$\mathcal{E}$psolute: Efficiently Querying Databases While Providing Differential Privacy

Differential Privacy, ORAM, differential obliviousness, sanitizers

[42] DOI: 10.1145/3460120.3484786

Dmytro Bogatov, Georgios Kellaris, George Kollios, Kobbi Nissim, Adam O'Neill
`dmytro@bu.edu`, `kellaris@bu.edu`, `gkollios@cs.bu.edu`,
`kobbi.nissim@georgetown.edu`, `adamo@cs.umass.edu`

Built from *078878e4* on October 17, 2021

Boston University
Graduate School of Arts and Sciences
Department of Computer Science

Background

- With vast amounts of data, organizations choose to use cloud solutions
- These solutions need to be both efficient and secure
- Recent attacks on access pattern (AP) [8, 9, 16, 18, 19, 21, 26, 29, 31] and communication volume (CV) [21, 30, 31, 36, 40]
- Existing solutions may be insufficient:
  - protection against snapshot adversary does not account for AP and CV
    CryptDB [6], Arx [37], Seabed [22] and SisoSPIR [20]
  - enclaves like SGX are still uncommon and limited in memory
    Cipherbase [11], HardIDX [25], StealthDB [38], EnclaveDB [33], ObliDB [35], Opaque [27] and Oblix [32]
  - other solutions protect either from one of AP or CV, or use linear scan and full padding
    Cryptϵ [41], Shrinkwrap [28], SEAL [39] and PINED-RQ [34]
- $\mathcal{E}$psolute: most secure and practical range- and point-query engine in the outsourced database model, that protects both AP and CV using Differential Privacy, while not relying on TEE, linear scan or full padding

Motivation

- With vast amounts of data, organizations choose to use cloud solutions
- These solutions need to be both efficient and secure
- Recent attacks on access pattern (AP) [8, 9, 16, 18, 19, 21, 26, 29, 31] and communication volume (CV) [21, 30, 31, 36, 40]
- Existing solutions may be insufficient:
  - protection against snapshot adversary does not account for AP and CV
    CryptDB [6], Arx [37], Seabed [22] and SisoSPIR [20]
  - enclaves like SGX are still uncommon and limited in memory
    Cipherbase [11], HardIDX [25], StealthDB [38], EnclaveDB [33], ObliDB [35], Opaque [27] and Oblix [32]
  - other solutions protect either from one of AP or CV, or use linear scan and full padding
    Cryptϵ [41], Shrinkwrap [28], SEAL [39] and PINED-RQ [34]
- $\mathcal{E}$psolute: most secure and practical range- and point-query engine in the outsourced database model, that protects both AP and CV using Differential Privacy, while not relying on TEE, linear scan or full padding

Motivation

- With vast amounts of data, organizations choose to use cloud solutions
- These solutions need to be both efficient and secure
- Recent attacks on access pattern (AP) [8, 9, 16, 18, 19, 21, 26, 29, 31] and communication volume (CV) [21, 30, 31, 36, 40]
- Existing solutions may be insufficient:
  - protection against snapshot adversary does not account for AP and CV
    CryptDB [6], Arx [37], Seabed [22] and SisoSPIR [20]
  - enclaves like SGX are still uncommon and limited in memory
    Cipherbase [11], HardIDX [25], StealthDB [38], EnclaveDB [33], ObliDB [35], Opaque [27] and Oblix [32]
  - other solutions protect either from one of AP or CV, or use linear scan and full padding
    Cryptϵ [41], Shrinkwrap [28], SEAL [39] and PINED-RQ [34]
- $\mathcal{E}$psolute: most secure and practical range- and point-query engine in the outsourced database model, that protects both AP and CV using Differential Privacy, while not relying on TEE, linear scan or full padding

- With vast amounts of data, organizations choose to use cloud solutions
- These solutions need to be both efficient and secure
- Recent attacks on access pattern (AP) [8, 9, 16, 18, 19, 21, 26, 29, 31] and communication volume (CV) [21, 30, 31, 36, 40]
- Existing solutions may be insufficient:
  - protection against snapshot adversary does not account for AP and CV
    CryptDB [6], Arx [37], Seabed [22] and SisoSPIR [20]
  - enclaves like SGX are still uncommon and limited in memory
    Cipherbase [11], HardIDX [25], StealthDB [38], EnclaveDB [33], ObliDB [35], Opaque [27] and Oblix [32]
  - other solutions protect either from one of AP or CV, or use linear scan and full padding
    Cryptϵ [41], Shrinkwrap [28], SEAL [39] and PINED-RQ [34]
- $\mathcal{E}$psolute: most secure and practical range- and point-query engine in the outsourced database model, that protects both AP and CV using Differential Privacy, while not relying on TEE, linear scan or full padding

Motivation

- With vast amounts of data, organizations choose to use cloud solutions
- These solutions need to be both efficient and secure
- Recent attacks on access pattern (AP) [8, 9, 16, 18, 19, 21, 26, 29, 31] and communication volume (CV) [21, 30, 31, 36, 40]
- Existing solutions may be insufficient:
  - protection against snapshot adversary does not account for AP and CV
    CryptDB [6], Arx [37], Seabed [22] and SisoSPIR [20]
  - enclaves like SGX are still uncommon and limited in memory
    Cipherbase [11], HardIDX [25], StealthDB [38], EnclaveDB [33], ObliDB [35], Opaque [27] and Oblix [32]
  - other solutions protect either from one of AP or CV, or use linear scan and full padding
    Cryptϵ [41], Shrinkwrap [28], SEAL [39] and PINED-RQ [34]
- $\mathcal{E}$psolute: most secure and practical range- and point-query engine in the outsourced database model, that protects both AP and CV using Differential Privacy, while not relying on TEE, linear scan or full padding

Motivation

- With vast amounts of data, organizations choose to use cloud solutions
- These solutions need to be both efficient and secure
- Recent attacks on access pattern (AP) [8, 9, 16, 18, 19, 21, 26, 29, 31] and communication volume (CV) [21, 30, 31, 36, 40]
- Existing solutions may be insufficient:
  - protection against snapshot adversary does not account for AP and CV
    CryptDB [6], Arx [37], Seabed [22] and SisoSPIR [20]
  - enclaves like SGX are still uncommon and limited in memory
    Cipherbase [11], HardIDX [25], StealthDB [38], EnclaveDB [33], ObliDB [35], Opaque [27] and Oblix [32]
  - other solutions protect either from one of AP or CV, or use linear scan and full padding
    Cryptϵ [41], Shrinkwrap [28], SEAL [39] and PINED-RQ [34]
- $\mathcal{E}$psolute: most secure and practical range- and point-query engine in the outsourced database model, that protects both AP and CV using Differential Privacy, while not relying on TEE, linear scan or full padding

Database

A database is modeled as a set of tuples consisting of record *payload*, record *unique ID*, and record *search key* (i.e., the indexed attribute).

Query

A query is a predicate that is evaluated on a search key. Evaluating a query on a database results in all records whose search keys satisfy that query.

Protocols

User $\mathcal{U}$ and server $\mathcal{S}$ are *stateful*. In **setup**, $\mathcal{U}$ receives a (plaintext) database, $\mathcal{S}$ has no input. $\mathcal{S}$ outputs data structure $\mathcal{DS}$, $\mathcal{U}$ has no output. In **query**, $\mathcal{U}$ receives a query, $\mathcal{S}$ receives the data structure $\mathcal{DS}$. $\mathcal{U}$ outputs the result of evaluating the query on a database, $\mathcal{S}$ has no output.

Correctness

For correctness, we require that for any database and query, it holds that running the protocols yields for $\mathcal{U}$ the correct output with overwhelming probability.

Database

A database is modeled as a set of tuples consisting of record *payload*, record *unique ID*, and record *search key* (i.e., the indexed attribute).

Query

A query is a predicate that is evaluated on a search key. Evaluating a query on a database results in all records whose search keys satisfy that query.

Protocols

User $\mathcal{U}$ and server $\mathcal{S}$ are *stateful*. In **setup**, $\mathcal{U}$ receives a (plaintext) database, $\mathcal{S}$ has no input. $\mathcal{S}$ outputs data structure $\mathcal{DS}$, $\mathcal{U}$ has no output. In **query**, $\mathcal{U}$ receives a query, $\mathcal{S}$ receives the data structure $\mathcal{DS}$. $\mathcal{U}$ outputs the result of evaluating the query on a database, $\mathcal{S}$ has no output.

Correctness

For correctness, we require that for any database and query, it holds that running the protocols yields for $\mathcal{U}$ the correct output with overwhelming probability.

Outsourced Database System

Database

A database is modeled as a set of tuples consisting of record *payload*, record *unique ID*, and record *search key* (i.e., the indexed attribute).

Query

A query is a predicate that is evaluated on a search key. Evaluating a query on a database results in all records whose search keys satisfy that query.

Protocols

User $\mathcal{U}$ and server $\mathcal{S}$ are *stateful*. In **setup**, $\mathcal{U}$ receives a (plaintext) database, $\mathcal{S}$ has no input. $\mathcal{S}$ outputs data structure $\mathcal{DS}$, $\mathcal{U}$ has no output. In **query**, $\mathcal{U}$ receives a query, $\mathcal{S}$ receives the data structure $\mathcal{DS}$. $\mathcal{U}$ outputs the result of evaluating the query on a database, $\mathcal{S}$ has no output.

Correctness

For correctness, we require that for any database and query, it holds that running the protocols yields for $\mathcal{U}$ the correct output with overwhelming probability.

Database

A database is modeled as a set of tuples consisting of record *payload*, record *unique ID*, and record *search key* (i.e., the indexed attribute).

Query

A query is a predicate that is evaluated on a search key. Evaluating a query on a database results in all records whose search keys satisfy that query.

Protocols

User $\mathcal{U}$ and server $\mathcal{S}$ are *stateful*. In **setup**, $\mathcal{U}$ receives a (plaintext) database, $\mathcal{S}$ has no input. $\mathcal{S}$ outputs data structure $\mathcal{DS}$, $\mathcal{U}$ has no output. In **query**, $\mathcal{U}$ receives a query, $\mathcal{S}$ receives the data structure $\mathcal{DS}$. $\mathcal{U}$ outputs the result of evaluating the query on a database, $\mathcal{S}$ has no output.

Correctness

For correctness, we require that for any database and query, it holds that running the protocols yields for $\mathcal{U}$ the correct output with overwhelming probability.

Definition (Differential Privacy, adapted from [3, 4])

A randomized algorithm A is (ϵ, δ)-differentially private if for all $\mathcal{D}_1 \sim \mathcal{D}_2 \in \mathcal{X}^n$, and for all subsets $\mathcal{O}$ of the output space of A,

$$\Pr\left[A\left(\mathcal{D}_1\right) \in \mathcal{O}\right] \leq \exp(\epsilon) \cdot \Pr\left[A\left(\mathcal{D}_2\right) \in \mathcal{O}\right] + \delta\ .$$

Theorem (Laplace Perturbation Algorithm (LPA), adapted Theorem 1 from [3])

An algorithm A that adds independently generated noise from a ***zero-mean Laplace distribution*** *with scale $\lambda = \frac{\text{sensitivity of query}}{\epsilon}$ to each coordinate of a query result, satisfies ϵ-differential privacy.*

Definition (Differentially Private Sanitizer, informal)

An $(\epsilon, \delta, \alpha, \beta)$-differentially private sanitizer is a pair of algorithms (A, B) such that:

- A is (ϵ, δ)-differentially private, and
- on input a dataset, A outputs a data structure $\mathcal{DS}$ such that with probability $1 - \beta$ for all queries, $B\left(\mathcal{DS}, q\right)$ is within α of a real query result.

Definition (Differential Privacy, adapted from [3, 4])

A randomized algorithm A is (ϵ, δ)-differentially private if for all $\mathcal{D}_1 \sim \mathcal{D}_2 \in \mathcal{X}^n$, and for all subsets $\mathcal{O}$ of the output space of A,

$$\Pr\left[\mathrm{A}\left(\mathcal{D}_1\right) \in \mathcal{O}\right] \leq \exp(\epsilon) \cdot \Pr\left[\mathrm{A}\left(\mathcal{D}_2\right) \in \mathcal{O}\right] + \delta \ .$$

Theorem (Laplace Perturbation Algorithm (LPA), adapted Theorem 1 from [3])

An algorithm A that adds independently generated noise from a ***zero-mean Laplace distribution*** *with scale $\lambda = \frac{\text{sensitivity of query}}{\epsilon}$ to each coordinate of a query result, satisfies ϵ-differential privacy.*

Definition (Differentially Private Sanitizer, informal)

An $(\epsilon, \delta, \alpha, \beta)$-differentially private sanitizer is a pair of algorithms (A, B) such that:

- A is (ϵ, δ)-differentially private, and
- on input a dataset, A outputs a data structure $\mathcal{DS}$ such that with probability $1 - \beta$ for all queries, $\mathrm{B}\left(\mathcal{DS}, q\right)$ is within α of a real query result.

Definition (Differential Privacy, adapted from [3, 4])

A randomized algorithm A is (ϵ, δ)-differentially private if for all $\mathcal{D}_1 \sim \mathcal{D}_2 \in \mathcal{X}^n$, and for all subsets $\mathcal{O}$ of the output space of A,

$$\Pr\left[\mathrm{A}\left(\mathcal{D}_1\right) \in \mathcal{O}\right] \leq \exp(\epsilon) \cdot \Pr\left[\mathrm{A}\left(\mathcal{D}_2\right) \in \mathcal{O}\right] + \delta \ .$$

Theorem (Laplace Perturbation Algorithm (LPA), adapted Theorem 1 from [3])

An algorithm A that adds independently generated noise from a ***zero-mean Laplace distribution*** *with scale* $\lambda = \frac{\text{sensitivity of query}}{\epsilon}$ *to each coordinate of a query result, satisfies* ϵ*-differential privacy.*

Definition (Differentially Private Sanitizer, informal)

An $(\epsilon, \delta, \alpha, \beta)$-differentially private sanitizer is a pair of algorithms (A, B) such that:

- *A* is (ϵ, δ)-differentially private, and
- on input a dataset, A outputs a data structure $\mathcal{DS}$ such that with probability $1 - \beta$ for all queries, $\mathrm{B}\left(\mathcal{DS}, q\right)$ is within α of a real query result.

Answering point and range queries with differential privacy. N is a domain size. Omitting the dependency on ϵ and δ, shown are values of α.

	Point Query	Range Query
Pure DP	$\Theta(\log N)$ [15]	$\Theta(\log N)$ [5, 13]
Approximate DP	$\mathcal{O}(1)$ [12]	$\mathcal{O}\left(2^{\log^* N}\right)$ [12, 17]

Theorem *(Composition)*

Let $A_1, \ldots, A_r$ be mechanisms, such that each A_i provides ϵ_i-differential privacy. Let $\mathcal{D}_1, \ldots, \mathcal{D}_r$ be pairwise non-disjoint (resp., disjoint) datasets. Let A be another mechanism that executes $A_1(\mathcal{D}_1), \ldots, A_r(\mathcal{D}_r)$ using independent randomness for each A_i, and returns their outputs. Then, mechanism A is $\left(\sum_{i=1}^{r} \epsilon_i\right)$-differentially private (resp., $\left(\max_{i=1}^{r} \epsilon_i\right)$-differentially private).

Answering point and range queries with differential privacy. N is a domain size. Omitting the dependency on ϵ and δ, shown are values of α.

	Point Query	Range Query
Pure DP	$\Theta(\log N)$ [15]	$\Theta(\log N)$ [5, 13]
Approximate DP	$\mathcal{O}(1)$ [12]	$\mathcal{O}(2^{\log^* N})$ [12, 17]

Theorem (Composition)

Let $A_1, \ldots, A_r$ be mechanisms, such that each A_i provides ϵ_i-differential privacy. Let $\mathcal{D}_1, \ldots, \mathcal{D}_r$ be pairwise non-disjoint (resp., disjoint) datasets. Let A be another mechanism that executes $A_1(\mathcal{D}_1), \ldots, A_r(\mathcal{D}_r)$ using independent randomness for each A_i, and returns their outputs. Then, mechanism A is $\left(\sum_{i=1}^{r} \epsilon_i\right)$-differentially private (resp., $\left(\max_{i=1}^{r} \epsilon_i\right)$-differentially private).

Access pattern and ORAM

Access pattern is a sequence of memory accesses $\mathbf{y}$, where each access consists of the memory *location o*, read $\mathbf{r}$ or write $\mathbf{w}$ *operation* and the *data d* to be written.

Oblivious RAM (ORAM) is a mechanism that hides the accesses pattern. ORAM is a protocol between $\mathcal{C}$ (who accesses) and $\mathcal{S}$ (who stores), with a guarantee that the view of the server is *indistinguishable* for any two sequences of the same lengths.

$$|\mathbf{y}_1| = |\mathbf{y}_2|$$
$$\textsc{View}_{\mathcal{S}}(\mathbf{y}_1) \overset{c}{\approx} \textsc{View}_{\mathcal{S}}(\mathbf{y}_2)$$

ORAM protocol

Client $\mathcal{C}$		**Server** $\mathcal{S}$
$\mathbf{y} = (\mathbf{r}, i, \perp)\vert_{i=1}^{5}$		
(client state)	$\xleftrightarrow{\text{ORAM}\,(\mathbf{y})}$	(server state)
$\{d_1, d_2, d_3, d_4, d_5\}$		

For example: Square Root ORAM [1], Hierarchical ORAM [2], Binary-Tree ORAM [7], Interleave Buffer Shuffle Square Root ORAM [24], TP-ORAM [10], **Path-ORAM** [14] and TaORAM [23]. ORAM incurs at least logarithmic overhead in the number of stored records. [2]

Contributions: Model, Single-Threaded and Parallel $\mathcal{E}$psolute

Dmytro Bogatov
Boston University

Definition (Computationally Differentially Private Outsourced Database System (CDP-ODB))

We say that an outsourced database system Π is (ϵ, δ)-computationally differentially private (a.k.a. CDP-ODB) if for every polynomial time distinguishing adversary $\mathcal{A}$, for every neighboring databases $\mathcal{D} \sim \mathcal{D}'$, and for every query sequence $q_1, \ldots, q_m \in \mathcal{Q}^m$ where $m = \text{poly}(\lambda)$,

$$\Pr\left[\mathcal{A}\left(1^\lambda, \textsc{View}_{\Pi,\mathcal{S}}\left(\mathcal{D}, q_1, \ldots, q_m\right)\right) = 1\right] \leq \exp \epsilon \cdot \Pr\left[\mathcal{A}\left(1^\lambda, \textsc{View}_{\Pi,\mathcal{S}}\left(\mathcal{D}', q_1, \ldots, q_m\right)\right) = 1\right] + \delta + \text{negl}(\lambda) ,$$

the probability is over the randomness of the distinguishing adversary $\mathcal{A}$ and the protocol Π.

Note:

- Entire view of the adversary is DP-protected
- Implies protection against communication volume and access pattern leakages
- Query sequence $q_1, \ldots, q_m \in \mathcal{Q}^m$ is fixed
- $\text{negl}(\lambda)$ accounts for the computational (as opposed to theoretical) DP definition

Definition (Computationally Differentially Private Outsourced Database System (CDP-ODB))

We say that an outsourced database system Π is (ϵ, δ)-computationally differentially private (a.k.a. CDP-ODB) if for every polynomial time distinguishing adversary $\mathcal{A}$, for every neighboring databases $\mathcal{D} \sim \mathcal{D}'$, and for every query sequence $q_1, \ldots, q_m \in \mathcal{Q}^m$ where $m = \text{poly}(\lambda)$,

$$\Pr\left[\mathcal{A}\left(1^\lambda, \textsc{View}_{\Pi,\mathcal{S}}\left(\mathcal{D}, q_1, \ldots, q_m\right)\right) = 1\right] \leq \exp\epsilon \cdot \Pr\left[\mathcal{A}\left(1^\lambda, \textsc{View}_{\Pi,\mathcal{S}}\left(\mathcal{D}', q_1, \ldots, q_m\right)\right) = 1\right] + \delta + \text{negl}(\lambda) \ ,$$

the probability is over the randomness of the distinguishing adversary $\mathcal{A}$ and the protocol Π.

Note:

- Entire view of the adversary is DP-protected
- Implies protection against communication volume and access pattern leakages
- Query sequence $q_1, \ldots, q_m \in \mathcal{Q}^m$ is fixed
- $\text{negl}(\lambda)$ accounts for the computational (as opposed to theoretical) DP definition

On impossibility of adaptive queries

Example

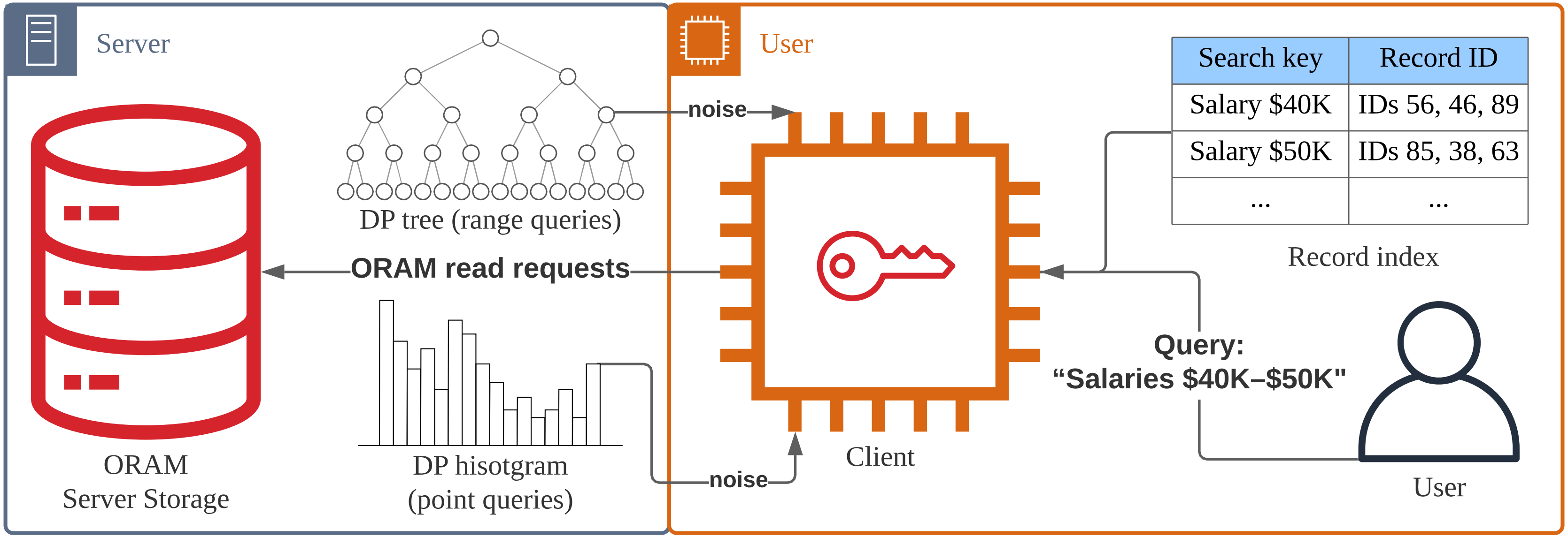
- Suppose neighboring medical databases differ in one record with a rare diagnosis “Alzheimer’s disease”
- A medical professional queries the database
  - for that diagnosis first (point query)

    `SELECT name FROM patients WHERE condition = 'ALZ'`
  - if there is a record, she queries the senior patients next (range query)

    `SELECT name FROM patients WHERE age >= 65`
  - otherwise she queries the general population,resulting in more records

    `SELECT name FROM patients`
- Efficient system cannot return nearly the same number of records in both cases, thus, the adversary can distinguish

Single-Threaded $\mathcal{E}$psolute



- In setup protocol, lookup index over the records is stored on the client $\mathcal{U}$ and all records are stored on the server $\mathcal{S}$ via ORAM (record ID is the ORAM location).
- In query protocol, the ORAM locations are retrieved from the index, the total (real + noise) records number is fetched from the sanitized $\mathcal{DS}$, records are retrieved via ORAM.
- Refer to the full paper [42] for formal description and security proof.

Sanitizers for point and range queries

Point queries

- Use the LPA method as the sanitizer to ensure pure differential privacy
- For every histogram bin, draw from $\textsc{Laplace}\left(\alpha, \frac{1}{\epsilon}\right)$

Range queries

- Use the aggregate tree method as the sanitizer, a k-ary tree over domain ($k = 16$)
  - a leaf node holds the number of records falling into each bin plus some noise;
  - a parent node holds sum of the leaf values in the range covered by this node, plus noise;
  - sanitizer's output is the best-range cover;
- For every node, draw from $\textsc{Laplace}\left(\alpha, \frac{\log_k N}{\epsilon}\right)$

Set α to the smallest such that if drawn N times, values are positive with probability $1 - \beta$.

$$\alpha = \left\lceil -\frac{\ln\left(2 - 2\sqrt[N]{1-\beta}\right)}{\epsilon} \right\rceil$$

$$\alpha = \left\lceil -\frac{\ln\left(2 - 2\sqrt[\text{nodes}]{1-\beta}\right) \cdot \log_k N}{\epsilon} \right\rceil$$

Point queries

- Use the LPA method as the sanitizer to ensure pure differential privacy
- For every histogram bin, draw from $\textsc{Laplace}\left(\alpha, \frac{1}{\epsilon}\right)$

Range queries

- Use the aggregate tree method as the sanitizer, a k-ary tree over domain ($k = 16$)
  - a leaf node holds the number of records falling into each bin plus some noise;
  - a parent node holds sum of the leaf values in the range covered by this node, plus noise;
  - sanitizer's output is the best-range cover;
- For every node, draw from $\textsc{Laplace}\left(\alpha, \frac{\log_k N}{\epsilon}\right)$

Set α to the smallest such that if drawn N times, values are positive with probability $1 - \beta$.

$$\alpha = \left\lceil - \frac{\ln\left(2 - 2\sqrt[N]{1-\beta}\right)}{\epsilon} \right\rceil$$

$$\alpha = \left\lceil - \frac{\ln\left(2 - 2\sqrt[\text{nodes}]{1-\beta}\right) \cdot \log_k N}{\epsilon} \right\rceil$$

Overhead and supporting multiple attributes

Storage efficiency is defined as the sum of the bit-lengths of the records in a database relative to the bit-length of a corresponding encrypted database. **Communication efficiency** is defined as the sum of the lengths of the records in bits whose search keys satisfy the query relative to the actual number of bits sent back as the result of a query. Efficiency is defined as (a_1, a_2), where a_1 is a multiplicative term and a_2 is the additive one.

$\mathcal{E}$psolute's storage efficiency is $(\mathcal{O}(1), 0)$. Communication efficiencies for different query and DP types are shown.

	Point Query	Range Query
Pure DP	$(\mathcal{O}(\log n), \mathcal{O}(\log N \log n))$	$(\mathcal{O}(\log n), \mathcal{O}(\log N \log n))$
Approximate DP	$(\mathcal{O}(\log n), \mathcal{O}(\log n))$	$(\mathcal{O}(\log n), \mathcal{O}(2^{\log^* N} \log n))$

Multiple indexed attributes

- Naïve way is to duplicate the entire stack of states of $\mathcal{U}$ and $\mathcal{S}$ and use one per attribute
- We can do better: ORAM state (the largest part) is shared, index and $\mathcal{DS}$ per attribute
- Need to split privacy budget ϵ among attributes (Composition Theorem)

Storage efficiency is defined as the sum of the bit-lengths of the records in a database relative to the bit-length of a corresponding encrypted database. **Communication efficiency** is defined as the sum of the lengths of the records in bits whose search keys satisfy the query relative to the actual number of bits sent back as the result of a query. Efficiency is defined as (a_1, a_2), where a_1 is a multiplicative term and a_2 is the additive one.

$\mathcal{E}$psolute's storage efficiency is $(\mathcal{O}(1), 0)$. Communication efficiencies for different query and DP types are shown.

	Point Query	Range Query
Pure DP	$(\mathcal{O}(\log n), \mathcal{O}(\log N \log n))$	$(\mathcal{O}(\log n), \mathcal{O}(\log N \log n))$
Approximate DP	$(\mathcal{O}(\log n), \mathcal{O}(\log n))$	$(\mathcal{O}(\log n), \mathcal{O}(2^{\log^* N} \log n))$

Multiple indexed attributes

- Naïve way is to duplicate the entire stack of states of $\mathcal{U}$ and $\mathcal{S}$ and use one per attribute
- We can do better: ORAM state (the largest part) is shared, index and $\mathcal{DS}$ per attribute
- Need to split privacy budget ϵ among attributes (Composition Theorem)

Parallel $\mathcal{E}$psolute: the choice of separate vs shared $\mathcal{DS}$

- Split $\mathcal{U}$ and $\mathcal{S}$ state into m ORAMs, run as separate machines
- Partition records randomly (by ID) into m partitions, generate m inverted indexes

No-γ method: $\mathcal{DS}$ per ORAM

- Composition of disjoint datasets: take max ϵ
- Each ORAM replies with $(1+\gamma)\frac{k_0}{m}$ records, where k_0 is a required number of records
- To bound probability to β, use $\gamma = \sqrt{\frac{-3m \log \beta}{k_0}}$

γ-method: shared $\mathcal{DS}$

- Same number of records per ORAM
- Use γ as in no-γ method, except
  - $k_0 \leftarrow k_0 + \frac{\log N}{\epsilon}$ for point queries
  - $k_0 \leftarrow k_0 + \frac{\log^{1.5} N}{\epsilon}$ for range queries

	Point Query	Range Query
no-γ-method	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0}}\right)\log\frac{n}{m}\right), \mathcal{O}\left(\frac{\log N}{\epsilon} m \log n\right)\right)$	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0}}\right)\log\frac{n}{m}\right), \mathcal{O}\left(\frac{\log^{1.5} N}{\epsilon} m \log n\right)\right)$
γ-method	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0+\frac{\log N}{\epsilon}}}\right)\log\frac{n}{m}\left(1+\frac{\log N}{\epsilon}\right)\right), 0\right)$	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0+\frac{\log^{1.5} N}{\epsilon}}}\right)\log\frac{n}{m}\left(1+\frac{\log^{1.5} N}{\epsilon}\right)\right), 0\right)$

Communication efficiencies for different query types and methods.

- Split $\mathcal{U}$ and $\mathcal{S}$ state into m ORAMs, run as separate machines
- Partition records randomly (by ID) into m partitions, generate m inverted indexes

No-γ method: $\mathcal{DS}$ per ORAM

- Composition of disjoint datasets: take max ϵ
- Each ORAM replies with $(1+\gamma)\frac{k_0}{m}$ records, where k_0 is a required number of records
- To bound probability to β, use $\gamma = \sqrt{\frac{-3m \log \beta}{k_0}}$

γ-method: shared $\mathcal{DS}$

- Same number of records per ORAM
- Use γ as in no-γ method, except
  - $k_0 \leftarrow k_0 + \frac{\log N}{\epsilon}$ for point queries
  - $k_0 \leftarrow k_0 + \frac{\log^{1.5} N}{\epsilon}$ for range queries

	Point Query	Range Query
no-γ-method	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0}}\right)\log\frac{n}{m}\right), \mathcal{O}\left(\frac{\log N}{\epsilon} m \log n\right)\right)$	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0}}\right)\log\frac{n}{m}\right), \mathcal{O}\left(\frac{\log^{1.5} N}{\epsilon} m \log n\right)\right)$
γ-method	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0+\frac{\log N}{\epsilon}}}\right)\log\frac{n}{m}\left(1+\frac{\log N}{\epsilon}\right)\right), 0\right)$	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0+\frac{\log^{1.5} N}{\epsilon}}}\right)\log\frac{n}{m}\left(1+\frac{\log^{1.5} N}{\epsilon}\right)\right), 0\right)$

Communication efficiencies for different query types and methods.

- Split $\mathcal{U}$ and $\mathcal{S}$ state into m ORAMs, run as separate machines
- Partition records randomly (by ID) into m partitions, generate m inverted indexes

No-γ method: $\mathcal{DS}$ per ORAM

- Composition of disjoint datasets: take max ϵ
- Each ORAM replies with $(1+\gamma)\frac{k_0}{m}$ records, where k_0 is a required number of records
- To bound probability to β, use $\gamma = \sqrt{\frac{-3m \log \beta}{k_0}}$

γ-method: shared $\mathcal{DS}$

- Same number of records per ORAM
- Use γ as in no-γ method, except
  - $k_0 \leftarrow k_0 + \frac{\log N}{\epsilon}$ for point queries
  - $k_0 \leftarrow k_0 + \frac{\log^{1.5} N}{\epsilon}$ for range queries

	Point Query	Range Query
no-γ-method	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0}}\right)\log\frac{n}{m}\right), \mathcal{O}\left(\frac{\log N}{\epsilon} m \log n\right)\right)$	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0}}\right)\log\frac{n}{m}\right), \mathcal{O}\left(\frac{\log^{1.5} N}{\epsilon} m \log n\right)\right)$
γ-method	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0+\frac{\log N}{\epsilon}}}\right)\log\frac{n}{m}\left(1+\frac{\log N}{\epsilon}\right)\right), 0\right)$	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m\log\beta}{k_0+\frac{\log^{1.5} N}{\epsilon}}}\right)\log\frac{n}{m}\left(1+\frac{\log^{1.5} N}{\epsilon}\right)\right), 0\right)$

Communication efficiencies for different query types and methods.

Parallel $\mathcal{E}$psolute: the choice of separate vs shared $\mathcal{DS}$

- Split $\mathcal{U}$ and $\mathcal{S}$ state into m ORAMs, run as separate machines
- Partition records randomly (by ID) into m partitions, generate m inverted indexes

No-γ method: $\mathcal{DS}$ per ORAM

- Composition of disjoint datasets: take max ϵ
- Each ORAM replies with $(1+\gamma)\frac{k_0}{m}$ records, where k_0 is a required number of records
- To bound probability to β, use $\gamma = \sqrt{\frac{-3m \log \beta}{k_0}}$

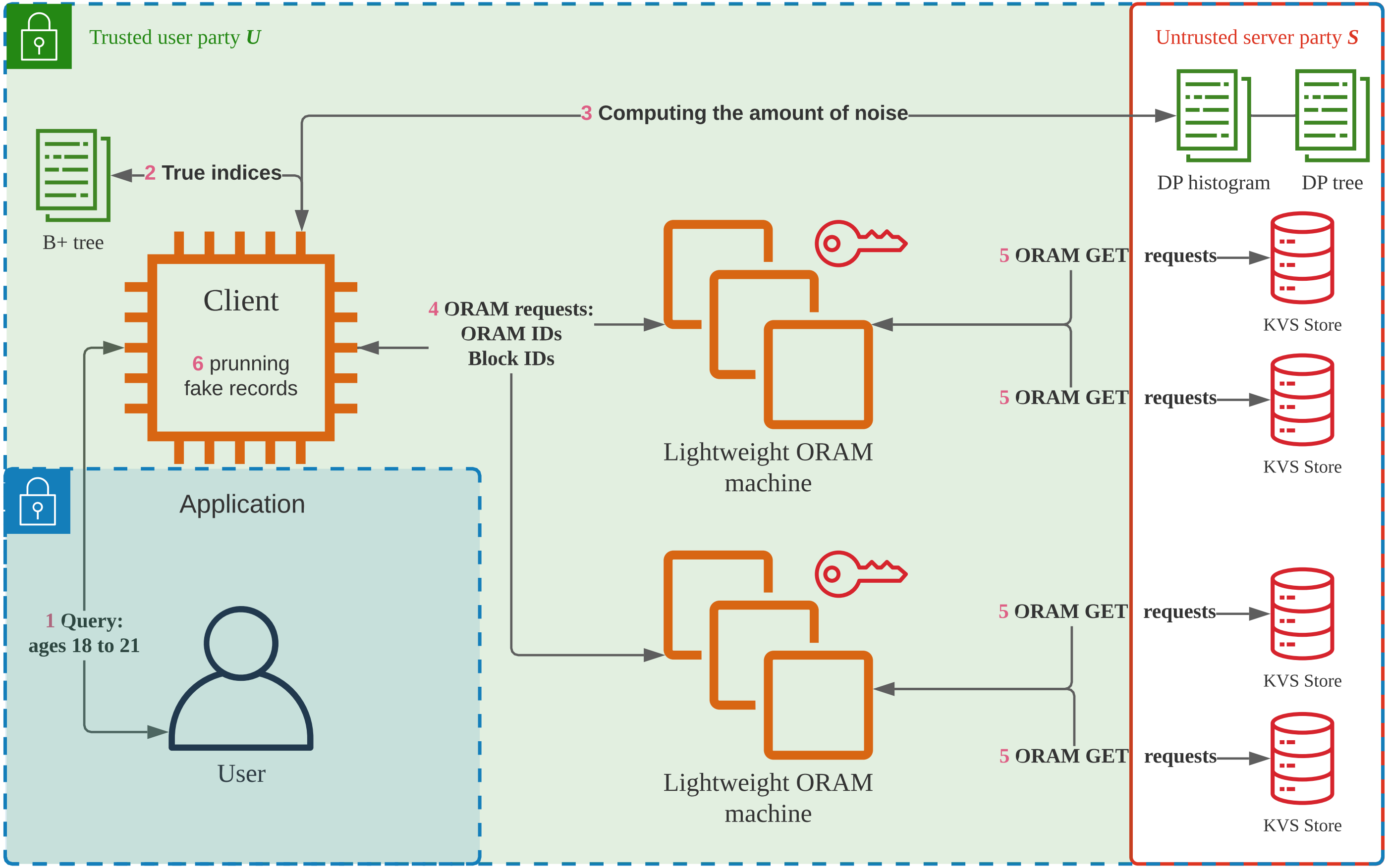
γ-method: shared $\mathcal{DS}$

- Same number of records per ORAM
- Use γ as in no-γ method, except
  - $k_0 \leftarrow k_0 + \frac{\log N}{\epsilon}$ for point queries
  - $k_0 \leftarrow k_0 + \frac{\log^{1.5} N}{\epsilon}$ for range queries

	Point Query	Range Query
no-γ-method	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m \log \beta}{k_0}}\right) \log \frac{n}{m}\right), \mathcal{O}\left(\frac{\log N}{\epsilon} m \log n\right)\right)$	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m \log \beta}{k_0}}\right) \log \frac{n}{m}\right), \mathcal{O}\left(\frac{\log^{1.5} N}{\epsilon} m \log n\right)\right)$
γ-method	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m \log \beta}{k_0 + \frac{\log N}{\epsilon}}}\right) \log \frac{n}{m} \left(1+\frac{\log N}{\epsilon}\right)\right), 0\right)$	$\left(\mathcal{O}\left(\left(1+\sqrt{\frac{-3m \log \beta}{k_0 + \frac{\log^{1.5} N}{\epsilon}}}\right) \log \frac{n}{m} \left(1+\frac{\log^{1.5} N}{\epsilon}\right)\right), 0\right)$

Communication efficiencies for different query types and methods.

Parallel $\mathcal{E}$psolute diagram (with improvements)

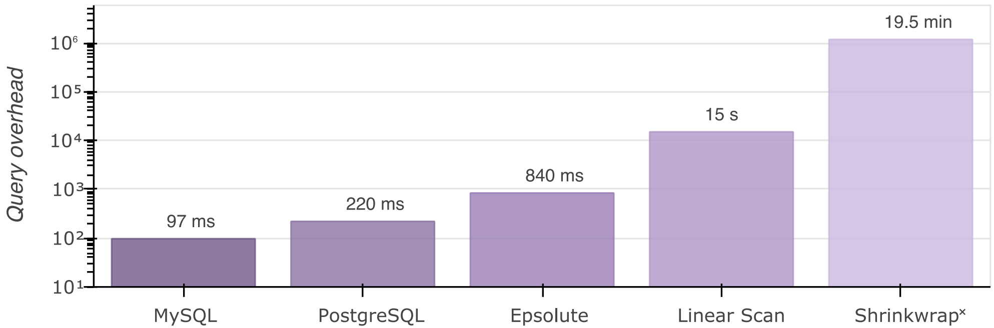


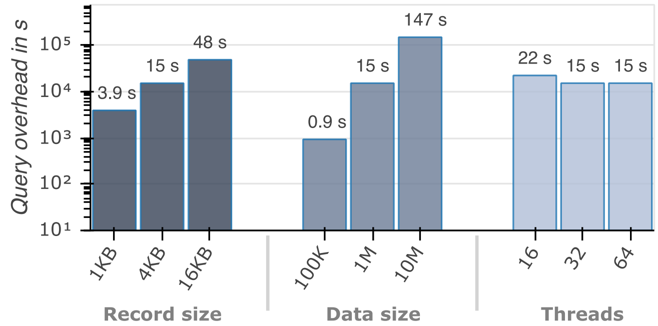
Experiments

Experiments setup

- Implemented in C++, PathORAM and B+ tree are modules, OpenSSL for crypto
- Run on GCP in different regions, 8 storage VMs, 8 ORAM VMs, one client
- Two real datasets (salaries) and one synthetic (uniform) dataset of sizes 100K, 1M and 10M
- Default setting
  - $\epsilon = \ln 2$ and $\beta = 2^{-20}$
  - 1M uniformly sampled 4 KiB records
  - selectivity 0.5 %
  - γ-method
- Mechanisms (besides $\mathcal{E}$psolute)
  - MySQL and PostgreSQL
  - Linear Scan (download everything every query)
  - Shrinkwrap [28] (adapted range queries from their source code)

How practical is our system compared to the most efficient and most private real-world solutions?





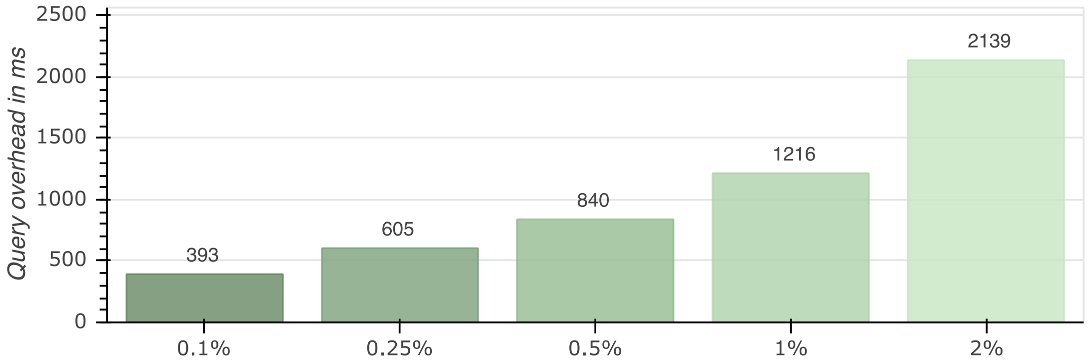
Three orders of magnitude faster than Shrinkwrap [28], 18 times faster than the linear scan and only 4–8 times slower than a conventional database.

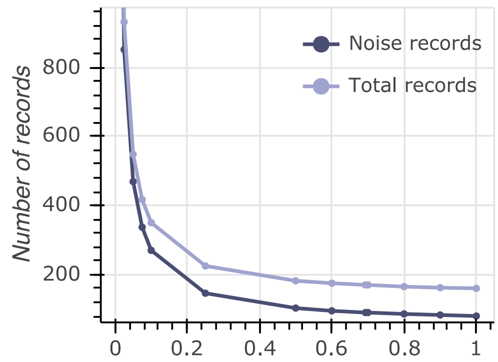
How practical is the storage overhead?

n \ Record	1 KiB		4 KiB		16 KiB	
10^5	400 KiB	400 B	400 KiB	102 KiB	400 KiB	1.6 MB
	396 MB	4.6 MB	**1.5 GB**	14 MB	**6.2 GB**	51 MB
10^6	3.9 MB	400 B	3.9 MB	102 KiB	3.9 MB	1.6 MB
	3.2 GB	15 MB	**12 GB**	25 MB	**48 GB**	62 MB
10^7	40 MB	400 B	40 MB	102 KiB	*40 MB*	*1.6 MB*
	24 GB	99 MB	**96 GB**	109 MB	***384 GB***	*146 MB*
n / N	100		10^4		10^6	

Left top: index $\mathcal{I}$ (B+ tree), right top: aggregate tree $\mathcal{DS}$, right bottom: ORAM $\mathcal{U}$ state and left bottom (bold): ORAM $\mathcal{S}$ state. *Italic* values are estimated. $\mathcal{S}$ to $\mathcal{U}$ storage size ratio is 85, 414 and over 2 000.

How different inputs and parameters of the system affect its performance?

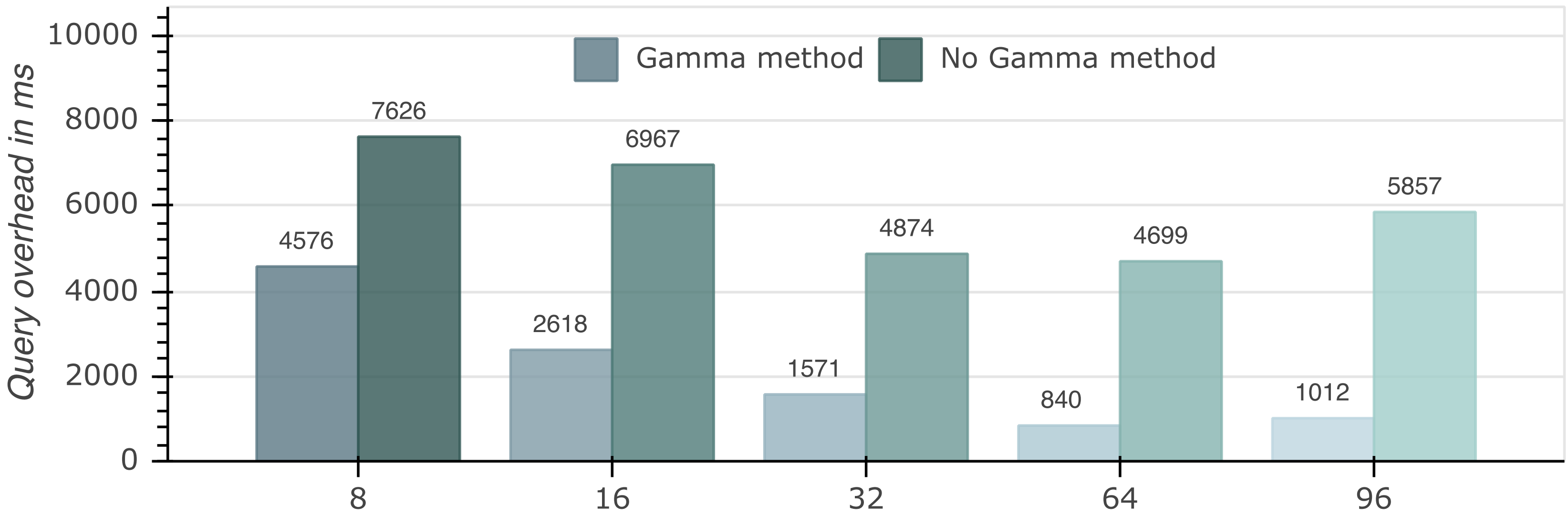




ϵ strictly contributes to the amount of noise, which grows exponentially as ϵ decreases.

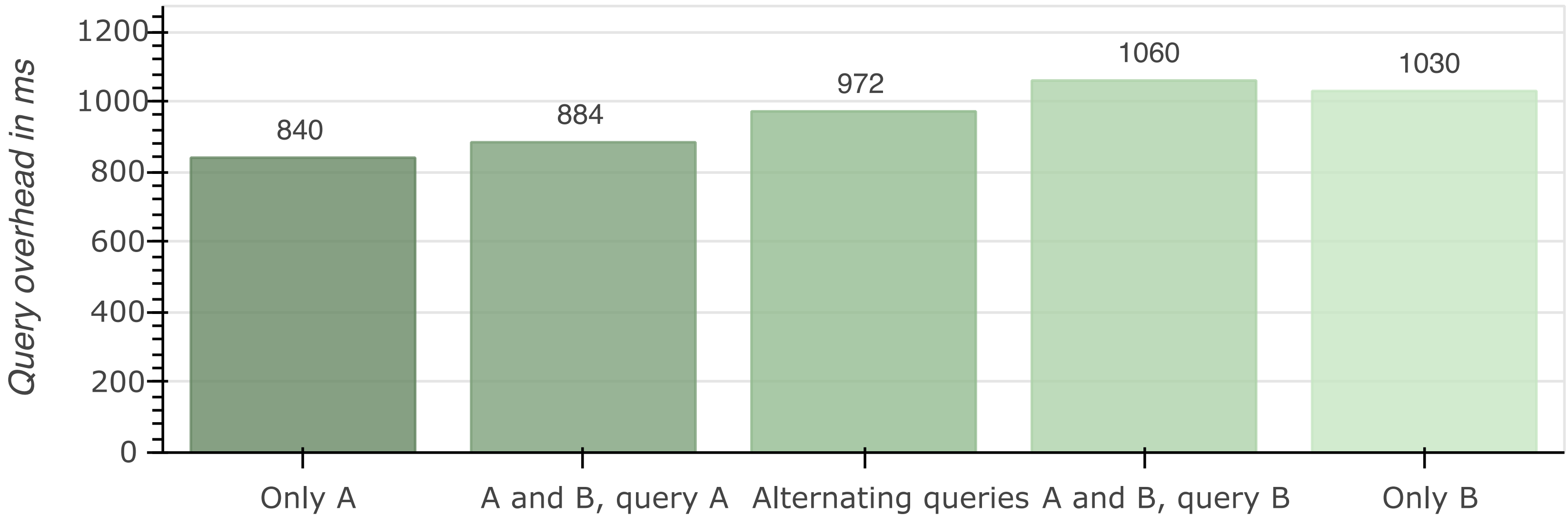
Overhead expectedly grows with the result size.

How well does the system scale?



The γ-method provides substantially better performance and storage efficiency, and when using this method the system scales **linearly** with the number of ORAMs.

What is the impact of supporting multiple attributes?



The overhead increases only slightly due to a lower privacy budget.

$\mathcal{E}$psolute: Efficiently Querying Databases While Providing Differential Privacy

Differential Privacy, ORAM, differential obliviousness, sanitizers

[42] DOI: 10.1145/3460120.3484786

Dmytro Bogatov, Georgios Kellaris, George Kollios, Kobbi Nissim, Adam O'Neill
dmytro@bu.edu, kellaris@bu.edu, gkollios@cs.bu.edu,
kobbi.nissim@georgetown.edu, adamo@cs.umass.edu

Built from *078878e4* on October 17, 2021

Boston University
Graduate School of Arts and Sciences
Department of Computer Science

References

[1] Oded Goldreich. "Towards a theory of software protection and simulation by oblivious RAMs". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. 1987, pp. 182–194. DOI: 10.1145/28395.28416.

[2] Oded Goldreich and Rafail Ostrovsky. "Software protection and simulation on oblivious RAMs". In: *Journal of the ACM (JACM)* 43.3 (1996), pp. 431–473. DOI: 10.1145/233551.233553.

[3] Cynthia Dwork *et al.* "Calibrating noise to sensitivity in private data analysis". In: *Theory of cryptography conference*. Springer. 2006, pp. 265–284. DOI: 10.1007/11681878_14.

[4] Cynthia Dwork *et al.* "Our data, ourselves: Privacy via distributed noise generation". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2006, pp. 486–503. DOI: 10.1007/11761679_29.

[5] Cynthia Dwork *et al.* “Differential privacy under continual observation”. In: *Proceedings of the forty-second ACM symposium on Theory of computing*. 2010, pp. 715–724. DOI: 10.1145/1806689.1806787.

[6] Raluca Ada Popa *et al.* “CryptDB: Protecting confidentiality with encrypted query processing”. In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. 2011, pp. 85–100.

[7] Elaine Shi *et al.* “Oblivious RAM with $O(\log^3 N)$ worst-case cost”. In: *International Conference on The Theory and Application of Cryptology and Information Security*. Springer. 2011, pp. 197–214. DOI: 10.1007/978-3-642-25385-0_11.

[8] Bijit Hore *et al.* “Secure multidimensional range queries over outsourced data”. In: *VLDBJ* 21.3 (2012), pp. 333–358. DOI: 10.1007/s00778-011-0245-7.

[9] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. "Access Pattern disclosure on Searchable Encryption: Ramification, Attack and Mitigation". In: *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012.

[10] Emil Stefanov, Elaine Shi, and Dawn Xiaodong Song. "Towards Practical Oblivious RAM". In: *Network and Distributed System Security Symposium (NDSS)*. 2012.

[11] Arvind Arasu *et al.* "Orthogonal Security With Cipherbase". In: *6th Biennial Conference on Innovative Data Systems Research (CIDR'13)*. 2013.

[12] Amos Beimel, Kobbi Nissim, and Uri Stemmer. "Private learning and sanitization: Pure vs. approximate differential privacy". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 2013, pp. 363–378. DOI: `10.1007/978-3-642-40328-6_26`.

[13] Avrim Blum, Katrina Ligett, and Aaron Roth. “A learning theory approach to non-interactive database privacy”. In: *Journal of the ACM (JACM)* 60.2 (2013), pp. 1–25. DOI: 10.1145/1374376.1374464.

[14] Emil Stefanov *et al.* “Path ORAM: an extremely simple oblivious RAM protocol”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.* 2013, pp. 299–310. DOI: 10.1145/3177872.

[15] Amos Beimel *et al.* “Bounds on the sample complexity for private learning and private data release”. In: *Machine learning* 94.3 (2014), pp. 401–437. DOI: 10.1007/s10994-013-5404-1.

[16] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. “Inference attack against encrypted range queries on outsourced databases”. In: *Proceedings of the 4th ACM conference on Data and application security and privacy.* 2014, pp. 235–246. DOI: 10.1145/2557547.2557561.

[17] Mark Bun *et al.* “Differentially private release and learning of threshold functions”. In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 634–649. DOI: 10.1109/FOCS.2015.45.

[18] David Cash *et al.* “Leakage-abuse attacks against searchable encryption”. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 668–679. DOI: 10.1145/2810103.2813700.

[19] Muhammad Naveed, Seny Kamara, and Charles V Wright. “Inference attacks on property-preserving encrypted databases”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 644–655. DOI: 10.1145/2810103.2813651.

[20] Yuval Ishai *et al.* “Private large-scale databases with distributed searchable symmetric encryption”. In: *Cryptographers’ Track at the RSA Conference*. Springer. 2016, pp. 90–107. DOI: 10.1007/978-3-319-29485-8_6.

[21] Georgios Kellaris *et al.* "Generic attacks on secure outsourced databases". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 1329–1340. DOI: 10.1145/2976749.2978386.

[22] Antonis Papadimitriou *et al.* "Big Data Analytics over Encrypted Datasets with Seabed". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, 2016, pp. 587–602.

[23] Cetin Sahin *et al.* "Taostore: Overcoming asynchronicity in oblivious data storage". In: *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2016, pp. 198–217. DOI: 10.1109/SP.2016.20.

[24] Dong Xie *et al.* "Practical private shortest path computation based on oblivious storage". In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE. 2016, pp. 361–372. DOI: 10.1109/ICDE.2016.7498254.

[25] Benny Fuhry *et al.* "HardIDX: Practical and secure index with SGX". In: *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer. 2017, pp. 386–408. DOI: 10.1007/978-3-319-61176-1_22.

[26] Paul Grubbs, Thomas Ristenpart, and Vitaly Shmatikov. "Why Your Encrypted Database Is Not Secure". In: *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. ACM, 2017, pp. 162–168. DOI: 10.1145/3102980.3103007.

[27] Wenting Zheng *et al.* "Opaque: An oblivious and encrypted distributed analytics platform". In: *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)*. 2017, pp. 283–298.

[28] Johes Bater *et al.* "Shrinkwrap: efficient sql query processing in differentially private data federations". In: *Proceedings of the VLDB Endowment* 12.3 (2018), pp. 307–320. DOI: 10.14778/3291264.3291274.

[29] Vincent Bindschaedler *et al.* "The Tao of Inference in Privacy-protected Databases". In: *PVLDB* 11.11 (2018), pp. 1715–1728. DOI: 10.14778/3236187.3236217.

[30] Paul Grubbs *et al.* “Pump up the volume: Practical database reconstruction from volume leakage on range queries”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 315–331. DOI: 10.1145/3243734.3243864.

[31] Marie-Sarah Lacharité, Brice Minaud, and Kenneth G Paterson. “Improved reconstruction attacks on encrypted data using range query leakage”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 297–314. DOI: 10.1109/SP.2018.00002.

[32] Pratyush Mishra *et al.* “Oblix: An efficient oblivious search index”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 279–296. DOI: 10.1109/SP.2018.00045.

[33] Christian Priebe, Kapil Vaswani, and Manuel Costa. “EnclaveDB: A secure database using SGX”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 264–278. DOI: 10.1109/SP.2018.00025.

[34] Cetin Sahin *et al.* “A Differentially Private Index for Range Query Processing in Clouds”. In: *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. 2018, pp. 857–868. DOI: 10.1109/ICDE.2018.00082.

[35] Saba Eskandarian and Matei Zaharia. "ObliDB: Oblivious query processing for secure databases". In: *PVLDB* 13.2 (2019), pp. 169–183. DOI: 10.14778/3364324.3364331.

[36] Zichen Gui, Oliver Johnson, and Bogdan Warinschi. "Encrypted databases: New volume attacks against range queries". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 361–378. DOI: 10.1145/3319535.3363210.

[37] Rishabh Poddar, Tobias Boelter, and Raluca Ada Popa. "Arx: an encrypted database using semantically secure encryption". In: *Proceedings of the VLDB Endowment* 12.11 (2019), pp. 1664–1678. DOI: 10.14778/3342263.3342641.

[38] Dhinakaran Vinayagamurthy, Alexey Gribov, and Sergey Gorbunov. "StealthDB: a scalable encrypted database with full SQL query support". In: *Proceedings on Privacy Enhancing Technologies* 2019.3 (2019), pp. 370–388. DOI: 10.2478/popets-2019-0052.

[39] Ioannis Demertzis *et al.* "SEAL: Attack Mitigation for Encrypted Databases via Adjustable Leakage". In: *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, 2020, pp. 2433–2450. ISBN: 978-1-939133-17-5. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/demertzis.

[40] Evgenios M Kornaropoulos, Charalampos Papamanthou, and Roberto Tamassia. "The state of the uniform: attacks on encrypted databases beyond the uniform query distribution". In: *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2020, pp. 1223–1240. DOI: 10.1109/SP40000.2020.00029.

[41] Amrita Roy Chowdhury *et al.* "Cryptϵ: Crypto-assisted differential privacy on untrusted servers". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 2020, pp. 603–619. DOI: 10.1145/3318464.3380596.

[42] Dmytro Bogatov *et al.* "$\mathcal{E}$psolute: Efficiently Querying Databases While Providing Differential Privacy". In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '2021)*. 2021. DOI: 10.1145/3460120.3484786.