The PINED-RQ Family: Differentially Private Indexes for Range Query Processing in Clouds

Tristan Allard (Univ Rennes, CNRS, Irisa)

Joint work with R. Akbarinia (INRIA, LIRMM) , A. El Abbadi (UCSB), L. d'Orazio (Univ Rennes, CNRS, IRISA), E. Pacitti (Univ Montpellier, LIRMM, CNRS), C. Sahin (UCSB), H. V. Tran (Univ Rennes, CNRS, IRISA)

Sosysec Seminar
13th Nov 2020

Thanks to C. Sahin and H. V. Tran for sharing some pictures and slides.

Who Am I ?

Main Focus

Design privacy-preserving personal data management and analysis systems and explore the resulting privacy-efficiency-quality tradeoffs

Main Tools

- **Distribution :** client-server architectures with untrusted parties, completely distributed architectures.
- **Sanitization models and mechanisms :** perturbation that satisfies differential privacy or variants
- **Encryption mechanisms :** block ciphers (*e.g.*, symmetric AES), homomorphic encryption (*e.g.*, additively-homomorphic Paillier), *etc.*

Leitmotiv: perturbation techniques as building blocks for privacy-preserving algorithms.

Why Using Differential Privacy as a Building Block ?

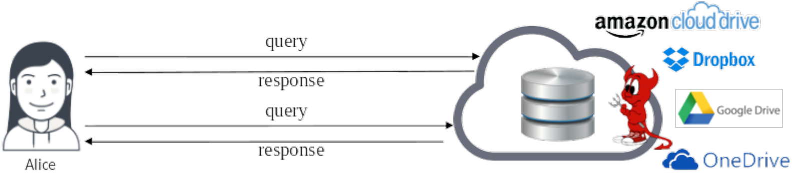
In general :

- Encryption alone : computation may be costly or may not cope with churn when distributed, the final result may reveal too much
- Differential privacy can allow to (for example) : switch to cleartext while keeping sound protections (perturbation), limit the leaks from the final result of encrypted functions
- It is interesting ! (security models, privacy budget management, algorithms adaptation)

A specific illustration below : the PINED-RQ family [16, 17] (and ongoing reviews).

Progress of the Talk

Outsourcing Private Data



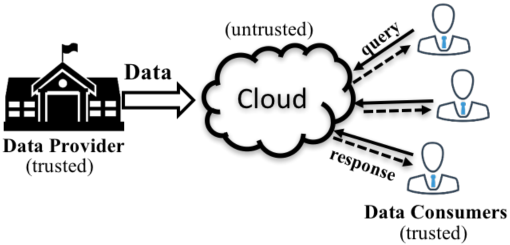
- Substantial advances in outsourced data management techniques but. . .
- Strong security concerns hamper the adoption of cloud solutions for private data
- And naive encryption of the complete database is not a viable solution

Secure Range Query ?

- Goal : Answer to queries that involve numerical comparisons with realistic performances
- Example query : `SELECT * FROM students WHERE grade≥3 AND grade≤4`
- A basic primitive for various applications (*e.g.*, transactions, analytics)

Objective

- Let a **honest-but-curious** cloud. . .
- Answer to **selection range queries** over encrypted personal data . . .
- While providing **differentially private guarantees**. . .
- Together with **realistic performances**.

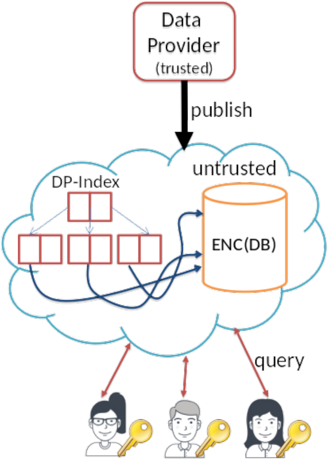


Related Work in a Nutshell

- Approaches based on bucketization do not provide formal privacy guarantees (*e.g.*, [10, 12, 11])
- Approaches based on order-preserving encryption schemes are vulnerable to statistical attacks (*e.g.*, [3, 4])
- Approaches based on symmetric searchable encryption suffer from high space and/or times requirements (*e.g.*, [13, 7])

Approach:

- **Data provider :** compute a one-dimensional **differentially-private index inspired from B+-Trees** over the **records encrypted** by any usual secret key semantically-secure encryption scheme (*e.g.*, AES), and send both to the cloud.
- **Cloud :** receive range queries (cleartext) and answer them by returning encrypted records based on the differentially-private (cleartext) index.
- **Both :** support updates ! (Inserts, modifies, and deletes.)



Progress of the Talk

Basic Data Structures

- **Dataset (private) :** a relation $\mathcal{D}(A_1, \ldots, A_d)$. Considered to be **private**.
- **Queries (public) :** non-aggregate single-dimensional range queries over a single attribute A_q.
- **Exact results (or relevant records) :** set of records $\mathcal{R}$ in $\mathcal{D}$ satisfying the query $\mathcal{Q}$.
- **Outsourced data structures (satisfy differential privacy) :**
  - Encrypted version of the dataset $\overline{\mathcal{D}}$ (semantically secure encryption scheme)
  - Index $\mathcal{I}(A_q)$ over the queriable attribute of $\mathcal{D}$, pointing to the encrypted records $\overline{r} \in \overline{\mathcal{D}}$.

Quality

- Inherent loss of information due to the differentially private perturbation
- Quantification of the quality we reach by recall and precision measures

Definition (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 $\mathtt{p} = |\mathcal{R} \cap \widetilde{\mathcal{R}}|/|\widetilde{\mathcal{R}}|$.

Original Differential Privacy Model

ϵ-differential privacy (from [8])

A **random function** `f` satisfies ϵ-differential privacy iff: **For all** $\mathcal{D}$ and $\mathcal{D}'$ **differing in at most one record**, and for any possible output $\mathcal{S}$ of `f`, then it is true that:
$\mathtt{Pr}[\mathtt{f}(\mathcal{D}) = \mathcal{S}] \leq e^{\epsilon} \times \mathtt{Pr}[\mathtt{f}(\mathcal{D}') = \mathcal{S}]$

- `f` : an agregate query perturbed (*originally*)
- "For all $\mathcal{D}$ and $\mathcal{D}'$": all possible datasets
- "$\mathcal{D}$ and $\mathcal{D}'$ differing in at most one record": $\mathcal{D}$ is $\mathcal{D}'$ with one tuple more or one tuple less
- ϵ : the privacy parameter, public, common values: 0.01, 0.1, $\ln 2$, $\ln 3$

Laplace Mechanism

Given ϵ, adding to the output of a `COUNT` aggregate query a random variable sampled from a Laplace distribution with mean 0 and scale factor $1/\epsilon$ satisfies ϵ-differential privacy [9].

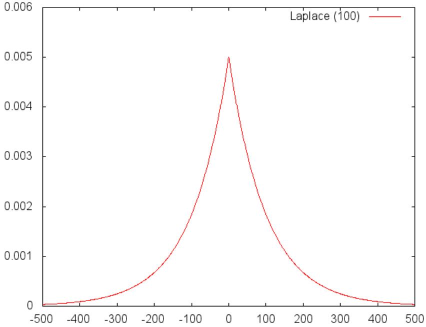


Figure: Laplace $(0, 1/0.01)$

Nice Properties

- **Self-composability** : composing the outputs of two independant releases sanitized by differentially-private function(s) satisfies differential privacy :
  - Where $\epsilon_{final} = \sum \epsilon_i$ if input datasets are **not** disjoint
  - Or $\epsilon_{final} = \max \epsilon_i$ otherwise
- **No breach from post-processing** : Any function applied to a differentially-private input produces a differentially-private output

Computational Differential Privacy

- Original differential privacy provides information theoretic guarantees...
- But when combined with a semantically secure encryption scheme, **the end-to-end guarantees become computational** !

Definition (ϵ_n-SIM-CDP privacy [14])

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 :

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

PINED-RQ Security Guarantees

A probabilistic relaxation of ϵ_n-SIM-CDP :

Definition ($(\epsilon, \delta)_n$-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 [14] to each individual with probability greater than or equal to δ, where $\delta \in]0, 1]$.

Problem(s) I

PINED-RQ

Design the following functions while satisfying $(\epsilon, \delta)_n$-Probabilistic-SIM-CDP and achieving realistic performance and quality levels :

1. **Index creation (`CREATE`) :** create the differentially private data structures
2. **Query execution :** answer to range queries based on the index
3. **Index updates (`INSERT`, `MOD/DEL`) :** maintain the index against inserts, modifies, and deletes operations *(will not be presented here for time reasons)*.

Problem(s) II

Extensions

- **Rationalize the software architecture (FRESQUE):** streamline and parallelize the `CREATION` and `INSERT` functions.
  $\Rightarrow$ Support high ingestion rates (100k+ records per second in our experiments).
- **Diversify the kind of index (PARADOT):** index the primary key attribute (based on PINED-RQ augmented with bitmaps).
  $\Rightarrow$ Support an infinite number of updates.

Progress of the Talk

Introduction

Problem Definition

PINED-RQ

The FRESQUE Extension (Quick Overview)

Conclusion

Step 1 : Compute $\mathcal{I}(A_q)$ (B+-Trees + DP histograms)

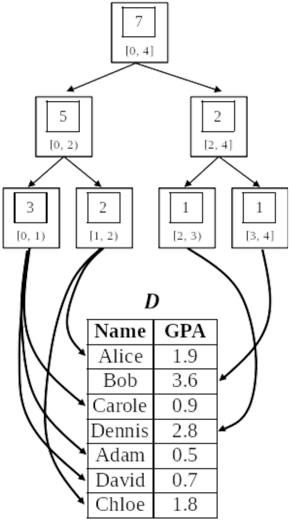


Figure: Step 1 *before perturbation*

Create the index $\mathcal{I}(A_q)$

- $\mathcal{I}(A_q)$ is a balanced tree with a fixed branching factor (see, *e.g.*, [15] for setting it)
- Each node is a bin, each level represents a histogram over A_q
- Leaf nodes are *unit-size* bins and point to encrypted records
- Perturb each bin based on the usual Laplace mechanism and apply known consistency constraints and privacy budget distribution strategies [6, 15]

Step 2 : Compute $\overline{\mathcal{D}}$ I

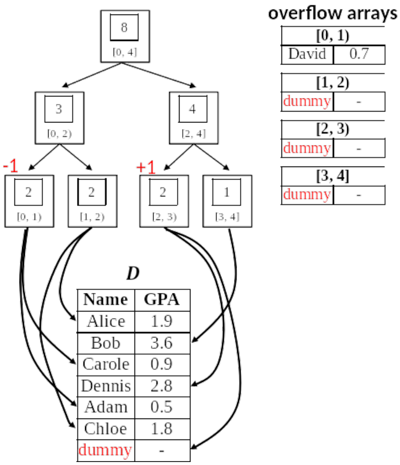


Figure: Step 2 *before encryption*

Create the encrypted dataset $\overline{\mathcal{D}}$

- **Main goal :** consistency between perturbed leaf nodes and number of encrypted records
- **Positive noise :** add dummy records
- **Negative noise :** remove records, put them in the *overflow array* of the leaf
- **Size of overflow arrays (fixed) :** computed based on Laplace CDF s. t. noise is lower with a given probability (*e.g.*, 99.99%)

Step 2 : Compute $\overline{\mathcal{D}}$ II

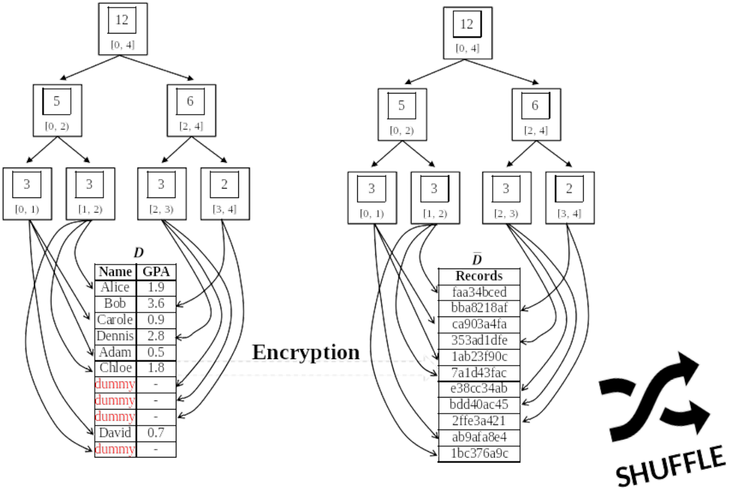


Figure: Step 2 *after encryption*

Privacy Guarantees of `CREATE`

Theorem

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 3.*

Proofs in the paper.

Query Processing Strategy

Naive Query Processing

Given a range query $\mathcal{Q}$:

1. Start at the root of the tree
2. Traverse the child of any node that has a non-negative intersection with $\mathcal{Q}$
3. If a leaf node has a positive count $\mathcal{Q}$: return the records pointed to by the corresponding node.
4. Otherwise : always return the overflow array (high recall priority).
5. (On data consumers) filter out false positives

Experimental results actually show that this strategy is sufficient !

Settings

- **Environment :** Java implementation, Windows 7 OS, i5-2320 3 GHz CPU, 8 GB RAM.
- **Parameters :** Branching factor is set to 16, total privacy budget $\epsilon_{total} = 1$, domain of A_q normalized to $[0, 100]$, size of overflow arrays with 99.99% confidence interval.
- **Synthetic datasets :** *uniform* or *Zipfian* with a skewness of 1, with 0.5 million records.
- **Real datasets :** Gowalla [5] (locations and times, $6,442,890$ records, relatively uniform), and US Postal Employees [1], USPS, dataset ($394,763$ records after cleaning, highly skewed).
- **Queries :** sample 1000 random queries inside each of the following ranges : $1\%, 5\%, 10\%,\ 25\%,\ 50\%$, and 75% of the entire domain.

Quality (sample) I

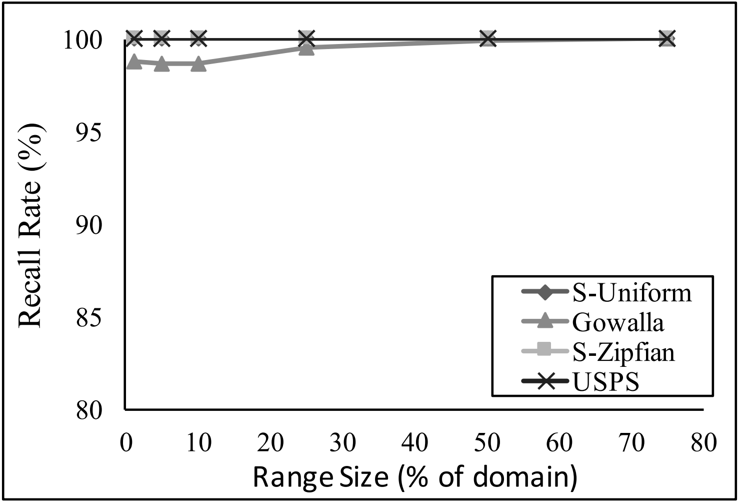


Figure: Recall (varying range size)

Quality (sample) II

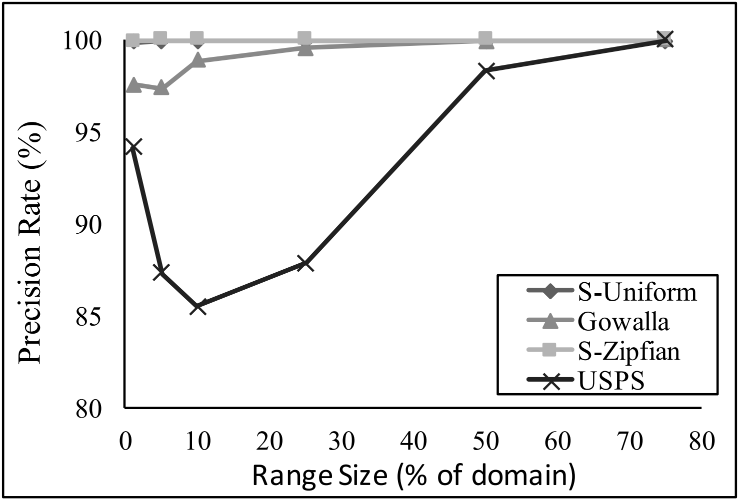


Figure: Precision (varying range size)

Quality (sample) III

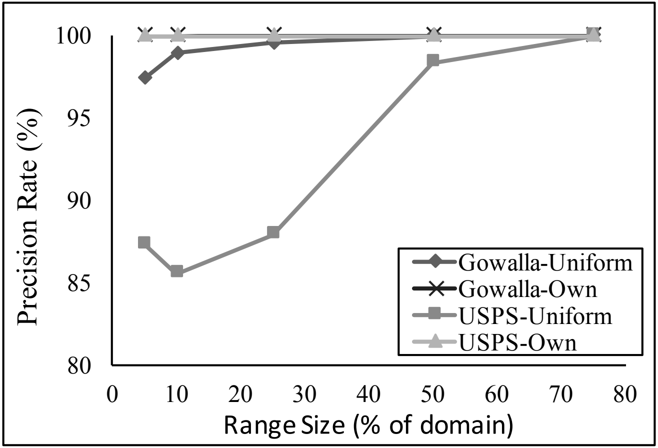


Figure: Precision (varying range size) - workload follows data distribution (own) or not (uniform)

Index Scan Timing

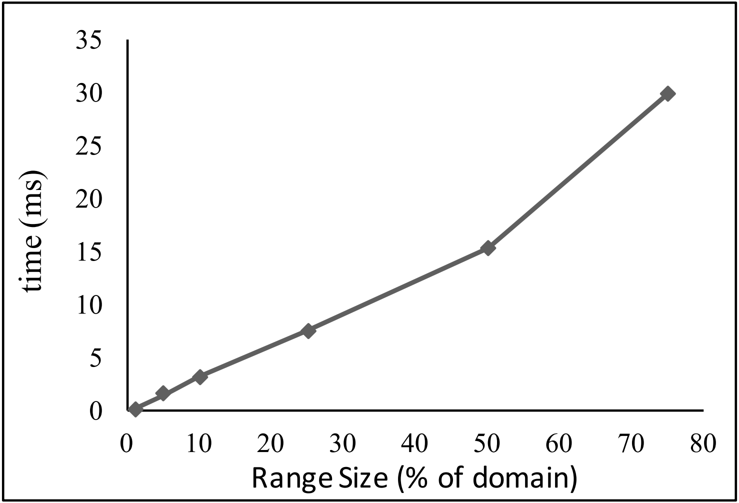


Figure: Index scan time (Gowalla, 500k records)

Progress of the Talk

Weaknesses of PINED-RQ

PINED-RQ generates bottlenecks

- **Batch publishing**
  ⇒ `FRESQUE` streams the incoming records to the cloud.
- **Naive data structures**
  ⇒ `FRESQUE` uses arrays extensively (constant-time accesses)
- **Naive architecture**
  ⇒ `FRESQUE` clearly defines the software modules of PINED-RQ and parallelizes them on a shared-nothing infrastructure.

Focus on the Architecture of FRESQUE

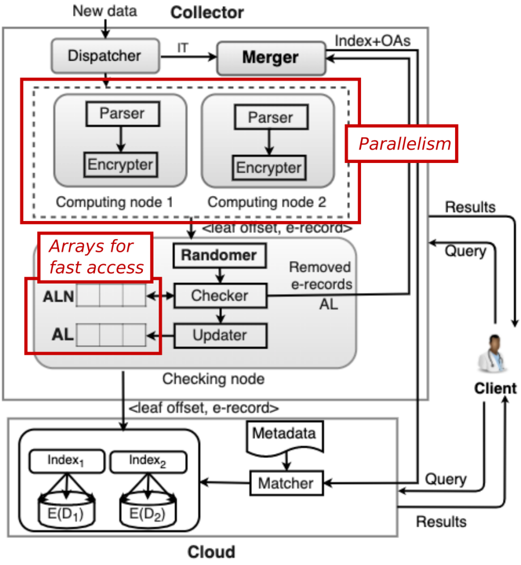


Figure: The Architecture of FRESQUE

Experimental Environment

- `FRESQUE` implemented in Java 1.8.0.
- Galactica platform, cluster of 17 nodes (Ubuntu 14.04.4 LTS)
- Datasets: NASA log [2] ($1,569,898$ records, five attributes, reply byte indexed), Gowalla [5] ($6,442,892$ records, three attributes, check-in time indexed).
- Incoming data rate: 200k records per second.
- PINED-RQ : fanout is 16, default ϵ is 1, δ is set to 99%.
- Results: average of ten experiments.

Table: Experimental environment

Component	**CPU (2.4 GHz)**	**Memory (GB)**	**Disk (GB)**
Dispatcher	4	8	80
Merger	4	8	80
Checking node	4	8	80
Computing node	2	2	20
Data source	4	16	80
Cloud	16	64	160

Ingestion Throughput

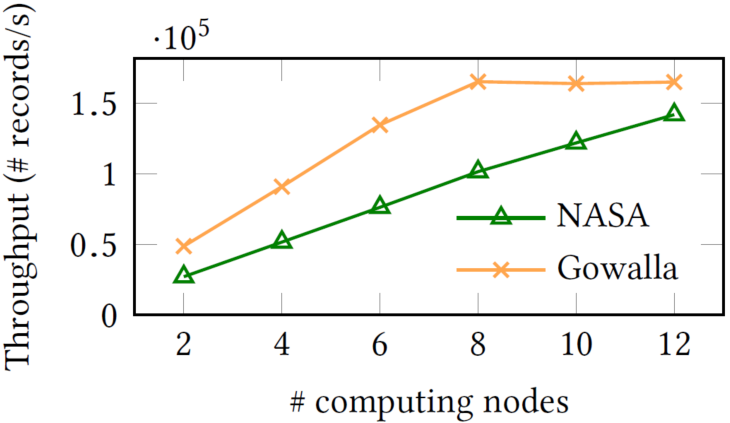


Figure: Ingestion throughput of FRESQUE

Ingestion Throughput - Comparison I

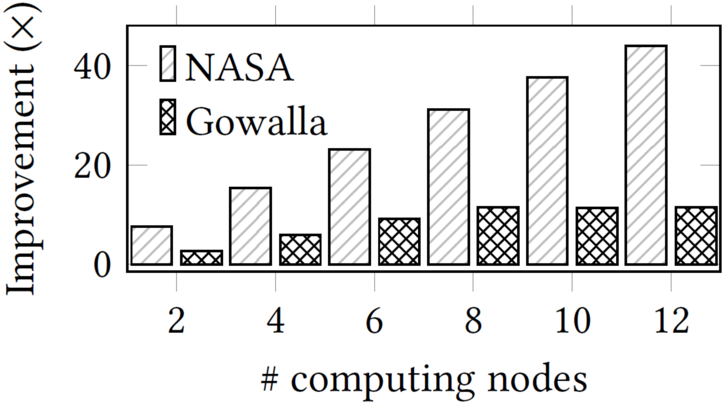


Figure: Ingestion throughput of FRESQUE - Improvement over our **non-parallel** version of PINED-RQ++ (≈ similar to PINED-RQ)

Ingestion Throughput - Comparison II

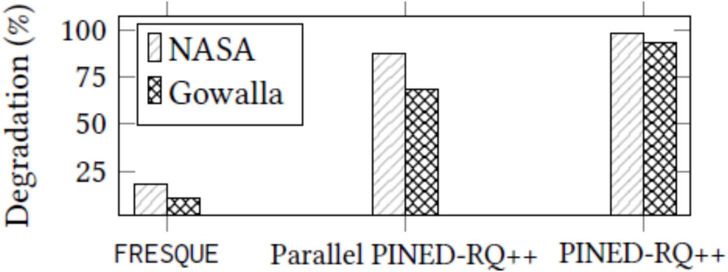


Figure: Ingestion throughput degradation of FRESQUE - Comparison with our previous versions

Progress of the Talk

Introduction

Problem Definition

PINED-RQ

The FRESQUE Extension (Quick Overview)

Conclusion

Key Take-Aways

- The PINED-RQ family : differential privacy together with encryption for efficient encrypted data querying
- Secure against a honest-but-curious cloud, where :
  - Security model is the **simulation-based flavor of computational differential privacy** [14]
  - **Differential privacy is satisfied probabilistically** for PINED-RQ and `FRESQUE`
  - Cleartext queries are assumed to be innocuous
- Overall, experimental evaluations show :
  - **High recall and precision levels**. Results depend on the size of the range (the larger the better), and on data distribution (high impact of the noise on the parts of the domain that contain few data)
  - **Index scan time is realistic** (much faster than related works).
  - **High ingestion throughput** for `FRESQUE` (orders of magnitude higher than related works).

Thank you !

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

[2] Nasa log.
http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html, 1996.

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

[4] A. Boldyreva, N. Chenette, and A. O'Neill.
Order-preserving encryption revisited: Improved security analysis and alternative solutions.
IACR Cryptology ePrint Archive, 2012:625, 2012.

[5] 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.

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

[7] 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.

[8] C. Dwork.
Differential privacy.
In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag.

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

[10] 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.

[11] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu.
Secure multidimensional range queries over outsourced data.
VLDB J., 21(3):333–358, 2012.

[12] B. Hore, S. Mehrotra, and G. Tsudik.
A privacy-preserving index for range queries.
In *Proc. of VLDB*, pages 720–731, 2004.

[13] 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.

[14] 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.

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

[16] C. Sahin, T. Allard, R. Akbarinia, A. El Abbadi, and E. Pacitti.
A differentially private index for range query processing in clouds.
In *ICDE '18*, 2018.

[17] H. V. Tran, T. Allard, L. d'Orazio, and A. El Abbadi.
Range query processing for monitoring applications over untrustworthy clouds.
In *EDBT '19*, 2019.