of 0.5 million to test scalability. Figure 7 shows the results with increasing size. Each curve in the graph represents a different query size. Although there are small fluctuations in terms of recall and precision from 500 thousand to 1.5 million records for small queries, 1% and 5%, after 1.5 million, the recall value does not fluctuate and achieves almost 100% for all cases, which is quite significant. Larger dataset size means higher counts in the nodes. The magnitude of noise sampled from the Laplace mechanism is independent of the dataset size. Therefore, as the dataset size increases, the impact of the perturbation noise becomes negligible. This is a significant design advantage of PINED-RQ. The precision results are also less sensitive to changes in dataset size. In all cases, PINED-RQ achieves $99-100\%$ precision, which is quite good from a system performance point of view.

D. Effect of Epsilon

The effect of a privacy budget in differentially private publications has been studied by many prior works and it is known that smaller privacy budgets provide less utility, since

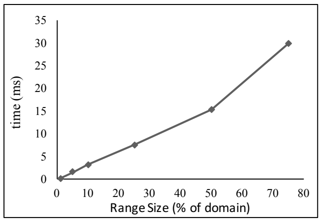


Fig. 9: Index scan time

less budget causes higher noise sampling from the Laplace distribution. This hypothesis is also valid in our system and more observable in precision. Thanks to its design, PINED-RQ is capable of delivering high recall even if the privacy budget is small. Moreover, PINED-RQ is expected to have higher precision with larger privacy budgets. The results of our experiments verifying this claim are presented in Figure 8. In the earlier experimental sections, the privacy budget was set to 1. When the budget is dropped to a 0.5 or 0.1, the smaller ranges suffer in terms of precision (down to 80% for range queries of size 5% in the worst case for $\epsilon = 0.1$). The recall rate slightly increases as the range size increases, since the added noise is quite small compared to the counts at the upper levels of the index. In the same case, the precision also increases. On the other hand, if the privacy budget is doubled, there is a slight improvement in recall with no significant improvement in terms of precision.

E. Index Scan Timing

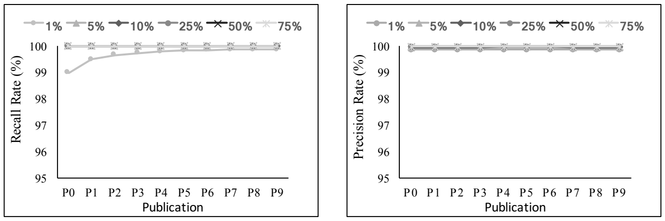
PINED-RQ maintains its index in the clear and this ensures fast query processing times. Figure 9 shows the average index scan times per query over different range sizes. We use our default dataset Gowalla dataset with 500,000 records in these experiments. The scan times are in the order of milliseconds. As the range size increases, the index scan time also increases since query processing has to consider more index nodes at the lower levels of the index. Compared to the related work discussed in Section VI, which perform heavy cryptographic computations during query execution, PINED-RQ is quite fast. Even for the largest sized range 75%, PINED-RQ scans the index in 29.78 **ms**. The execution of similarly sized range queries over similar datasets takes slightly less than 10^3 **seconds** in [5] when the most secure *Logarithmic-SRC* index model is deployed. [5] guarantees 100% recall - note that precision can also be quite low, e.g., 50% for small to medium sized ranges. In contrast, PINED-RQ achieves approximately 100% recall in almost all cases and in the worst case 85% precision. However, PINED-RQ does not guarantee 100% recall. This is a reasonable performance trade-off given the orders of magnitude improvement in execution times.

F. Updates

This section evaluates the performance of PINED-RQ's update management system. To simulate updates, we use an update-only uniform workload generator where 80% of the updates are new inserts, whereas the remaining 20% are modifications to earlier records. Due to the lack of space, we do not discuss the behavior of PINED-RQ for different update parameters. Here, we consider a specific scenario which fits well with the targeted application and evaluate PINED-RQ in terms of precision and recall. The results are presented in Figure 10. In this setting, we set ϵ_{total} to 1, the initial publication privacy budget ϵ_{init} to 0.7, the minimum publication budget ϵ_{min} to 0.3, the cost for storing data at the data provider α to 5, and the update frequency parameter μ to 2. The initial publication uses a synthetic uniform dataset of size $500k$ records. The system is tested after each publication to the server. To focus on the performance of indexes, the reported recall and precision rates do not consider the answers returned from the data provider (the results returned from the data provider have 100% recall and precision, which will obviously increase the recall and precision of query answers). The x-axis of the graphs in Figure 10 represents the publication id, i.e., the initial publication is denoted by $P0$ and the next publication is denoted by $P1$. There are 10 publications where nine of them are publications of inserts while $P7$ is a special publication consisting solely of modifications to $P0$, with a size of $307,693$ records, on the initially published dataset.

The initial publication uses a budget of 0.7. This is also true for other publications except for $P7$, which uses a budget of 0.3. Thanks to the parallel composition of differential privacy, PINED-RQ maintains its privacy while publishing new datasets with similar accuracy. Therefore, there is no significant effect on the performance of PINED-RQ while new datasets are published. The later publications only have a positive outcome since recall for the smallest range 1% starts from 99% with the initial publication and slightly increases with further updates. On the other hand, the precision for the same range is constant over the publications and is not affected by further publications. For all other ranges, PINED-RQ delivers decent performance with almost 100% recall and precision. Note that for each query, PINED-RQ scans all indexes in parallel, therefore, there is very small overhead in the index scan time compared to a single index case.

To compare our heuristic budget allocation mechanism, we ran experiments using the equal budget allocation [20]. Since there is no specific publication triggering mechanism in [20], we use PINED-RQ's publication triggering mechanism for both systems. Considering the initial publication, there are 7 publications where ϵ_{total} is distributed. This experimental setup only considers modifications/deletes which relies on sequential composition. ϵ_{init} is set to 0.1. In this setup, PINED-RQ's heuristic approach and the equal budget allocation provide approximately similar recall around 99.99%. The precision results are also very close to each other although our heuristic approach provides 0.1% more precision. This is simply the result of negligible noise sampling compared to the number of records on the leaves. When we ran similar experiments with a smaller dataset, e.g., 10k records, the difference between the two approaches is more obvious. PINED-RQ's budget allocation strategy provides $3-4\%$ more recall and 1% more precision, since the heuristic budget allocation strategy favors allocating more budget to the smaller dataset while the uniform budget allocation does not consider dataset size.



(a) Recall

(b) Precision

Fig. 10: PINED-RQ update performance

VI. Related Work

There have been many research efforts to improve the quality of privacy preserving histograms for aggregate queries, *e.g.,* [18]. In our work, we take advantage of these results to improve the quality of the histograms used in the nodes of PINED-RQ. Bucketization has been used for answering range queries over outsourced data [12]. These approaches are complementary to ours as they can be used for optimal distribution of the encrypted data in the outsourced database. Order preserving encryption (OPE) and its variations [2] have been also used for range query processing. Unlike PINED-RQ, they reveal the underlying data distribution and are vulnerable to statistical attacks. Eu-Jin Goh [9] proposed a secure index that allows a user with a trapdoor for a data x to test if the database contains x or not. The index reveals no information about the data for which the user does not have the trapdoor. The objective of PINED-RQ index is different from that of [9] as it targets range queries instead of exact match queries. The recent works by Li et al. [14] and Demertzis et al. [5] rely on *Searchable Symmetric Encryption* (SSE) To take advantage of SSE, Demertzis et al. propose three types of indexing approaches the most secure approach of which has high space requirements, *i.e.,* $O(n * m^2)$ where n is the database size and m the domain size. Even if the space requirement is improved, the proposed index scheme suffers from a high number of false positives along with execution times in the hundreds of seconds, unlike PINED-RQ which has execution times in the hundreds of milliseconds. The objective in [14], [5] is to provide index indistinguishability for the structure and node values in the index. PINED-RQ differs from this work in its objectives, as we aim at providing index differential privacy.

VII. Conclusion

PINED-RQ is a highly efficient and differentially private one-dimensional range query execution framework that constructs a novel differentially private index over an outsourced database. Unlike other differentially private systems, PINED-RQ is extended to support update operations. To the best of our knowledge, PINED-RQ is the first work that builds, uses and maintains a differentially private index for performing *selection range queries*. We have demonstrated the security of PINED-RQ and shown empirically its practicality and efficiency through extensive experiments performed on synthetic and real datasets. Future work includes generalizing the PINED-RQ approach to other index structures, especially those designed for higher dimension data.

Acknowledgment

This work is partly supported by NSF grants CNS-1528178, CNS-1649469, the Inria Bigdatanet project, and the European Union's Horizon 2020 - The EU Framework Programme for Research and Innovation 2014-2020, under grant agreement No. 732051.

References

[1] US Postal Employees. Data Universe, Asbury Park Press. http://php.app.com/agent/postalemployees/search.

[2] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *Proc. of ACM SIGMOD*, pages 563–574, 2004.

[3] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proc. of ACM SIGKDD*, pages 1082–1090, 2011.

[4] G. Cormode, C. M. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.

[5] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. N. Garofalakis. Practical private range search revisited. In *Proc. of SIGMOD*, pages 185–198, 2016.

[6] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[7] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[8] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3&#8211;4):211–407, 2014.

[9] E. Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.

[10] O. Goldreich. *The Foundations of Cryptography - Vol. 2, Basic Applications*. Cambridge University Press, 2004.

[11] O. Goldreich. Foundations of cryptography: a primer. *Found. Trends in Theoretical Computer Science*, 1(1):1–116, April 2005.

[12] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *Proc. of SIGMOD*, pages 216–227, 2002.

[13] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.

[14] R. Li, A. X. Liu, A. L. Wang, and B. Bruhadeshwar. Fast range query processing with strong privacy protection for cloud computing. *PVLDB*, 7(14):1953–1964, 2014.

[15] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proc. of ACM SIGMOD*, 2009.

[16] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. Computational Differential Privacy. In *Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '09, pages 126–142, Berlin, Heidelberg, 2009. Springer-Verlag.

[17] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In *Proc. of the ACM SOSP*, pages 85–100. ACM, 2011.

[18] W. H. Qardaji, W. Yang, and N. Li. Understanding hierarchical methods for differentially private histograms. *PVLDB*, 6(14):1954–1965, 2013.

[19] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich. Processing analytical queries over encrypted data. In *Proc. of VLDB*, pages 289–300, 2013.

[20] X. Zhang, X. Meng, and R. Chen. *Differentially Private Set-Valued Data Release against Incremental Updates*, pages 392–406. Springer Berlin Heidelberg, 2013.