BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation Prospectus

SECURE AND EFFICIENT QUERY PROCESSING IN OUTSOURCED DATABASES

by

DMYTRO BOGATOV

B.S., Worcester Polytechinc Institute, 2017
M.S., Boston University, 2019

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2022

SECURE AND EFFICIENT QUERY PROCESSING IN OUTSOURCED DATABASES

DMYTRO BOGATOV

Boston University, Graduate School of Arts and Sciences, 2022

Major Professor: George Kollios, PhD
Professor of Computer Science

ABSTRACT

As organizations struggle with processing vast amounts of information, outsourcing sensitive data to third parties becomes a necessity. To protect the data, various cryptographic techniques are used in outsourced database systems to ensure data privacy while allowing efficient querying.

In this prospectus, I propose a definition and components of a new secure and efficient outsourced database system, which answers various types of queries, with different privacy guarantees in different security models. I start with my survey work on five order-preserving and order-revealing encryption schemes that can be used directly in many database indices, such as the B+ tree, and five range query protocols with various trade-offs in terms of security and efficiency. This work systematizes the state-of-the-art range query solutions in a snapshot adversary setting and offers some non-obvious observations regarding the efficiency of the constructions. I then follow with a recently published work, $\mathcal{E}$psolute — an efficient range query engine in a persistent adversary model. In this work, we achieve security in a setting with a much stronger adversary where she can continuously observe everything on the server, and leaking even the result size can enable a reconstruction attack. We propose a

definition, construction, analysis and experimental evaluation of a system that provably hides both access pattern and communication volume, while remaining efficient. Finally, I describe an ongoing work on secure k-nearest-neighbor queries, in which the security is achieved similarly to OPE/ORE solutions — encrypting the input with an approximate Distance Comparison Preserving Encryption scheme, so that the inputs are perturbed, but the query algorithm still produces accurate results.

Contents

List of Tables

List of Figures

List of Acronyms

kNN	.....	k-nearest-neighbor
CDP-ODB	.....	Computationally DP Outsourced Database System
DCG	.....	Discounted Cumulative Gain
DCPE	.....	Distance Comparison Preserving Encryption
DP	.....	Differential Privacy
FAISS	.....	Facebook AI Similarity Search
GPU	.....	Graphics Processing Unit
HG	.....	hypergeometric probability distribution
I/O	.....	Input / Output
LPA	.....	Laplacian Perturbation Algorithm
MRR	.....	Mean Reciprocal Rank
OPE	.....	Order-Preserving Encryption
ORAM	.....	Oblivious Random Access Machine
ORE	.....	Order-Revealing Encryption
PPH	.....	Property-Preserving Hash
PRF	.....	Pseudo-Random Function
PRP	.....	Pseudo-Random Permutation
RDBMS	.....	Relational Database Management System
SQL	.....	Structured Query Language
SSE	.....	Searchable Symmetric Encryption
TREC	.....	Text Retrieval Conference
VM	.....	Virtual Machine

Chapter 1

Proposal

1.1 Introduction and Background

As the organizations struggle with demands for storage and processing of their data, they increasingly turn to third parties for outsourcing capabilities. A number of companies including Amazon (AWS), Microsoft (Azure) and Google (GCP) offer outsourced database solutions to individuals and other businesses. This model is lucrative because not only do the clients pay exactly for what they use in terms of pure computational resources, but also the cloud takes care of the entire deployment process, including availability, scalability, replication, and, most importantly, security. The cloud business model provides resources on-demand — from bare-bones VM to database-as-a-service products.

While the cloud providers typically have strict customer data privacy policies and even offer server-side encryption-at-rest services, the clients still have to trust the provider with their plaintext data. Server-side encryption-at-rest by definition requires the provider to know the encryption key to manipulate the data, even if the key is ephemeral and is not stored in the cloud permanently. Moreover, cloud providers in general, and customers' VMs in particular, may be vulnerable to external attacks — from snapshot-level attacks, in which the adversary has a copy of the VM memory, to more devastating persistent attacks, in which the adversary continuously monitors the VM processes.

Protecting the private information beyond cloud provider guarantees typically requires encrypting it in a way that preserves the ability to process it. A line of research targets securing outsourced database systems, but often achieves protection at the cost of efficiency too high for a solution to be viable for practical applications. In this prospectus, I will cover the constructions that are used to answer different types of database queries in the outsourced model while providing both provable security and practical efficiency guarantees.

1.1.1 Model

In this work, I consider *an outsourced database system* model adapted from [KKNO16] and [Bog+21].

Outsourced database model

Similar to [Bog+21], a database is abstracted as a collection of n records r, each with a unique identifier r^{ID}, associated with search keys SK: $\mathcal{D} = \{(r_1, r_1^{\mathsf{ID}}, \mathsf{SK}_1), \ldots, (r_n, r_n^{\mathsf{ID}}, \mathsf{SK}_n)\}$. All records are assumed to have an identical fixed bit-length, and the search keys are elements of some domain $\mathcal{X}$. A query is modeled as a predicate $q \in \mathcal{Q} : \mathcal{X} \to \{0, 1\}$. Evaluating a query q on a database $\mathcal{D}$ results in $q(\mathcal{D}) = \{r_i : q(\mathsf{SK}_i) = 1\}$, all records whose search keys satisfy q.

Formally, an outsourced database system consists of two protocols between a stateful user $\mathcal{U}$, who owns the data, and an untrusted stateful server $\mathcal{S}$, to whom these data are outsourced. In setup protocol Π_{setup}, $\mathcal{U}$ receives as input a database $\mathcal{D} = \{(r_1, r_1^{\mathsf{ID}}, \mathsf{SK}_1), \ldots, (r_n, r_n^{\mathsf{ID}}, \mathsf{SK}_n)\}$ and $\mathcal{S}$ may optionally output a data structure $\mathcal{DS}$. In query protocol Π_{query}, $\mathcal{U}$ receives a query $q \in \mathcal{Q}$, $\mathcal{S}$ receives $\mathcal{DS}$ produced in the setup protocol, and $\mathcal{U}$ outputs the result of the query $q(\mathcal{D})$. Both parties may update their internal states. We call a system *correct* if it holds with overwhelming

probability over the randomness of the above runs that running Π_{setup} and Π_{query} on the corresponding inputs yields the correct result $\{r_i : q(\mathsf{SK}_i) = 1\}$.

Security model

I define two types of adversaries — a *snapshot* and a *persistent* adversaries.

As the name suggests, a snapshot adversary can see a "snapshot" of the server's data at multiple points in time. One can think of such an attack as if someone steals a hard drive or accesses a backup. Formally, $\mathcal{A}$ knows $\mathcal{S}$ state at all stages of the protocol.

A persistent adversary is stronger in that she monitors the server continuously. Therefore, she can see what snapshot adversary can plus the network traffic and the access pattern. Such adversary can be thought of as malicious software (virus) that runs as a background process with broad permissions. Formally, on top of the $\mathcal{S}$ state, $\mathcal{A}$ knows the size and content of $\mathcal{S}$ communication and the sequence of accesses $\mathcal{S}$ makes to its internal state at all protocol stages.

Intuitively, one can think that encrypting records should protect the data. Depending on how the records are encrypted (i.e., whether the symmetric or property-preserving encryption is used), this approach can mitigate the snapshot adversary. Persistent adversary, however, can observe the communication size even if the traffic itself is encrypted, and can see the access pattern even if the records are protected. It has been shown that the knowledge of access pattern [HMCK12; IKK14; CGPR15; NKW15; KKNO16; Bin+18; GRS17; IKK12; LMP18] or communication volume [KKNO16; KPT20; LMP18; GLMP18; GJW19] alone can enable a series of reconstruction attacks. Also note that both adversaries are *honest-but-curios* — they only observe and never interfere. Denial-of-service attacks and integrity protection are out of scope of this work.

Query types

The type of query q is deliberately left abstract. The outsourced database system assumes that a query contains a way (a predicate) to select only the records whose search keys satisfy it. In this work, I consider the following types of queries.

A point query. This query selects records whose key is equal to a given value. The domain of the point value does not have to be ordered; it can be categorical, like color names. The relevant SQL query can be

```
SELECT * FROM t1 WHERE zip = '02215'.
```

A range query. This query selects records whose key lies between two values from an ordered domain. The relevant SQL query can be

```
SELECT * FROM t1 WHERE age BETWEEN 18 and 65.
```

A kNN query. k-nearest-neighbor query selects k records whose keys are "closest" to a given value. This query type requires a definition of distance metric over the domain of search keys, for example, simple Euclidean distance. The relevant SQL query can be

```
SELECT * FROM t1 ORDER BY location <-> '(29.9691,-95.6972)' LIMIT 5.
```

1.1.2 Proposal

I propose to structure the thesis around the problem of answering the aforementioned three query types securely in the presence of snapshot or persistent adversaries. The thesis will include a rich background on proposed secure query processing solutions and attacks against these solutions. I will explain the components needed to construct the solutions, including symmetric encryption, property-preserving encryption, Oblivious Random Access Machine, and Differential Privacy.

1.2 Range queries in a snapshot model

1.2.1 Problem

There are plenty of works offering range query solutions in a snapshot adversary model. Some authors propose a new Order-Revealing Encryption (ORE) scheme to be used inside an existing range query index, see [BCLO09; CLWW16a; LW16; Cas+18; Ker15]. Others suggest complete client-server protocols with different security guarantees, see [KT19; RACY16; CLWW16b].

Instead of adding another solution to the mix, we developed a robust evaluation framework to compare and audit different ORE schemes and range query protocols [BKR19]. We have noticed that all proposed solutions offer their own formal protocol, security definition (which they obliviously satisfy), and even performance metrics. Moreover, only a few of these algorithms have ready-to-use open-sourced code. More often, the implementation is a prototype (or does not even exist), rendering experiments non-reproducible.

1.2.2 Solution

Our work addresses these shortcomings. We proposed a common framework where all analyzed solutions follow the same formal protocol, a variant of an outsourced database from Section 1.1.1. The schemes and protocols are also tested against the same security definition, a variant of a simulation-based security game where the simulator has a leakage function that is different for each solution. Lastly, all protocols are implemented in the same language and runtime, and are using the same cryptographic primitive implementations.

We have paid extra attention to how we quantify the performance. Published works typically claim that they have implemented the algorithm in some language and ran it on a server many times measuring the wall-clock time. Such metric is deeply

flawed since it only measures just that — the wall-clock time it took for a particular machine to execute a particular code that references particular implementations of primitives and makes (or even does not make) I/O requests to particular hardware. What we offer is a metric that drops all these "particulars" (i.e., dependencies) from the protocol execution. On top of having all solutions implemented in the same language and using the same primitives, instead of measuring the time we count the number of times an ORE scheme makes use of a primitive and a protocol makes an I/O request. This metric is much more demonstrative: one can always get a wall-clock time estimate given these counts and the specs of the hardware. Moreover, we can analyze these counts theoretically from the pseudocode and then verify them practically after running the code.

1.2.3 Results

In our work, we apply the framework to ten range query solutions. Five protocols are built with an Order-Revealing Encryption (ORE) scheme coupled with a B+ tree. In Π_{setup}, $\mathcal{C}$ encrypts all search keys with the ORE scheme, constructs a B+ tree from the ciphertexts and uploads it to $\mathcal{S}$. In Π_{query}, $\mathcal{C}$ sends encrypted query endpoints and $\mathcal{S}$ follows standard B+ traversal. Such protocol leaks the total order of the records plus the leakage of the underlying ORE scheme.

We also analyzed dedicated range query protocols from [KT19; RACY16; Dem+16] and two more baselines. Baselines are chosen at the extremes of performance / security spectrum — a plaintext B+ tree and a B+ tree in ORAM. Intuitively, all solutions should lie between the baselines in both performance and security, but we show that at least performance-wise it is not the case (see one of the plots in Figure 1·1).

Please see Tables A.1 and A.2 in Appendix A for the analytical results for each solution, and [BKR19] for experimental results.

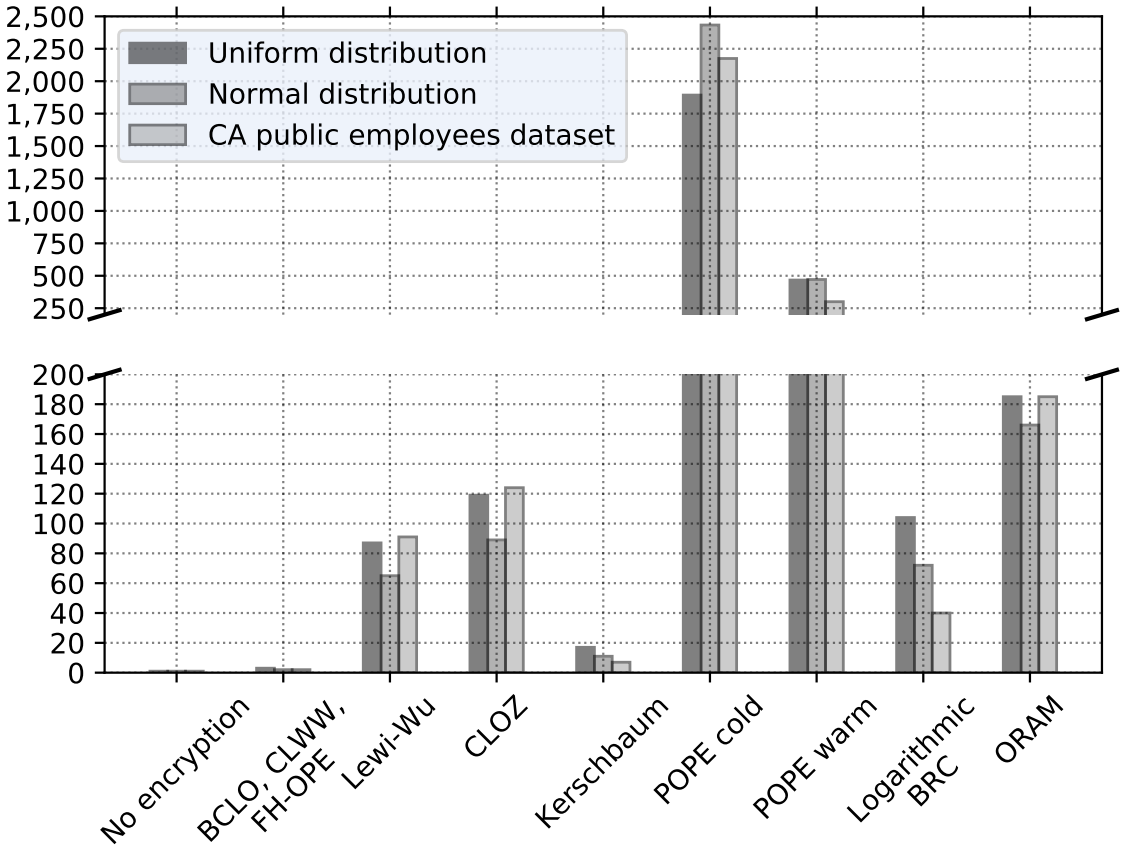

Figure 1·1: Query stage number of I/O requests [BKR19, Figure 2d]

1.3 Range and point queries in a persistent model

1.3.1 Motivation and Background

A persistent adversary is capable of observing not only the data at any point in time (like the snapshot attacker), but also all processes and communication on the server, including the network messages, their size, and the access pattern to the storage. A series of attacks starting with a seminal work of Kellaris et al. [KKNO16] showed that the query result size [KKNO16; KPT20; LMP18; GLMP18; GJW19] and access pattern [HMCK12; IKK14; CGPR15; NKW15; KKNO16; Bin+18; GRS17; IKK12; LMP18] alone could enable reconstruction attacks. The attacks against the access pattern typically exploit the fact that some records are accessed more often than others and in the same query as others. The adversary maps this knowledge to some public auxiliary dataset and queryset, and guesses the values correctly with a non-negligible probability. Attacks on communication volume are more elaborate and use the fact that there are a well-defined number of distinct interval queries over the domain. Kellaris et al. [KKNO16], for example, construct a polynomial with the observed number of queries that return a different number of records, and solve it deriving the exact reconstruction.

A generic way to protect the access pattern is the Oblivious Random Access Machine (ORAM), an interactive client-server protocol that conceals the access pattern from the observing adversary by making at least $\mathcal{O}(\log n)$ extra requests. A naïve solution is to simply proxy all requests through the ORAM protocol, but it is generally inefficient and not trivially parallelizable.

Protecting against communication volume leakage usually involves returning a number of extra fake records and letting the client filter them out locally. Where approaches differ is in the number of these fake records. A naïve way of adding a constant amount of noise or even sampling noise uniformly is insecure since such

noise can be filtered out by the adversary either immediately (constant noise) or after observing a certain number of queries (uniform noise). Another dimension to the problem is how much noise to add to have a provable guarantee and yet add the smallest amount of noise possible.

Differential Privacy

The most effective approach is sampling noise using Differential Privacy (DP). DP is a guarantee on a query algorithm that takes a database and returns some result. The guarantee states that for two neighboring databases (that differ in exactly one record), the probability that the adversary will understand by looking at the output, which of the two databases was used as an input, is bounded. More formally, the Differential Privacy is defined in Definition 1.

Definition 1 (Differential Privacy, adapted from [Dwo+06; DMNS06])**.** *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*,*

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

One way to interpret this definition is the following. Probabilities are taken over the coins of algorithm A, which answers a query based on a dataset. A natural instantiation of A is a view of a distinguishing adversary $\mathcal{A}$, who tries to guess which of the two datasets was used. The expression in Definition 1 then bounds the advantage of $\mathcal{A}$ with ϵ and δ parameters. Note that $\exp(x) \approx 1 + x + \frac{x^2}{2!}$ and for sufficiently small x the last term is negligible. If we put $\epsilon + 1$ in place of $\exp(\epsilon)$, it becomes clear that ϵ is the exact value by which two probabilities are allowed to differ. For $\epsilon = 0$, they have to be equal, for $\epsilon = 0.01$, probabilities may differ by 1 %. Therefore, ϵ is called *a privacy budget* of a DP system. δ term is additive and

therefore must be small by itself. This term is essentially a probability that the entire system fails. For example, if A is a randomized algorithm that fails with a certain chance, this probability will be δ. For instance, a PathORAM [Ste+13] algorithm can have a stash overflow with a bounded probability [Ste+13, Theorem 1] and it will cause the entire algorithm to fail. If PathORAM is used in a DP system then this probability, however small, bounded and negligible, will have to be accounted for in δ.

Note that Definition 1 describes a property of A and not a construction method. To construct A, the seminal [DMNS06] offers an algorithm called Laplacian Perturbation Algorithm (LPA). The idea is to tune the noise sampled from the Laplacian distribution to the *sensitivity* of a query, defined as the change of output with respect to change in input. For example, if a change in one record of the dataset causes a change in the output value of at most one (e.g., a count query), then the sensitivity is 1. [DMNS06] proves that if one adds $\textsc{Laplace}\left(0, \frac{\mathsf{sensitivity}}{\epsilon}\right)$ to the real result of a query, the resulting mechanism is ϵ-DP.

Lastly, instead of generating and adding noise each query, it is possible to generate noisy values once, embed them into the data and release the sanitized data without compromising privacy guarantees. The intuition is that once the noise is embedded in the dataset, the adversary can run a regular query over it and the entire query will still be DP. Such methods are collectively called *sanitization*, and there are known bounds for point and range queries, for pure ($\delta = 0$) and approximate ($\delta > 0$) DP [BBKN14; BNS13; BLR13; DNPR10; BNSV15; Kap+20].

1.3.2 Security definition

In this work we aim for the most practical definition of security, which needs to capture both access pattern and communication volume protection and yet be feasible enough

so that an efficient system exists that can satisfy it. In $\mathcal{E}$psolute [Bog+21] we propose a definition of computationally DP outsourced database.

Definition 2 (Computationally DP Outsourced Database System (CDP-ODB) from Definition 3.1 in [Bog+21])**.** *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 pair of neighboring databases* $\mathcal{D} \sim \mathcal{D}'$*, and for every query sequence* $q_1, \ldots, q_m \in \mathcal{Q}^m$ *where* $m = \mathsf{poly}(\lambda)$*,*

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

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

The first thing to note in this definition is that unlike in prior works where only a part of $\mathcal{A}$'s view is DP-protected (e.g., only access pattern in [CCMS19; BNZ19]), here the entire view of the adversary is protected. That is, if *anything* visible during the execution of the protocols to a persistent adversary can be used to distinguish the neighboring datasets, the adversary wins and the system fails to satisfy the definition. Subsequently, it implies protection against both access pattern and communication volume, since they are both a part of the view and thus if the adversary sees either of them, she can distinguish.

Also note that the CDP-ODB system is DP, but not necessarily semantically secure. A semantically secure database would have had probabilities almost equal (or, in Definition 2, $\epsilon = 0$). The trade-off has been made to allow construction of an efficient system. While a naïve solution that satisfies the semantically secure definition

would return the entire dataset every query, constructing a practical instantiation is hard, is not necessarily possible, and remains an open problem.

The negligible term $\mathsf{negl}(\lambda)$ at the end was added to account for the computational (as opposed to the information-theoretical) nature of the definition. In the information-theoretical system the security is guaranteed by the clear mathematical proof that the adversary can do nothing better than brute-force the system (e.g., iterating over one-time-pad keys). A computational security definition relies on the fact that the adversary simply does not have enough resources to break a system, which may or may not hold true in the future. For example, the problems of finding a discrete logarithm and factorization are only assumed to be hard, as are standardized block ciphers, such as AES [Dwo+01]. Therefore, if we want to allow a system to use these cryptographic blocks for security, we must include the negligible term in the definition.

Finally, note the query sequence $q_1, \ldots, q_m \in \mathcal{Q}$ is fixed. The definition is non-adaptive meaning that the user $\mathcal{U}$ must not choose the next query based on the result of the previous one. In [Bog+21, Section 3.1.1] we prove by example why this limitation is inherent and no efficient system can be built that would allow adaptive queries.

1.3.3 $\mathcal{E}$psolute

In $\mathcal{E}$psolute [Bog+21], we develop a method of answering point and range queries that satisfies Definition 2. At the heart of the construction is the use of ORAM to hide the access pattern and DP to add extra fake records to the result to hide communication volume.

ORAM component is straightforward — communication between $\mathcal{U}$ and $\mathcal{S}$ is routed through the ORAM protocol. This approach, however, substantially increases the overhead, which I will address in Section 1.3.4.

To optimally add noise to the true result we make use of sanitizers for point and range query types. For point queries, we construct a histogram from the domain values. We put the number of records for the domain value plus the noise sampled from the Laplacian distribution into each bin.

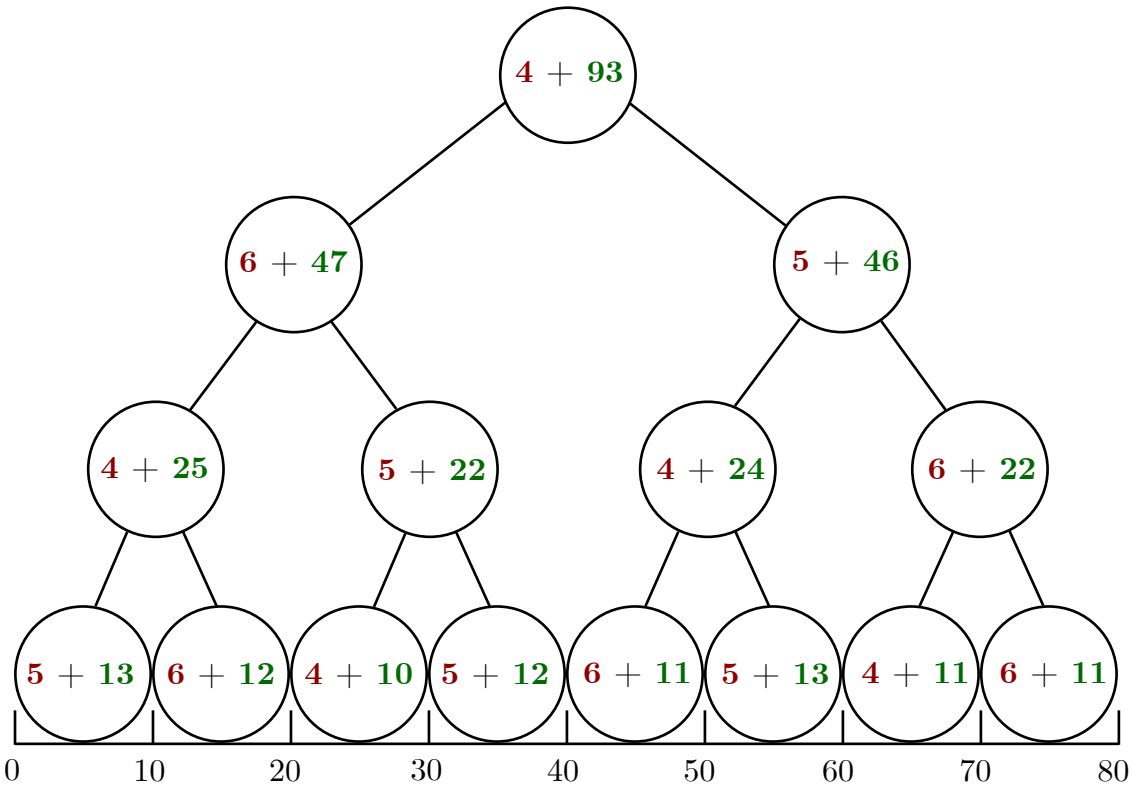

Figure 1·2: An example of a hierarchical sanitizer [QYL13] with fanout $k = 2$. Left (red) value is noise and right (green) value is the number of records covered by the node. To answer a query $[3, 28]$, we take only two nodes $(4 + 25)$ and $(4 + 10)$ with a resulting noise of 8 and 35 real records.

For the range queries we use a hierarchical sanitizer [QYL13], see example in Figure 1·2. We first split the continuous domain into bins. Then we build a k-ary tree over domain bins and put the number of records plus noise into each node. The

leaves get values the same way as in histogram, and intermediate node value is a sum of counts of all records in all children bins plus a single noise sample. Compared to a plain histogram, this construction has the benefit that if the range is wide we can answer the query using only the intermediate node and not all leaf nodes. Since each bin is now associated with $\lceil \log_k B \rceil$ nodes, where B is the number of bins, we amplify our noise respectively.

Note that the Laplacian distribution is symmetrical and unbounded, so if we sample from a zero-mean distribution some of the samples will be negative. Since we do not allow negative noise and since naïvely truncating the distribution will break DP guarantees, we need to shift the mean of the distribution. Doing so will only increase the noise thereby maintaining the guarantees. We have analytically derived the smallest mean of the distribution to ensure that the samples are non-negative for the entire histogram or tree with a set probability δ. These values are

$$\mu_{\mathsf{point}} = \left\lceil -\frac{\ln\left(2 - 2\sqrt[N]{1-\delta}\right)}{\epsilon} \right\rceil \quad \text{and} \quad \mu_{\mathsf{range}} = \left\lceil -\frac{\ln\left(2 - 2\sqrt[\mathsf{nodes}]{1-\delta}\right) \cdot \log_k N}{\epsilon} \right\rceil$$

for domain size N and the number of nodes in a tree nodes.

With these prerequisites we construct the actual setup and query protocols. See the visualization of Π_{query} in Figure 1·3.

- Setup protocol Π_{setup}

    1. On input a dataset $\mathcal{D}$, $\mathcal{U}$ creates and keeps locally an inverted index over the records mapping the search keys SK to record IDs r^{ID}.
    2. $\mathcal{U}$ and $\mathcal{S}$ engage in the ORAM protocol, where $\mathcal{U}$ is a client. $\mathcal{U}$ stores all records by their IDs.
    3. $\mathcal{U}$ constructs a sanitized structure $\mathcal{DS}$, a DP-histogram for point queries or a DP-tree for range queries, over the dataset domain and posts it to $\mathcal{S}$.

- Query protocol Π_{query}

  1. On input a query q , $\mathcal{U}$ looks up record IDs in its local index.
  2. $\mathcal{U}$ requests $\mathcal{DS}$ from $\mathcal{S}$ and uses it to calculate the total number of requests c (the number of true records for the query plus noise).
  3. $\mathcal{U}$ and $\mathcal{S}$ engage in the ORAM protocol, where $\mathcal{U}$ is a client. $\mathcal{U}$ loads c records where these requests include all true records and the rest are random and non-repeating.
  4. $\mathcal{U}$ prunes the fake records and returns the true ones.

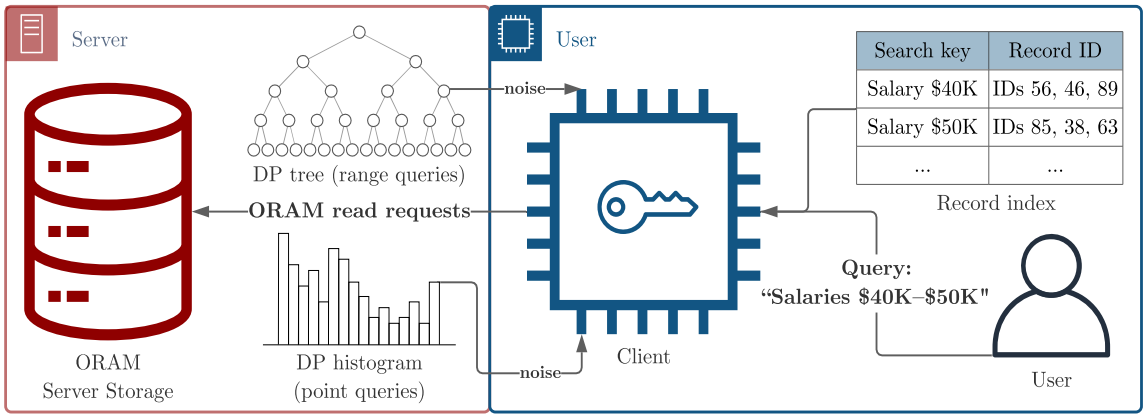

Figure 1·3: Query protocol Π_{query} of $\mathcal{E}$psolute.

This method, to which I will refer to as *a single-threaded $\mathcal{E}$psolute*, is a Computationally DP Outsourced Database System as defined in Definition 2, see proof in [Bog+21, Section 4.2].

1.3.4 Parallel $\mathcal{E}$psolute

While the single-threaded $\mathcal{E}$psolute is an efficient CDP-ODB in the sense that it returns only a fraction of total records per query, it is still slow in practice because of the $\mathcal{O}(\log n \log N)$ overhead due to ORAM and DP-sanitizer. To bring the construction closer to real-world demands in performance we devise a parallel solution.

A choice of separate vs shared sanitized dataset

A natural approach is to split the dataset $\mathcal{D}$ into m partitions and run parallel ORAM protocols against a distributed $\mathcal{S}$. The partitioning part is not hard — we split records by the hash of their unique ID and construct smaller (and thus logarithmically faster) ORAMs over the partitions along with m inverted indices on $\mathcal{U}$. However, we need to decide how we sample DP noise in this setting. The two approaches are generating a separate $\mathcal{DS}$ per partition or using a shared $\mathcal{DS}$.

When generating a separate DP-sanitized dataset $\mathcal{DS}$ per partition there are two competing effects on performance. On the one hand, each thread on average returns a $1/m$ fraction of the result from an ORAM with overhead $\log \frac{n}{m}$. On the other hand, each thread now adds independent noise, which results in m times amplification of noise. Moreover, due to the DP composition theorem [McS09; DMNS06; Dwo+06] for disjoint datasets, the privacy budget of the system is the privacy budget of the least private component. Since we cannot assume that the distribution of resulting records in all queries is always uniform, we bound a worst-case scenario of a single ORAM having most of the result records. In [Bog+21, Section 5.1] we have found that the noise is amplified by an extra $\left(1 + \sqrt{\frac{-3m \log \delta}{k_0}}\right)$ factor, where k_0 is the number of true records for a query.

When generating a single shared $\mathcal{DS}$ for all m partitions, we need to ensure that the total number of records returned by each thread is the same. We reuse the amplification factor from previous approach and set new $\tilde{k}_0 = k_0 + \frac{\log^{1.5} N}{\epsilon}$ for range queries and $\tilde{k}_0 = k_0 + \frac{\log N}{\epsilon}$ for point queries due to the use of DP sanitizers. The total generated noise is now larger but it is now split among m threads and it grows slower than linear in m.

Comparing these two methods we conclude that in most cases the shared $\mathcal{DS}$ is faster (i.e., results in fewer total fetched records), especially for an increasing par-

allelization factor. First, the former method is not guaranteed to have a uniform load among threads. In fact, while on average the amount of work per thread may be smaller, there may be a single disproportionally overloaded thread, which would cause the entire system to slow down. Second, while the number of true records per thread will decrease linearly for the former method, the noise factor will stay constant and will at larger m dominate the amount of work precluding further scalability. The latter method is more scalable in this regard since the amount of work decreases linearly while the amount of noise increases only logarithmically with the number of threads.

The construction

With this analysis I present an upgraded construction, *a parallel $\mathcal{E}psolute$*, see Figure 1·4. The core protocol is similar to the single-threaded version except some of the operations are distributed and executed in parallel.

- Setup protocol Π_{setup}

    1. On input a dataset $\mathcal{D}$, $\mathcal{U}$ partitions it into m segments $\mathcal{D}_1, \ldots, \mathcal{D}_m$ by a hash of record IDs r^{ID}. $\mathcal{U}$ then creates and keeps locally m inverted indices over the records mapping the search keys SK to record IDs r^{ID} in a respective partition.
    2. $\mathcal{U}$ and distributed $\mathcal{S}$ engage into m ORAM protocols, where $\mathcal{U}$ is a client. $\mathcal{U}$ stores all records by their IDs in respective $\mathcal{S}$ partitions.
    3. $\mathcal{U}$ constructs a sanitized structure $\mathcal{DS}$ (or m such structures depending on the choice of method) over the domain of the dataset and posts it to $\mathcal{S}$.

- Query protocol Π_{query}

    1. On input a query q , $\mathcal{U}$ looks up record IDs in its local indices.

2. $\mathcal{U}$ requests (one or multiple) $\mathcal{DS}$ from $\mathcal{S}$ and uses it to calculate the total number of requests c. Depending on the choice of method, $\mathcal{U}$ prepares a sequence of requests to $\mathcal{S}$.
3. $\mathcal{U}$ and $\mathcal{S}$ engage into m ORAM protocols, where $\mathcal{U}$ is a client. $\mathcal{U}$ loads a total of c records where these requests include all true records and the rest are random and non-repeating.
4. $\mathcal{U}$ waits for the last thread to finish, prunes the fake records and returns the true ones.

As a purely practical optimization, instead of making a single client VM engage into m (up to 128 in the experiments) connections and heavy cryptographic threads, we logically split $\mathcal{U}$ into a principal VM, which handles main logic, and a set of helper VMs that only engage in a number of ORAM protocols.

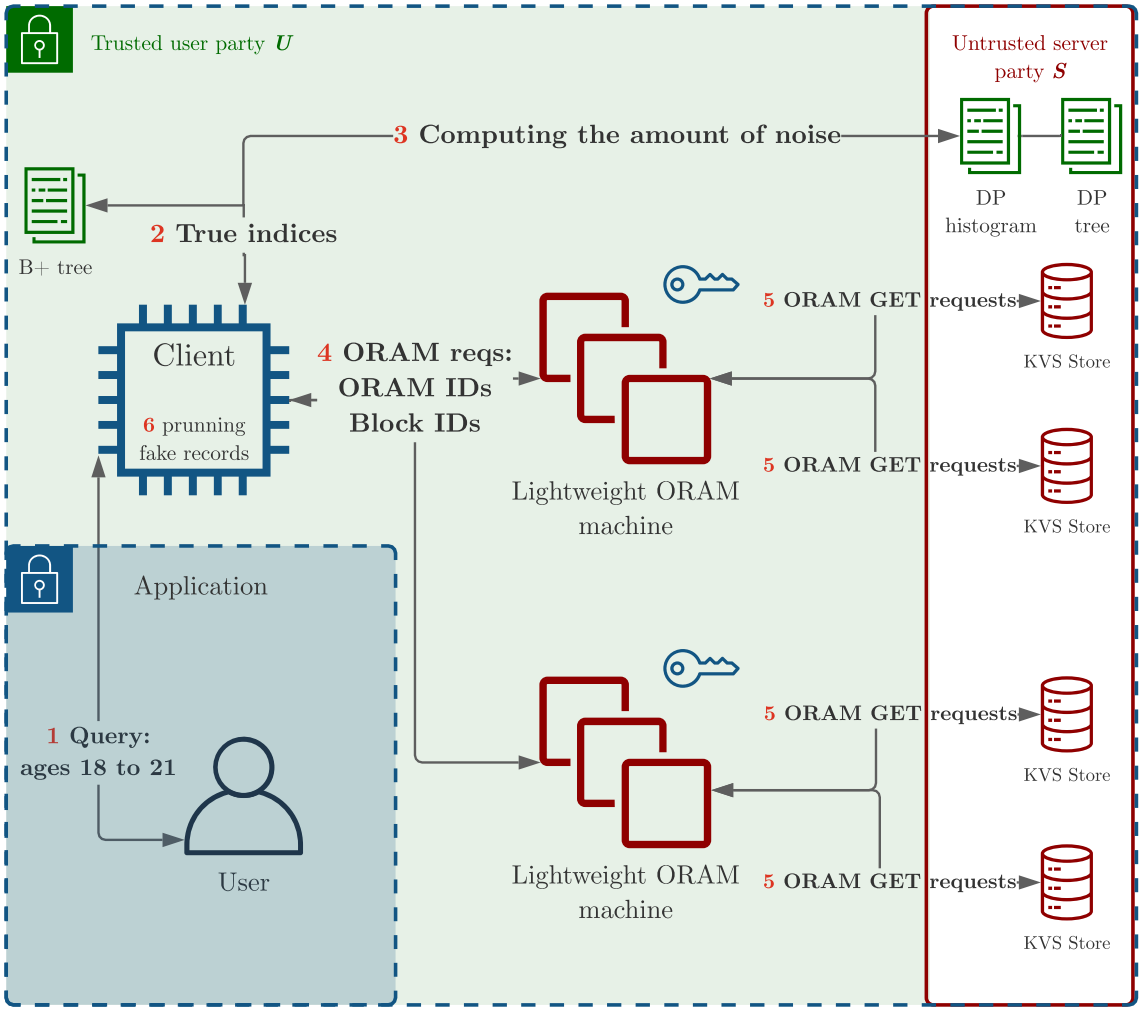

Figure 1·4: Query protocol Π_{query} of parallel $\mathcal{E}$psolute.

1.3.5 Experimental evaluation

To empirically verify the efficiency and practicality of our solution, we have implemented the construction in C++ [Bog21] and have run an extensive set of experiments. We have designed the experiments to (a) compare our system to competing proposed systems, (b) measure the storage overhead, (c) tune the parameters of the system and observe the effect on performance, (d) see how the system scales, (e) understand the quantitative impact of our optimizations, and (f) measure the impact of supporting multiple attributes.

For the default setting we ran range queries in the parallel $\mathcal{E}$psolute using the shared $\mathcal{DS}$ method. Client VM managed $m = 64$ threads, which are split among 8 lightweight ORAM VMs. Client $\mathcal{U}$ and server $\mathcal{S}$ are located in geographically distributed regions, with the ping time between the regions 12 ms and the effective bandwidth 150 MB/s. In this section I will focus on comparing $\mathcal{E}$psolute to relevant systems and on $\mathcal{E}$psolute's scalability. I refer to the full work for the rest of the experimental findings [Bog+21, Section 6].

Against relevant solutions

In [Bog+21], we compared $\mathcal{E}$psolute (a) to conventional SQL RDBMSs (PostgreSQL and MySQL), (b) to a linear scan approach, and (c) to Shrinkwrap [Bat+18].

RDBMSs are highly optimized for range queries — the records indexed by search keys are typically stored continuously in-order and the database streams back these blocks very efficiently. As a baseline approach, RDBMSs were configured for maximum performance and minimum security.

Linear scan is a strawman approach, which downloads the entire encrypted dataset every query, decrypts it and then runs the query locally. This is the opposite baseline of maximum security and poorest performance. Such an approach is not entirely

theoretical: some commercial RDBMSs exhibit linear scan behavior when configured for maximum security, for example MS-SQL Always Encrypted with Randomized Encryption [Mic21] and Oracle Column Transparent Data Encryption [Ora21].

Shrinkwrap [Bat+18] is a construction that executes federated SQL queries securely, albeit inefficiently. To hide communication volume it fully pads the intermediate results of the sub-queries (although it "shrinks" the output before passing it to the next stage). To hide the access pattern, the algorithm makes a linear scan over the entire input. Moreover, to process the encrypted data (e.g., evaluate a predicate), the whole program is compiled into garbled circuits [Yao86; WMK16].

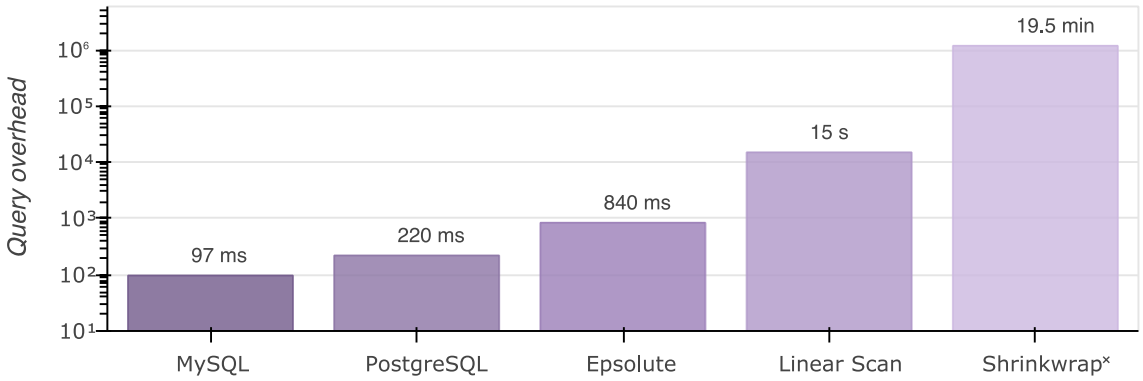

Figure 1·5: Different range-query mechanisms (log scale). Default setting: 10^6 4 KiB uniformly-sampled records with the range 10^4.

We conclude that $\mathcal{E}$psolute is efficient — while answering a query in 840 ms, it is only 8 to 4 times slower than a typical Relational Database Management System, 18 times faster than the scan, and three orders of magnitude faster than the Shrinkwrap, see Figure 1·5.

Scalability

To measure scalability we have run parallel $\mathcal{E}$psolute with shared and separate DP-sanitized datasets $\mathcal{DS}$ for a varying number of threads m. For a truly scalable system we would observe a drop in overhead roughly proportional to the increase in par-

allelization factor m. We confirm experimentally that $\mathcal{E}$psolute is scalable when a shared $\mathcal{DS}$ is used as a method of noise generation, see Figure 1·6. The noise dominates the performance of the separate $\mathcal{DS}$ method for a higher number of threads, while the shared $\mathcal{DS}$ method continues to scale.

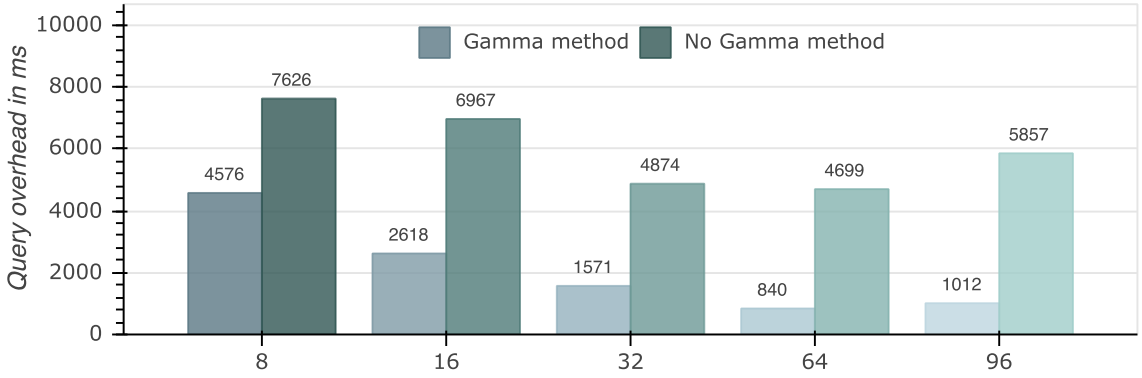

Figure 1·6: Scalability measurements for shared $\mathcal{DS}$ structure Π_{γ} and separate $\mathcal{DS}$ structures $\Pi_{\mathsf{no}-\gamma}$

1.4 kNN queries in a snapshot model

Nearest neighbor search is a type of optimization problem that, given a set of objects and a distance metric, requires finding the object closest to a given object according to the distance metric. A k-nearest-neighbor (kNN) search is a subtype of a general nearest neighbor problem where k closest objects are requested. Applications that use kNN search only need to define the objects and the metric. For example, a street map application would define the 2D coordinates of the buildings as objects and Euclidean distance as a metric, then the query could be "give 5 restaurants closest to the current user position". A document search application would define the keyword vector for a document as an object and an inner product distance as a metric, then the query could be "give 3 documents most similar to the given text" (similar applications may search images, videos and sounds).

To run a kNN query securely in an outsourced database model, I propose to use an approach similar to ORE for range queries. While to run a range query one needs to be able to *compare* the ciphertexts (exactly what ORE does), to run a kNN query one needs to know *a distance comparison* between pairs of ciphertexts. That is, one needs to maintain (or reveal) an order of distances of all pairs of encrypted objects. For example, if x was further from y than z was, then the encryption of x needs to be further from the encryption of y than the encryption of z is.

Although such a Distance Comparison Preserving Encryption (DCPE) scheme would allow running the query with absolute accuracy, the construction would suffer from the same security issues that the ORE methods did. Most of all, while ORE-encrypted dataset reveals the total order, the DCPE-encrypted values will reveal *the total relative position* of all elements. The exact distances will be hidden (subject to the DCPE scheme's leakage), but the relative position is a substantial leakage that may lead to reconstruction attacks. To mitigate the leakage we may use a form of *an approximate* DCPE, which ideally would add controlled noise or inaccuracy to the relative distances.

1.4.1 Distance Comparison Preserving Encryption

A candidate approximate DCPE scheme has been proposed by Fuchsbauer et al. [FGHO21]. The scheme provides the following guarantee on its ciphertexts

$$\forall x, y, z \in \mathbb{X} : \textsc{Dist}\,(x, y) < \textsc{Dist}\,(x, z) - \beta \implies \textsc{Dist}\,(f(x), f(y)) < \textsc{Dist}\,(f(x), f(z))$$

where $\mathbb{X} \subseteq \mathbb{R}^d$ is the set of d-dimensional vectors of real numbers, Dist is the inner product distance over elements in $\mathbb{X}$, and β is the approximation factor. Parameter β partially defines the security of the encrypted set — the larger β, the fewer distance

comparisons are preserved, the less accurate the search and the reconstruction attacks would be. Fuchsbauer et al. [FGHO21] prove protection against membership inference attacks [YGFJ18] (whether an individual is in the database or not), and against approximate frequency-finding attacks (how many times the element appears in the set, see [Gru+17] for ORE frequency attacks). As for the choice of β, [FGHO21] proves that $\beta \approx \sqrt{\max N}$ would hide about half of the inputs bits, for $\max N$ being the maximum vector length in the dataset.

The scheme works by amplifying the input vector length by a secret factor, then constructing a d-dimensional β-radius hyperball, and finally setting the vector's tip to a uniformly sampled point on that ball. The decryption uses the same secret seed and thus constructs the same amplification factor and the same ball, then makes the inverse arithmetic steps to derive the original vector. The security of the scheme thus depends on the maximum amount of amplification, the radius of the hyperball β and the size of seed for the samplers. As the construction operates on real numbers, an open question remains on how to avoid negative side-effects of floating point numbers bit representation. I refer to the original paper [FGHO21] for the detailed β-DCPE construction.

1.4.2 The setup and query protocols

With the β-DCPE as a component, we can model the protocols similar to ORE ones. In the setup protocol Π_{setup}, $\mathcal{U}$ simply encrypts the entire input, one vector at a time, and sends the encrypted data over to $\mathcal{S}$. In the query protocol Π_{query}, $\mathcal{U}$ encrypts the query with DCPE, sends the ciphertext to $\mathcal{S}$, while $\mathcal{S}$ runs a standard kNN search against the ciphertext. k encrypted vectors are then returned to $\mathcal{U}$, who decrypts them as the last step. These protocols run for a single set of secrets including β.

To measure the effect of different levels of security on search accuracy, I propose to repeat the experiments for different values of β.

For the choice of the dataset, I suggest using the established information retrieval TREC test collections. Such collection consists of a set of documents, a set of topics (questions) and a corresponding set of relevance judgments (correct answers). To use the test collection, we need to convert the documents and queries into vectors.[1] A benefit of using a TREC dataset is being able to evaluate relevant metrics over the produced results, for example, MRR [Cra09] and DCG [JK02]. We can then track how these metrics, along with the simpler edit distance and set difference over the result, degrade with higher security.

Lastly, for an actual implementation I suggest using an existing component for the bare kNN search. FAISS [JDJ17] is a GPU-enabled library for efficient similarity search and clustering of dense vectors. Given that the TREC vectors are $d = 768$ dimensional with a maximum Euclidean length of 11, FAISS seems to be an ideal candidate.

[1] Hamed Zamani has collaborated with us and provided the vectorized data and query sets.

Appendix A

Summary of OPE/ORE and Range Queries analysis

Table A.1: [BKR19, Tables 1 and 4]. Primitive usage by OPE / ORE schemes. Ordered by security rank — most secure below. n is the input length in bits, d is a block size for Lewi-Wu [LW16] scheme, λ is a PRF output size, N is a total data size, **HG** is a hyper-geometric distribution sampler, **PPH** is a property-preserving hash with h-bit outputs built with bilinear maps and **bolded** are weak points of the schemes. Values in parentheses are simulation-derived. $N = 10^3$, $n = 32$, $d = 2$, $\lambda = 128$ and $h = 128$ in this simulation.

Scheme	Primitive usage (number of invocations)		Ciphertext size, or state size (bits)	Leakage (in addition to inherent total order)
	Encryption	Comparison		
[BCLO09]	**$\approx n$ (41) HG**	none	$2n$ (64)	**$\approx$ Top half of the bits**
[CLWW16a]	n (32) PRF	none	$2n$ (64)	**Most-significant differing bit**
[LW16]	**$2^{2n/d}$ (32) PRP** $2\frac{n}{d}\left(2^d+1\right)$ (160) PRF $\frac{n}{d}2^d$ (64) Hash	$\frac{n}{2d}$ (9) Hash	$\frac{n}{d}\left(\lambda+n+2^{d+1}\right)+\lambda$ (2816)	Most-significant differing block
[Cas+18]	n (32) PRF n (32) PPH 1 PRP	**n^2 (1046) PPH**	$n \cdot h$ (4096)	Equality pattern of the most-significant differing bit
[Ker15]	1 Traversal	3 Traversals	**$3 \cdot n \cdot N$ (86842)**	Insertion order

Table A.2: [BKR19, Tables 2 and 3]. Performance of the range query protocols. Ordered by security rank — most secure below. N is a total data size, B is an I/O page size, L is a POPE tree branching factor, r is the result size in records and **bolded** are weak points of the protocols. The cell content is structured as follows: top value is the analytical result in $\mathcal{O}$ notation, bottom value is the number of requests for I/O requests or number of messages and their total size for communication. In these experiments, $N = 247K$, $B = 4\,\text{kB}$, $r \approx 247K \cdot 0.5\,\% = 1235$, and $L = 60$.

Protocol	I/O requests		Leakage	Communication	
	Construction	Query		Construction	Query
B+ tree with ORE	$\log_B \frac{N}{B}$ 3 requests	$\log_B \frac{N}{B} + \frac{r}{B}$ 44 requests	**Same as ORE**	1 2 / 177 B	1 2 / 342 B
[KT19]	$\boldsymbol{\frac{N}{B}}$ **494 requests**	$\log_2 \frac{N}{B} + \frac{r}{B}$ 17 requests	**Total order**	$\log_2 N$ 40 / 671 B	$\log_2 N$ 86 / 1 453 B
[RACY16] warm	1 1 request	$\log_L \frac{N}{B} + \frac{r}{B}$ 300 requests	**Partial order**	1 2 / 32 B	$\log_L N$ 914 / 347 kB
[RACY16] cold		$\boldsymbol{N/B}$ **2175 requests**	Fully hiding		$\boldsymbol{N}$ **498K / 9 MB**
[Dem+16]	—	$\boldsymbol{r}$ **40 requests**	Same as SSE	—	$\log_2 N$ 2 / 391 B
ORAM	$\boldsymbol{\log^2 \frac{N}{B}}$ **31 requests**	$\boldsymbol{\log_2 \frac{N}{B} \left(\log_B \frac{N}{B} + \frac{r}{B}\right)}$ **185 requests**	Fully hiding (access pattern)	$\boldsymbol{\log^2 \frac{N}{B}}$ **143 / 18 kB**	$\boldsymbol{\log^2 \frac{N}{B}}$ **490 / 63 kB**

Bibliography

[Bog21] D. Bogatov. *$\mathcal{E}$psolute*. https://github.com/epsolute/epsolute. 2021.

[Bog+21] **D. Bogatov**, G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. "$\mathcal{E}$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.

[FGHO21] G. Fuchsbauer, R. Ghosal, N. Hauke, and A. O'Neill. *Approximate Distance-Comparison-Preserving Symmetric Encryption*. Cryptology ePrint Archive, Report 2021/1666. https://ia.cr/2021/1666. 2021.

[Mic21] Microsoft. *MS-SQL Always Encrypted*. https://docs.microsoft.com/sql/relational-databases/security/encryption/always-encrypted-database-engine. 2021.

[Ora21] Oracle. *Introduction to Transparent Data Encryption*. https://docs.oracle.com/database/121/ASOAG/introduction-to-transparent-data-encryption.htm. 2021.

[Kap+20] H. Kaplan, K. Ligett, Y. Mansour, M. Naor, and U. Stemmer. "Privately Learning Thresholds: Closing the Exponential Gap". In: *Proceedings of Thirty Third Conference on Learning Theory*. Vol. 125. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2263–2285.

[KPT20] E. M. Kornaropoulos, C. Papamanthou, and R. 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.

[BNZ19] A. Beimel, K. Nissim, and M. Zaheri. "Exploring Differential Obliviousness". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2019)*. Vol. 145. Leibniz International Proceedings in Informatics (LIPIcs). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, 65:1–65:20. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2019.65.

[BKR19] **D. Bogatov**, G. Kollios, and L. Reyzin. "A comparative evaluation of order-revealing encryption schemes and secure range-query protocols". In: *Proceedings of the VLDB Endowment* 12.8 (2019), pp. 933–947. DOI: 10.14778/3324301.3324309.

[CCMS19] T.-H. H. Chan, K.-M. Chung, B. M. Maggs, and E. Shi. "Foundations of Differentially Oblivious Algorithms". In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '19. San Diego, California: Society for Industrial and Applied Mathematics, 2019, pp. 2448–2467.

[GJW19] Z. Gui, O. Johnson, and B. 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.

[KT19] F. Kerschbaum and A. Tueno. "An Efficiently Searchable Encrypted Data Structure for Range Queries". In: *Computer Security – ESORICS 2019*. Springer International Publishing, 2019, pp. 344–364.

[Bat+18] J. Bater, X. He, W. Ehrich, A. Machanavajjhala, and J. Rogers. "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.

[Bin+18] V. Bindschaedler, P. Grubbs, D. Cash, T. Ristenpart, and V. Shmatikov. "The Tao of Inference in Privacy-Protected Databases". In: *PVLDB* 11.11 (2018), pp. 1715–1728. DOI: 10.14778/3236187.3236217.

[Cas+18] D. Cash, F.-H. Liu, A. O'Neill, M. Zhandry, and C. Zhang. "Parameter-Hiding Order Revealing Encryption". In: *Advances in Cryptology – ASIACRYPT 2018*. 2018, pp. 181–210.

[GLMP18] P. Grubbs, M.-S. Lacharité, B. Minaud, and K. G. Paterson. "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.

[LMP18] M.-S. Lacharité, B. Minaud, and K. 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.

[YGFJ18] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha. "Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting". In: *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. 2018, pp. 268–282. DOI: 10.1109/CSF.2018.00027.

[GRS17] P. Grubbs, T. Ristenpart, and V. 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.

[Gru+17] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. "Leakage-Abuse Attacks against Order-Revealing Encryption". In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 655–672. DOI: 10.1109/SP.2017.44.

[JDJ17] J. Johnson, M. Douze, and H. Jégou. "Billion-scale similarity search with GPUs". In: *arXiv preprint arXiv:1702.08734* (2017).

[CLWW16a] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. "Practical Order-Revealing Encryption with Limited Leakage". In: *Fast Software Encryption*. Springer Berlin Heidelberg, 2016, pp. 474–493.

[CLWW16b] N. Chenette, K. Lewi, S. A. Weis, and D. J. Wu. "Practical Order-Revealing Encryption with Limited Leakage". In: *Fast Software Encryption*. Springer Berlin Heidelberg, 2016, pp. 474–493.

[Dem+16] I. Demertzis, S. Papadopoulos, O. Papapetrou, A. Deligiannakis, and M. Garofalakis. "Practical private range search revisited". In: *Proceedings of the 2016 International Conference on Management of Data*. 2016, pp. 185–198. DOI: 10.1145/2882903.2882911.

[KKNO16] G. Kellaris, G. Kollios, K. Nissim, and A. O'neill. "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.

[LW16] K. Lewi and D. J. Wu. "Order-revealing encryption: New constructions, applications, and lower bounds". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 1167–1178. DOI: 10.1145/2976749.2978376.

[RACY16] D. S. Roche, D. Apon, S. G. Choi, and A. Yerukhimovich. "POPE: Partial Order Preserving Encoding". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1131–1142.

[WMK16] X. Wang, A. J. Malozemoff, and J. Katz. *EMP-toolkit: Efficient Multi-Party computation toolkit*. https://github.com/emp-toolkit. 2016.

[BNSV15] M. Bun, K. Nissim, U. Stemmer, and S. Vadhan. "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.

[CGPR15] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. "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.

[Ker15] F. Kerschbaum. "Frequency-Hiding Order-Preserving Encryption". In: *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 656–667.

[NKW15] M. Naveed, S. Kamara, and C. 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.

[BBKN14] A. Beimel, H. Brenner, S. P. Kasiviswanathan, and K. Nissim. "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.

[IKK14] M. S. Islam, M. Kuzu, and M. 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.

[BNS13] A. Beimel, K. Nissim, and U. 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.

[BLR13] A. Blum, K. Ligett, and A. 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.

[QYL13] W. Qardaji, W. Yang, and N. Li. "Understanding hierarchical methods for differentially private histograms". In: *Proceedings of the VLDB Endowment* 6.14 (2013), pp. 1954–1965. DOI: 10.14778/2556549.2556576.

[Ste+13] E. Stefanov, M. van Dijk, E. Shi, C. Fletcher, L. Ren, X. Yu, and S. Devadas. "Path ORAM: An Extremely Simple Oblivious RAM Protocol". In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security*. ACM, 2013, pp. 299–310.

[HMCK12] B. Hore, S. Mehrotra, M. Canim, and M. Kantarcioglu. "Secure multidimensional range queries over outsourced data". In: *VLDBJ* 21.3 (2012), pp. 333–358. DOI: 10.1007/s00778-011-0245-7.

[IKK12] M. S. Islam, M. Kuzu, and M. 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.

[DNPR10] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. "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.

[BCLO09] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. "Order-Preserving Symmetric Encryption". In: *Advances in Cryptology - EUROCRYPT 2009*. Springer Berlin Heidelberg, 2009, pp. 224–241. DOI: 10.1007/978-3-642-01001-9_13.

[Cra09] N. Craswell. "Mean Reciprocal Rank". In: *Encyclopedia of Database Systems*. Boston, MA: Springer US, 2009, pp. 1703–1703. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_488.

[McS09] F. D. McSherry. "Privacy integrated queries: an extensible platform for privacy-preserving data analysis". In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 2009, pp. 19–30. DOI: 10.1145/1559845.1559850.

[Dwo+06] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. "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.

[DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. "Calibrating noise to sensitivity in private data analysis". In: *Theory of cryptography conference*. Springer. 2006, pp. 265–284. DOI: 10.1007/11681878_14.

[JK02] K. Järvelin and J. Kekäläinen. "Cumulated Gain-Based Evaluation of IR Techniques". In: *ACM Trans. Inf. Syst.* 20.4 (Oct. 2002), pp. 422–446. ISSN: 1046-8188. DOI: 10.1145/582415.582418.

[Dwo+01] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and J. Dray. *Advanced Encryption Standard (AES)*. en. 2001. DOI: 10.6028/NIST.FIPS.197.

[Yao86] A. C.-C. Yao. "How to generate and exchange secrets". In: *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. 1986, pp. 162–167. DOI: 10.1109/SFCS.1986.25.