

BOSTON UNIVERSITY  
GRADUATE SCHOOL OF ARTS AND SCIENCES

Dissertation

**SECURE AND EFFICIENT QUERY PROCESSING IN  
OUTSOURCED DATABASES**

by

**DMYTRO BOGATOV**

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

Submitted in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy

2022

© 2022 by  
DMYTRO BOGATOV  
All rights reserved

Approved by

First Reader Approved

George Kollios, Ph.D.  
Professor of Computer Science

Second Reader Approved

Leonid Reyzin, Ph.D.  
Professor of Computer Science

Third Reader Approved

Manos Athanassoulis, Ph.D.  
Assistant Professor of Computer Science

Fourth Reader Approved

Adam O'Neil, Ph.D.  
Assistant Professor of Computer Science  
University of Massachusetts — Amherst

*Ask not what your country can do for you —  
ask what you can do for your country.*

JOHN FITZGERALD KENNEDY

## Dedication

I dedicate my dissertation work to my wife, my parents and the brave defenders of Ukraine. Dasha, love of my life, thank you for supporting me since the very first day of the program, through the long distance, once-in-a-century world-wide health crisis and the war. It was with your help that I was able to come so far. A special feeling of gratitude to my loving parents, Kostiantyn and Nataliia, whose help early in my academic career was critical and enabled my doctorate program in the first place, and whose support never waned to this day. Lastly, my heart and my thoughts go to the brave people of Ukraine, who are heroically defending the Motherland from the Russian invasion with their lives. We are with you, physically and virtually, in Lviv and Mariupol, in Kharkiv and in Kyiv. My sincere hope, my wish, is that this work will contribute, however little, to the cause of my beloved nation. Слава Україні!

## Acknowledgments

First and foremost, I want to thank my advisor, George Kollios, for his guidance and continued support throughout my doctorate journey, from the day he accepted me to his lab to this moment. I am honored to call him and his family my friends. I also thank Leo Reyzin for the time and effort he spent helping me with my work, and for the many cups of coffee and long life talks we have had over these years. I want to especially note the many hours-long philosophical discussions with Manos Athanassoulis — the discussions I truly enjoyed. In the heat of our debates we have lived up to the Ph. part of our degrees. My special thanks go to Adam O’Neil for his countless contributions to my works. He has been an invaluable co-author, contributor, reviewer and friend.

I would like to express my gratitude to my closest ally in this life, my beloved wife, Daria Bogatova. There is not a single piece of writing I did without her review, not a single figure I crafted without her keen eye refining the tiniest details, from fonts to colors. Even the source code of this very thesis has been branched off from her Bachelor work [Bog21], and the names of my systems,  $\mathcal{E}$ psolute and *k-anon* are her inventions. She has always been the first to listen to my ideas, and the first to support them unconditionally.

I would particularly like to thank my bosom friend, roommate and fellow Doctor, Oleksandr Narykov. Living in Boston will not be the same without him, and I hope his career will bring him here in no time. I also want to thank my good friend and fellow BU alumnus, Vasili Ramanishka. I miss our arguments and discussions in the middle of a night.

My deepest appreciation goes to my colleagues and co-authors, Georgios Kellaris, Björn Tackmann, Kaoutar Elkhiyaoui, Angelo De Caro, Kobbi Nissim and Hamed Zamani. It has been a great pleasure and a rewarding experience working with you.

Your input in my work has been indispensable. I am deeply grateful to my fellow Amazonians, Kiran Chinta, Sriram Krishnamurthy, Naresh Chainani, Ramchandra Kulkarni and Abhishek Rai Sharma with whom I shared a number of internships. I am very excited to join the team full-time!

I owe my deepest gratitude to my family, my parents, grandma and a little sister. I thank you for always being there for me. For attending my defense and admiring my presentation even though the content may have been very technical and in a foreign language. I am also infinitely glad that my family has expanded during my doctorate program. I thank Dasha's side of the family, in the similar composition of parents, grandma and a little sister, for accepting me and supporting me through my journey.

Last but not least, I want to note my little fluffy friend, Pixel. My years of the program would not have been the same without him sleeping on my keyboard.

# SECURE AND EFFICIENT QUERY PROCESSING IN OUTSOURCED DATABASES

DMYTRO BOGATOV

Boston University, Graduate School of Arts and Sciences, 2022

Major Professor: George Kollios, Ph.D.  
Professor of Computer Science

## ABSTRACT

As organizations struggle with processing vast amounts of information, outsourcing sensitive data to third parties becomes a necessity. Various cryptographic techniques are used in outsourced database systems to ensure data privacy while allowing for efficient querying. This thesis proposes 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.

This work starts with the survey of 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 tradeoffs in terms of security and efficiency. The survey 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.

The thesis then proceeds with  $\mathcal{E}$ psolute — an efficient range query engine in a persistent adversary model. In  $\mathcal{E}$ psolute, security is achieved 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.  $\mathcal{E}$ psolute proposes a



definition, construction, analysis, and experimental evaluation of a system that provably hides both access pattern and communication volume while remaining efficient.

The dissertation concludes with *k-anon* — a secure similarity search engine in a snapshot adversary model. The work presents a construction in which the security of  $k$ NN queries is achieved similarly to OPE / ORE solutions — encrypting the input with an approximate Distance Comparison Preserving Encryption scheme so that the inputs, the points in a hyperspace, are perturbed, but the query algorithm still produces accurate results. Analyzing the solution, we run a series of experiments to observe the tradeoff between search accuracy and attack effectiveness. We use TREC datasets and queries for the search, and track the rank quality metrics such as MRR and nDCG. For the attacks, we build an LSTM model that trains on the correlation between a sentence and its embedding and then predicts words from the embedding. We conclude on viability and practicality of the solution.

# Contents

<b>Contents</b>	<b>x</b>
<b>List of Algorithms</b>	<b>xvi</b>
<b>List of Tables</b>	<b>xvii</b>
<b>List of Figures</b>	<b>xix</b>
<b>List of Acronyms</b>	<b>xx</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Model . . . . .	2
1.1.1 Outsourced database model . . . . .	2
1.1.2 Security model . . . . .	3
1.1.3 Query types . . . . .	4
1.2 Thesis structure . . . . .	5
1.2.1 Works completed during the Ph.D. program . . . . .	5
<b>2 Background</b>	<b>7</b>
2.1 Symmetric encryption . . . . .	7
2.1.1 Security . . . . .	7
2.1.2 Components . . . . .	9
2.2 Oblivious Random Access Machine . . . . .	10
2.2.1 PathORAM . . . . .	11
2.3 Differential Privacy . . . . .	12
2.3.1 DP sanitizers . . . . .	14

	Answering point and range queries with differential privacy . . .	15
	Composition . . . . .	15
2.4	Trusted Execution Environments . . . . .	16
2.4.1	Software Guard Extensions . . . . .	17
2.4.2	Issues with SGX . . . . .	18
<b>3</b>	<b>Related work</b>	<b>19</b>
3.1	Range query security in a snapshot model . . . . .	19
3.2	Range query security in a persistent model . . . . .	20
3.2.1	Obliviousness and volume-hiding without enclave . . . . .	21
3.2.2	Enclave-based solutions . . . . .	22
3.3	$k$ NN query security in a snapshot model . . . . .	22
<b>4</b>	<b>Range queries in the snapshot model</b>	<b>25</b>
4.1	Introduction . . . . .	26
4.2	Security Perspective . . . . .	28
4.2.1	A note on variable-length inputs . . . . .	30
4.3	OPE and ORE Schemes . . . . .	30
4.3.1	BCLO OPE [BCLO09] . . . . .	31
	Security . . . . .	32
	Analysis and implementation challenges . . . . .	32
4.3.2	CLWW ORE [CLWW16] . . . . .	33
	Security . . . . .	33
	Analysis and implementation challenges . . . . .	34
4.3.3	Lewi-Wu ORE [LW16] . . . . .	34
	Security . . . . .	35
	Analysis and implementation challenges . . . . .	35
4.3.4	CLOZ ORE [Cas+18] . . . . .	36

	Security . . . . .	37
	Analysis and implementation challenges . . . . .	37
4.3.5	FH-OPE [Ker15] . . . . .	38
	Security . . . . .	39
	Analysis and implementation challenges . . . . .	39
4.4	Secure Range Query Protocols . . . . .	40
4.4.1	Range query protocol from ORE . . . . .	42
	Security . . . . .	43
4.4.2	Kerschbaum-Tueno [KT19] . . . . .	43
	Security . . . . .	43
	Analysis and implementation challenges . . . . .	44
4.4.3	POPE [RACY16] . . . . .	45
	Security . . . . .	45
	Analysis and implementation challenges . . . . .	46
4.4.4	Logarithmic-BRC [Dem+16] . . . . .	47
	Security . . . . .	48
	Analysis and implementation challenges . . . . .	49
4.4.5	The two extremes . . . . .	50
	No encryption . . . . .	50
	ORAM . . . . .	50
4.5	Evaluation . . . . .	51
4.5.1	Implementation . . . . .	53
	Primitives . . . . .	54
	Schemes and protocols . . . . .	55
	Simulations . . . . .	55
4.5.2	Setup . . . . .	57

4.5.3	Results . . . . .	57
	Primitive usage by schemes . . . . .	57
	Benchmarks of schemes and primitives . . . . .	58
	Protocols . . . . .	59
4.6	Remarks and conclusion . . . . .	62
<b>5</b>	<b>Range queries in the persistent model</b>	<b>64</b>
5.1	Introduction . . . . .	65
5.2	Differentially private outsourced database systems . . . . .	68
5.2.1	Adversarial model . . . . .	68
	On impossibility of adaptive queries . . . . .	69
5.2.2	Query types . . . . .	70
5.2.3	Measuring Efficiency . . . . .	71
5.3	$\mathcal{E}$ psolute . . . . .	72
5.3.1	General construction . . . . .	72
5.3.2	Security . . . . .	74
5.3.3	Efficiency . . . . .	75
5.3.4	Extending to multiple attributes . . . . .	76
5.3.5	$\mathcal{E}$ psolute for point queries . . . . .	77
5.3.6	$\mathcal{E}$ psolute for range queries . . . . .	77
5.4	An efficient Parallel $\mathcal{E}$ psolute . . . . .	79
5.4.1	No- $\gamma$ -method: DP structure per ORAM . . . . .	80
5.4.2	$\gamma$ -method: shared DP structure . . . . .	81
5.4.3	Practical improvements . . . . .	82
	ORAM request batching . . . . .	82
	Lightweight ORAM servers . . . . .	83
5.5	Experimental Evaluation . . . . .	85

5.5.1	Data sets . . . . .	86
5.5.2	Default setting . . . . .	87
5.5.3	Experiment stages . . . . .	88
5.5.4	RDBMS, Linear Scan and Shrinkwrap . . . . .	88
5.5.5	Results and Observations . . . . .	90
5.6	Conclusion and Future Work . . . . .	99
<b>6</b>	<b><i>k</i>NN queries in the snapshot model</b>	<b>100</b>
6.1	Introduction . . . . .	100
6.2	Distance Comparison Preserving Encryption . . . . .	102
6.2.1	DCPE construction . . . . .	102
6.2.2	DCPE security . . . . .	104
6.2.3	DCPE implementation and benchmarks . . . . .	106
6.3	<i>k</i> NN search accuracy . . . . .	107
6.3.1	Secure <i>k</i> NN protocol . . . . .	107
6.3.2	Experimental evaluation . . . . .	108
	Ranking quality metrics . . . . .	109
6.3.3	Results for varying $\beta$ . . . . .	110
6.4	Security against attacks . . . . .	112
6.4.1	Black-box model inversion attack [SR20] . . . . .	113
6.4.2	Experimental evaluation . . . . .	114
	Attack efficiency metrics . . . . .	115
	Baselines . . . . .	116
	Public model . . . . .	118
	Private model . . . . .	119
6.5	Search accuracy against security tradeoff . . . . .	120
6.6	Conclusions . . . . .	122

Future Work . . . . .	122
<b>7 Conclusions and Future Work</b>	<b>123</b>
Practicality and reproducibility . . . . .	123
Practicality of property-preserving encryption . . . . .	124
Practicality of using “heavy” primitives and protocols . . . . .	124
More query types . . . . .	125
<b>A Abstract of [BCET21]</b>	<b>126</b>
<b>B Abstract of [NBK19]</b>	<b>127</b>
<b>Bibliography</b>	<b>128</b>
<b>Curriculum Vitæ</b>	<b>150</b>

# List of Algorithms

1	$\mathcal{E}$ psolute protocol . . . . .	71
2	Parallel $\mathcal{E}$ psolute for $\Pi_\gamma$ . . . . .	78
3	DCPE scheme . . . . .	103



# List of Tables

4.1	Primitive usage by OPE / ORE schemes . . . . .	41
4.2	Performance of the range query protocols . . . . .	52
5.1	$\mathcal{E}$ psolute storage usage for varying data, record and domain sizes . . .	93
5.2	Improvements over parallel $\mathcal{E}$ psolute . . . . .	97
6.1	DCPE implementation benchmarks . . . . .	106
6.2	Inversion attack performance for the private model experiments . . .	120

# List of Figures

2-1	Attack surface with TEE . . . . .	16
4-1	OPE / ORE schemes benchmark . . . . .	54
4-2	Cryptographic primitives benchmark . . . . .	54
4-3	Number of I/O requests for different protocols and data distributions	56
4-4	Communication size for different protocols and data distributions . .	56
4-5	Communication volume for different protocols and data distributions	57
4-6	Scalability: number of I/O requests . . . . .	60
4-7	Scalability: communication volume . . . . .	60
4-8	Performance for different query sizes . . . . .	61
4-9	Performance over time (queries) . . . . .	61
5-1	$\mathcal{E}$ psolute construction . . . . .	74
5-2	Parallel $\mathcal{E}$ psolute construction . . . . .	84
5-3	Different range-query mechanisms . . . . .	91
5-4	Linear scan performance . . . . .	92
5-5	Privacy budget $\epsilon$ . . . . .	94
5-6	Effect of $\epsilon$ . . . . .	94
5-7	Selectivity . . . . .	94
5-8	Record size . . . . .	95
5-9	Data size . . . . .	95
5-10	Domain size . . . . .	95
5-11	Data distribution . . . . .	96

5.12	Query distribution . . . . .	96
5.13	Scalability measurements for $\Pi_\gamma$ and $\Pi_{no-\gamma}$ . . . . .	97
5.14	Query overhead when using multiple attributes . . . . .	98
6.1	Schematic description of DCPE . . . . .	105
6.2	Search accuracy for $\beta \in \{0.0, \dots, 50.0\}$ . . . . .	111
6.3	Search accuracy for $\beta \in \{0.0, \dots, 5.0\}$ . . . . .	112
6.4	Inversion attack $F_1$ score for different epochs . . . . .	117
6.5	Inversion attack accuracy metrics for different $\beta$ for TREC dataset . .	119
6.6	The correlation of search accuracy and the attack efficiency with $\beta$ . .	121

# Acronyms

<b><math>k</math>NN</b>	.....	$k$ -nearest-neighbor
<b>AES</b>	.....	Advanced Encryption Standard
<b>AP</b>	.....	Access Pattern
<b>API</b>	.....	Application Programming Interface
<b>ASPE</b>	.....	Asymmetric Scalar-Product-Preserving Encryption
<b>AWS</b>	.....	Amazon Web Services
<b>BERT</b>	.....	Bidirectional Encoder Representation from Transformer
<b>BRC</b>	.....	Best Range Cover
<b>CBC</b>	.....	Cipher Block Chaining
<b>CCA</b>	.....	Chosen Ciphertext Attack
<b>CDF</b>	.....	Cumulative Distribution Function
<b>CDP-ODB</b>	.....	Computationally DP Outsourced Database System
<b>CPA</b>	.....	Chosen Plaintext Attack
<b>CPU</b>	.....	Central Processing Unit
<b>CTR</b>	.....	Counter
<b>CV</b>	.....	Communication Volume
<b>DCPE</b>	.....	Distance Comparison Preserving Encryption
<b>DP</b>	.....	Differential Privacy
<b>EMP-toolkit</b>	.....	Efficient MultiParty Computation Toolkit
<b>FAISS</b>	.....	Facebook AI Similarity Search
<b>FPGA</b>	.....	Field-Programmable Gate Array
<b>GCM</b>	.....	Galois / Counter
<b>GCP</b>	.....	Google Cloud Platform
<b>GPU</b>	.....	Graphics Processing Unit
<b>HG</b>	.....	Hypergeometric Probability Distribution
<b>HIPAA</b>	.....	Health Insurance Portability and Accountability Act
<b>I/O</b>	.....	Input / Output
<b>ID</b>	.....	Identifier
<b>IND-CCA</b>	.....	Indistinguishability under Chosen Ciphertext Attack
<b>IND-CPA</b>	.....	Indistinguishability under Chosen Plaintext Attack
<b>IV</b>	.....	Initialization Vector
<b>KVS</b>	.....	Key-Value Store
<b>LPA</b>	.....	Laplacian Perturbation Algorithm
<b>LSTM</b>	.....	Long Short-Term Memory

<b>ML</b>	.....	Machine Learning
<b>MLC</b>	.....	Multi-Label Classification
<b>MRR</b>	.....	Mean Reciprocal Rank
<b>MSP</b>	.....	Multi-Set Prediction
<b>nDCG</b>	.....	Normalized Discounted Cumulative Gain
<b>NIST</b>	.....	National Institute of Standards and Technology
<b>NLP</b>	.....	Natural Language Processing
<b>OPE</b>	.....	Order-Preserving Encryption
<b>ORAM</b>	.....	Oblivious Random Access Machine
<b>ORE</b>	.....	Order-Revealing Encryption
<b>OS</b>	.....	Operating System
<b>PBC</b>	.....	Pairing-Based Crypto
<b>PPH</b>	.....	Property-Preserving Hash
<b>PRF</b>	.....	Pseudo-Random Function
<b>PRG</b>	.....	Pseudo-Random Generator
<b>PRP</b>	.....	Pseudo-Random Permutation
<b>RAM</b>	.....	Random Access Memory
<b>RDBMS</b>	.....	Relational Database Management System
<b>SGX</b>	.....	Software Guard Extensions
<b>SHA</b>	.....	Secure Hash Algorithm
<b>SQL</b>	.....	Structured Query Language
<b>SSD</b>	.....	Solid State Drive
<b>SSE</b>	.....	Searchable Symmetric Encryption
<b>TEE</b>	.....	Trusted Execution Environment
<b>TREC</b>	.....	Text Retrieval Conference
<b>VM</b>	.....	Virtual Machine
<b>XEX</b>	.....	XOR-Encrypt-XOR
<b>XML</b>	.....	Extensible Markup Language
<b>XTS</b>	.....	XEX-based tweaked-codebook with ciphertext stealing

## Chapter 1

# Introduction

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 **VMs** 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 obtains 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 thesis, we 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 Model

In this work, we consider *an outsourced database system* model, a notion first introduced in [HILM02], adapted from [KKNO16] and [Bog+21].

### 1.1.1 Outsourced database model

Similar to [Bog+21], a database is abstracted as a collection of  $n$  records  $r$ , each with a unique identifier  $r^{\text{ID}}$ , associated with search keys  $\text{SK}$ :  $\mathcal{D} = \{(r_1, r_1^{\text{ID}}, \text{SK}_1), \dots, (r_n, r_n^{\text{ID}}, \text{SK}_n)\}$ . 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  $q \in \mathcal{Q} : \mathcal{X} \rightarrow \{0, 1\}$ . Evaluating a query  $q$  on a database  $\mathcal{D}$  results in  $q(\mathcal{D}) = \{r_i : q(\text{SK}_i) = 1\}$ , 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  $\Pi_{\text{setup}}$ ,  $\mathcal{U}$  receives as input a database  $\mathcal{D} = \{(r_1, r_1^{\text{ID}}, \text{SK}_1), \dots, (r_n, r_n^{\text{ID}}, \text{SK}_n)\}$  and  $\mathcal{S}$  may optionally output a data structure  $\mathcal{DS}$ . In query protocol  $\Pi_{\text{query}}$ ,  $\mathcal{U}$  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  $\Pi_{\text{setup}}$  and  $\Pi_{\text{query}}$  on the corresponding inputs yields the correct result  $\{r_i : q(\text{SK}_i) = 1\}$ .

### 1.1.2 Security model

In an outsourced database system we are not aiming for perfect security, because the stronger the data protection guarantees are the harder it is to manipulate these data, the less efficient and functional the system becomes. There will always be some leakage, and our goal is to quantify, analyze and reduce it, while retaining the system’s performance and usability. In terms of security models, we define two types of adversaries — a *snapshot* and a *persistent* adversary.

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 the same information as the snapshot adversary plus the network traffic and the access pattern. Such adversary can be thought of as a 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.



Note that both adversaries are *honest-but-curious* — they only observe and never interfere. Denial-of-service attacks and integrity protection are out of scope of this work.

### 1.1.3 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, we 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 color = 'blue'.
```

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

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

**A  $k$ NN 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.
```

## 1.2 Thesis structure

In Chapter 2, we will cover the building blocks that are used in the outsourced database systems. These blocks include symmetric encryption, **ORAM** and **Differential Privacy**. Chapter 3 includes the overview of approaches that provide the privacy and/or security in the outsourced setting. The chapter also includes the overview of the attacks against the mechanisms. Chapter 4 analyses in detail the range query mechanisms in the snapshot adversary model. The chapter offers a comparative evaluation of five **Order-Revealing Encryption** schemes and five secure range query protocols [BKR19]. Chapter 5 proposes a novel solution for the range queries in the persistent adversary model,  $\mathcal{E}$ psolute [BKR19]. In Chapter 6, we describe *k-anon* [BKOZ22] — a mechanism to answer similarity search (i.e., *k-nearest-neighbor*) queries in a snapshot adversary setting using a type of property-preserving encryption similar to **OPE**. In the chapter we describe the encryption method, the set of experiments to empirically measure the search accuracy and the level of protection against the attacks. The source code of this thesis is publicly available.<sup>1</sup>

### 1.2.1 Works completed during the Ph.D. program

[BKR19] **Dmytro Bogatov**, George Kollios, and Leonid 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](https://doi.org/10.14778/3324301.3324309)

[Bog+21] **Dmytro Bogatov**, Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O’Neill. “ $\mathcal{E}$ psolute: Efficiently Querying Databases While Providing Differential Privacy”. In: *Proceedings of the 2021 ACM SIG-*

---

<sup>1</sup>The thesis has been compiled from [dbogatov/doctoral-thesis00796b906](https://github.com/dbogatov/doctoral-thesis00796b906) on September 15, 2022.

*SAC Conference on Computer and Communications Security — CCS '2021*. 2021. DOI: [10.1145/3460120.3484786](https://doi.org/10.1145/3460120.3484786)

- [BKOZ22]** **Dmytro Bogatov**, George Kollios, Adam O’Neill, and Hamed Zamani. “*k-anon*: Secure Similarity Search in Outsourced Databases”. Apr. 2022
- [BCET21]** **Dmytro Bogatov**, Angelo De Caro, Kaoutar Elkhiyaoui, and Björn Tackmann. “Anonymous Transactions with Revocation and Auditing in Hyperledger Fabric”. In: *International Conference on Cryptology and Network Security*. Springer. 2021. DOI: [10.1007/978-3-030-92548-2\\_23](https://doi.org/10.1007/978-3-030-92548-2_23)
- [NBK19]** Oleksandr Narykov, **Dmytro Bogatov**, and Dmitry Korkin. “DISPOT: a simple knowledge-based protein domain interaction statistical potential”. In: *Bioinformatics* 35.24 (July 2019), pp. 5374–5378. DOI: [10.1093/bioinformatics/btz587](https://doi.org/10.1093/bioinformatics/btz587)

The works [\[BKR19\]](#); [Bog+21](#); [BKOZ22](#) are discussed in Chapters 4 to 6 respectively. The other two works [\[BCET21\]](#); [NBK19](#) fall outside of scope of this thesis. [\[BCET21\]](#) proposes an improved anonymous delegatable credential scheme and its novel extensions, auditability and revocation, along with the instantiation and comprehensible set of experiments, see the abstract in Appendix A. [\[NBK19\]](#) is from the domain of Bioinformatics, it offers a knowledge-based statistical potential that estimates the propensity of an interaction between a pair of protein domains, see the abstract in Appendix B. I also want to note the works I have completed on the route to my doctorate program [\[Bog17\]](#); [BH16](#); [NBXO15](#)].

## Chapter 2

# Background

In this section, we will go over the building blocks required to construct the outsourced database systems and their components that we will discuss in the next chapters. These prerequisites include the symmetric encryption, ORAMs and PathORAM [Ste+13] in particular, Differential Privacy and DP sanitizers, and finally, Trusted Execution Environments. *Some of the following sections were paraphrased or taken verbatim from my published work [BKR19; Bog+21].*

### 2.1 Symmetric encryption

Symmetric encryption scheme is a tuple of algorithms  $E = \{\text{KEYGEN}, \text{ENC}, \text{DEC}\}$  with the following properties.  $E.\text{KEYGEN}(1^\lambda) \rightarrow k$  is a *randomized* algorithm that on a security parameter  $1^\lambda$  returns a key that will be used for both encryption and decryption.  $E.\text{ENC}(m) \rightarrow c$  is a *randomized* algorithm that on a plaintext message  $m \in \{0, 1\}^*$  produces its ciphertext  $c \in \{0, 1\}^*$ .  $E.\text{DEC}(c) \rightarrow m$  is a *deterministic* algorithm that on a ciphertext  $c \in \{0, 1\}^*$  produces its original plaintext message  $m \in \{0, 1\}^*$ .

#### 2.1.1 Security

The security of the symmetric encryption is typically defined as the indistinguishability under a certain attack. The definition is structured around the game between

the challenger and the adversary  $\mathcal{A}$ . The challenger fixes one of the two “worlds”, left or right, and the adversary wins the game if she can reliably tell which world it was.

A weaker security definition, **Indistinguishability under Chosen Plaintext Attack (IND-CPA)**, intuitively, requires that the ciphertext leaks nothing about the plaintext. To formalize the requirement, the adversary can give the challenger a set of plaintext pairs to encrypt, and the challenger responds with a set of ciphertexts where the left or the right part was encrypted. The adversary can then use any (polynomial-time) algorithm over the ciphertexts and make a guess of whether the left or the right part was encrypted. The claim is, if there is anything that the ciphertext leaks about the plaintext, there exists an adversary who will win. The security claim is therefore contrapositive — the scheme is **IND-CPA** secure iff there is no such adversary that wins the game.

**IND-CPA** security is not by itself sufficient since it does not account for the decryption part of the scheme. There are known attacks that decrypt the plaintext (i.e., defeat the encryption) if the adversary can trigger the decryption and observe the process or the result (see padding attack [Vau02] and XML encryption attack [JS11]).

The stronger definition, **Indistinguishability under Chosen Ciphertext Attack (IND-CCA)**, captures the decryption component. It extends the **IND-CPA** game in that the adversary can now request the challenger to decrypt *any* ciphertext of her choice *except* the ones that the challenger himself encrypted for the adversary. The adversary still outputs a guess of the two worlds and wins if reliably guesses correctly. Note that in this game if the decryption can fail for any reason, or even if  $\mathcal{A}$  can observe any difference in execution for different inputs, the scheme is insecure. Therefore, **IND-CCA** immediately rules out the aforementioned attacks [Vau02; JS11].

### 2.1.2 Components

Note that for the practical purposes we define the encryption algorithm as randomized — producing different ciphertext for the same plaintext on every invocation. While this is how the symmetric encryption scheme is used in applications, formally, producing the deterministic ciphertext and randomizing it are different operations.

The randomness is produced independently, typically using a **Pseudo-Random Generator (PRG)**, and is used for both secrecy and integrity (which is necessary for **CCA** security). After obtaining the random bits, the algorithm repeatedly uses the block cipher (formally, a **Pseudo-Random Permutation (PRP)**), with the number of invocations linear in the message length. How the randomness and the message are combined is defined by the *mode of operation* and differs from one encryption scheme to another. Typically, the randomness comes in a form of a block, an **Initialization Vector (IV)**, filled with random bits. The mode of operation then defines how the blocks and the **IV** are combined together.

In practical systems, the ciphertext is then broken up into components, like the ciphertext material itself, the **IV** (varies by the scheme), the version of the key, etc. Also note that the encryption scheme key has a maximum number of times it can be used for encryption (its *operational lifetime*).

When it comes to the real-world encryption systems, we use standardized primitives — a block cipher (**PRP**), a **Pseudo-Random Generator** and a mode of operation. **Advanced Encryption Standard (AES)** [Dwo+01] is a NIST-standardized block cipher, which operates on 128-bit blocks. NIST also offers recommendations for random number generator mechanisms [BK15], constructions [BK16] and sources of entropy [Tur+18]. Lastly, some of the commonly used modes of operation are **CBC** and **CTR** [Dwo01] modes for general-purpose encryption, **GCM** [Dwo07] for an authenticated

encryption and XTS [HH19] for encryption of data on block-oriented devices (e.g., disks).

## 2.2 Oblivious Random Access Machine

Informally, Oblivious Random Access Machine (ORAM) is a mechanism that lets the users hide their access pattern to remote storage. An adversarial server can monitor the actual accessed locations, but she cannot tell a read from a write, the content of the block or even whether the same logical location is being referenced. The notion was first defined by Goldreich [Gol87] and Goldreich and Ostrovsky [GO96].

More formally, a  $(\eta_1, \eta_2)$ -ORAM protocol is a two-party protocol between a client  $\mathcal{C}$  and a server  $\mathcal{S}$  who maintains the storage in a form of array of blocks. In each round, the client  $\mathcal{C}$  has input  $(o, a, d)$ , where  $o$  is an access type (**r** or **w**),  $a$  is a storage block address and  $d$  is a new data value, or  $\perp$  for read operation. The input of  $\mathcal{S}$  is the current storage array. Via the protocol, the server updates the storage or returns to  $\mathcal{U}$  the data stored at the requested block, respectively. We speak of a sequence of such operations as a program  $\mathbf{y}$  being *executed under the ORAM*.

An ORAM protocol must satisfy correctness and security. Correctness requires that  $\mathcal{C}$  obtains the correct output of the computation except with at most probability  $\eta_1$ . For security, we require that for every client  $\mathcal{C}$  there exists a simulator  $\text{SIM}_{\text{ORAM}}$  which provides a simulation of the server’s view in the above experiment given only the number of operations. That is, the output distribution of  $\text{SIM}_{\text{ORAM}}(c)$  is indistinguishable from  $\text{VIEW}_{\mathcal{S}}$  with probability at most  $\eta_2$  after  $c$  protocol rounds. Note that the ORAM protocols typically differ in the way the storage is organized and manipulated, but are similar in that the records or blocks are symmetrically encrypted (see Section 2.1).

ORAM protocols are generally stateful, after each execution the client and server states are updated. *For brevity, throughout the thesis we will assume the ORAM state updates are implicit, including the encryption key generated and maintained by the client.*

Some existing efficient ORAM protocols are Square Root ORAM [Gol87], Hierarchical ORAM [GO96], Binary-Tree ORAM [SCSL11], Interleave Buffer Shuffle Square Root ORAM [Xie+16], TP-ORAM [SSS12], PathORAM [Ste+13] and TaORAM [Sah+16]. For detailed descriptions of each protocol, we recommend the work of Chang, Xie, and Li [CXL16]. The latter three ORAMs achieve the lowest communication and storage overheads,  $\mathcal{O}(\log n)$  and  $\mathcal{O}(n)$ , respectively.

### 2.2.1 PathORAM

PathORAM [Ste+13] is one of the most commonly used ORAM protocols due to its efficiency and simplicity. In this section we briefly describe this construction as it is used as an ORAM instantiation in the rest of the thesis.

In the PathORAM, both the client  $\mathcal{C}$  and the server  $\mathcal{S}$  are stateful. The server stores the encrypted records (blocks) grouped in buckets, and the buckets form a binary tree. The client’s storage, although asymptotically linear in the data size, is relatively small in practice. The client stores the *position map* that maps the record ID to a leaf in the server tree storage and a small amount of stash, which can store some plaintext blocks on the client side.

The main invariant of the protocol is that at all times the ciphertext of the record  $a$  is stored somewhere on the *path* from the root to the leaf that is mapped to  $a$ , or in the stash (hence the name of the construction).

The ORAM is initialized with a binary tree of buckets with all buckets containing valid encryptions of dummy records. The position map is sampled at random (filled with permuted distinct numbers).



Main routine of the **ORAM** is an access sub-protocol, which is similar for read  $\mathbf{r}$  and write  $\mathbf{w}$  types of access (remember, **ORAM** hides the type of access from the curious server). In PathORAM, the access  $(o, a, d)$  consists of four steps. First, remap the current leaf  $x$  for  $a$  to a new random leaf  $x'$ . Second, read the entire path to leaf  $x$  (all buckets from root to leaf) into the client stash. Third, the client updates the block value to  $d$  if the access is a write  $\mathbf{w}$ . Finally, write back the path to leaf  $x$  filling the buckets with all blocks from stash in a way that maintains the invariant.

The newly updated block  $a$  with the new value  $d$  may be included in the new path, or it may stay in the stash. It is important that the stash size be provably bounded. [Ste+13, Theorem 1] does exactly that — at least for the bucket size of 5, the probability of stash overflow and its size are related as in Equation (2.1).

$$\Pr[\text{stash size} > x] \leq 14 \cdot (0.6002)^x \tag{2.1}$$

If the probability of a protocol failure is thought of as the adversary’s advantage (the probability of breaking the security), then the stash size equivalent to 128-bit security is about 100 blocks for the bucket of size 5 [Ste+13, Figure 5].

## 2.3 Differential Privacy

**Differential Privacy (DP)** is a guarantee on a mechanism that takes a dataset 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, **Differential Privacy** is defined in Definition 2.3.1.

**Definition 2.3.1** (Differential Privacy, adapted from [Dwo+06; DMNS06]). *A randomized algorithm  $A$  is  $(\epsilon, \delta)$ -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  $A$ ,

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

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 2.3.1 then bounds the advantage of  $\mathcal{A}$  with  $\epsilon$  and  $\delta$  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  $\epsilon$  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,  $\epsilon$  is called a *privacy budget* of a DP system.  $\delta$  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 randomized and fails with a certain chance, this probability will be  $\delta$ . 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 system 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  $\delta$ .

Note that Definition 2.3.1 describes a property of  $A$  and not a construction method. To construct  $A$ , the seminal work of Dwork et al. [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  $\text{LAPLACE}\left(0, \frac{\text{sensitivity}}{\epsilon}\right)$  to the real result of a query, the resulting mechanism is  $\epsilon$ -DP.

### 2.3.1 DP sanitizers

While the **Laplacian Perturbation Algorithm** is an effective and simple way of answering a single count query, we will need to answer a sequence of count queries, ideally, without imposing a bound on the length of this sequence. We will hence make use of *sanitization* algorithms.

**Definition 2.3.2.** *Let  $\mathcal{Q}$  be a collection of queries. An  $(\epsilon, \delta, \alpha, \beta)$ -differentially private sanitizer for  $\mathcal{Q}$  is a pair of algorithms  $(A, B)$  such that:*

- *A is  $(\epsilon, \delta)$ -differentially private, and*
- *on input a dataset  $\mathcal{D} = d_1, \dots, d_n \in \mathcal{X}^n$ , A outputs a data structure  $\mathcal{DS}$  such that with probability  $1 - \beta$  for all  $q \in \mathcal{Q}$ ,  $|B(\mathcal{DS}, q) - \sum_i q(d_i)| \leq \alpha$ .*

In Definition 2.3.2, the query is a predicate, which is defined as in Section 1.1.1, and it returns a binary 0 or 1 when executed over a search key (an attribute)  $d \in \mathcal{X}$ .  $B(\mathcal{DS}, q)$  then returns a count, a scalar value which bounds the number of search keys that satisfy the query. For example, a query  $q$  can be a range query, then  $\sum_i q(d_i)$  is the number of records in the range, and  $B(\mathcal{DS}, q)$  returns a scalar value of noise (the fake records), such that it is within  $\alpha$  from the true count.

**Remark 2.3.1.** *Given an  $(\epsilon, \delta, \alpha, \beta)$ -DP sanitizer as in Definition 2.3.2 one can replace the answer  $B(\mathcal{DS}, q)$  with  $B'(\mathcal{DS}, q) = B(\mathcal{DS}, q) + \alpha$ . Hence, with probability  $1 - \beta$ , for all  $q \in \mathcal{Q}$ ,  $0 \leq B'(\mathcal{DS}, q) - \sum_i q(d_i) \leq 2\alpha$ . We will hence assume from now on that sanitizers have this latter guarantee on their error.*

The main idea of *sanitization* (a.k.a. private data release) is to release specific noisy statistics on a private dataset once, which can then be combined in order to answer an arbitrary number of queries without violating privacy. Depending on the query type (see Section 1.1.3) and the notion of **Differential Privacy** (i.e., pure or approximate), different upper bounds on the error have been proven. Omitting the dependency on  $\epsilon$  and  $\delta$ , in case of point queries over domain size  $N$ , pure **Differential**

Privacy results in  $\alpha = \Theta(\log N)$  [BBKN14], while for approximate Differential Privacy  $\alpha = \mathcal{O}(1)$  [BNS13]. Note that the sanitizers add noise to a count of records, and for the point queries the count is the number of records with a given categorical value (i.e., number of female students in a class). For range queries over domain size  $N$ , these bounds are  $\alpha = \Theta(\log N)$  for pure Differential Privacy [BLR13; DNPR10], and  $\alpha = \mathcal{O}((\log^* N)^{1.5})$  for approximate Differential Privacy (with an almost matching lower bound of  $\alpha = \Omega(\log^* N)$ ) [BNS13; BNSV15; Kap+20]. More generally, Blum, Ligett, and Roth [BLR13] showed that any finite query set  $\mathcal{Q}$  can be sanitized, albeit non-efficiently.

### Answering point and range queries with differential privacy

Utilizing the LPA for answering point queries results in error  $\alpha = \mathcal{O}(\log N)$ . A practical solution for answering range queries with error bounds very close to the optimal ones is the hierarchical method [DNPR10; HRMS10; XWG10]. The main idea is to build an aggregate tree on the domain, and add noise to each node proportional to the tree height (i.e., noise scale logarithmic in the domain size  $N$ ). Then, every range query is answered using the minimum number of tree nodes. Qardaji, Yang, and Li [QYL13] showed that the hierarchical algorithm of Hay et al. [HRMS10], when combined with their proposed optimizations, offers the lowest error.

### Composition

Finally, in this thesis we will make use of a *composition* theorem (adapted from [McS09]) based on [DMNS06; Dwo+06]. It concerns executions of multiple DP mechanisms on non-disjoint and disjoint inputs.

**Theorem 2.3.1.** *Let  $A_1, \dots, A_r$  be mechanisms, such that each  $A_i$  provides  $\epsilon_i$ -DP. Let  $\mathcal{D}_1, \dots, \mathcal{D}_r$  be pairwise non-disjoint (resp., disjoint) datasets. Let  $A$  be another mechanism that executes  $A_1(\mathcal{D}_1), \dots, A_r(\mathcal{D}_r)$  using independent randomness for each*

$A_i$ , and returns their outputs. Then, mechanism  $A$  is  $(\sum_{i=1}^r \epsilon_i)$ -DP (resp.,  $(\max_{i=1}^r \epsilon_i)$ -DP).

## 2.4 Trusted Execution Environments

Trusted Execution Environments is a generalized term for a “protected” part of a processing engine. Security in this setting means a combination of confidentiality, integrity, secrecy, isolation and protection against side-channel attacks. TEE cannot be entered through system calls, jumps or register manipulations. Environment’s memory content and integrity are protected, and neither OS, nor a hypervisor can access it. Main purpose of the TEE is to vastly reduce the attack surface, see Figure 2.1.

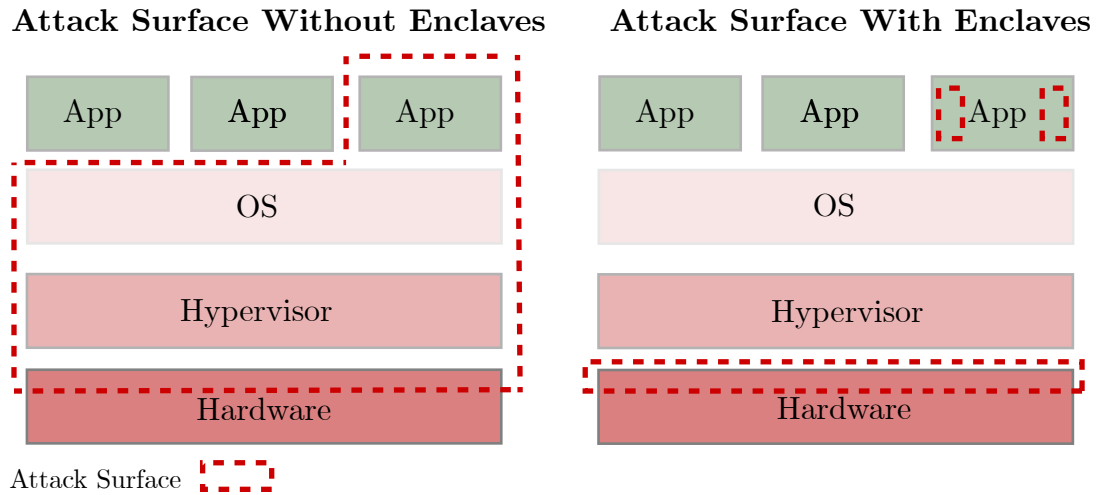


Figure 2.1: Attack surface with TEE. Adapted from [sgx101.gitbook.io](https://sgx101.gitbook.io).

While TEE is a concept defining an execution environment, specific solutions include a hardware security module (a plug-in device with a protected memory and a crypto-optimized processing unit), an FPGA, and a set of extensions to an existing processor architecture, such as an SGX for Intel x86, Secure Encrypted Virtualization

[Dav16] for AMD and Apple’s Secure Enclave<sup>1</sup>. Of all these, Intel SGX is the most widely used technology.

### 2.4.1 Software Guard Extensions

Software Guard Extensions (SGX) is a set of instructions for the Intel x86 architecture that allow a user or an Operating System to define a region of protected memory, called the *enclave*, and interact with it. The enclave can only be accessed using the SGX instructions (i.e, regular mov instruction would not work), and all pages of the enclave are symmetrically encrypted and physically protected. SGX guarantees the integrity and security of the memory pages within the enclave. Although the size of enclave memory is very limited, SGX can use regular RAM by transparently swapping the pages between the trusted and untrusted memory. The pages are then encrypted with integrity protection when placed in the RAM (*sealing* in SGX terms).

An SGX-enabled application declares its trusted and untrusted components upfront. Trusted part will live entirely in the enclave, while the untrusted part is a normal process that runs within the OS. The application has to be digitally signed for the enclave to accept it, and the enclave itself can authenticate to the user via an *attestation* process. Conceptually, the simplest SGX application in an outsourced database system can be seen as a trusted component that operates over sensitive material (e.g., keys, tokens, plaintext user data), a remote trusted client application that communicates with the enclave, and a layer of code that passes requests through (untrusted part of an SGX application). Cipherbase [Ara+13] and StealthDB [VGG19] are good examples of such approach.

---

<sup>1</sup><https://support.apple.com/guide/security/secure-enclave-sec59b0b31ff/web>

### 2.4.2 Issues with SGX

SGX, as the closest instantiation of TEE available, has been extensively targeted. The attacks include Foreshadow [Van+18], Prime+Probe cache attack [Sch+17], an attack from within the enclave [SWG19], Spectre line of attacks that can bypass SGX [Koc+19], replay attack [Ska+19], Plundervolt attack [Mur+20], Load Value Injection attack [Van+20] and SGXaxe attack [SKG20]. Due to its many security issues, SGX has been officially discontinued.<sup>2</sup>

Besides the attacks against the SGX, the design itself has a number of restrictions. First of all, the enclave memory is capped at 128 MB, part of which is occupied by the SGX control structures, leaving the application about 96 MB. SGX allows the use of external memory pages through the sealing mechanism, but it imposes high overhead of re-encryption and crossing the enclave physical boundary. Second, the code execution inside the enclave is significantly slower. Third, by design, only the untrusted application component can interact with the OS, for example, make network or storage I/O requests. Finally, from the security standpoint the enclave is vulnerable against the side-channel attacks, most of all, access pattern leakage. Such leakage implies that the normal database application cannot be placed directly in the enclave and be deemed secure, because access pattern has been effectively exploited up to a reconstruction attack [KKNO16]. One then has to design the application specifically to conceal the access pattern. For example, ZeroTrace [SGF18] is a variant of PathORAM [Ste+13] that is internally oblivious and thus can work in SGX. Oblix [Mis+18] is another example of a structure that is, in Mishra et al. [Mis+18] own terms, doubly-oblivious — internally in how it manages memory and registers, and externally in how it interacts with a storage.

---

<sup>2</sup>As of 11<sup>th</sup> Generation Intel Core processor.

## Chapter 3

### Related work

In this section we will go over the relevant work in the area. The discussion is grouped by the query and adversary type, consistent with the overall thesis structure. *Some of the following sections were paraphrased or taken verbatim from my published work [BKR19; Bog+21].*

#### 3.1 Range query security in a snapshot model

In Chapter 4 we explore the methods of answering range queries in a snapshot adversary setting. We group these methods into those relying on OPE / ORE schemes coupled with regular database range indices, such as B<sup>+</sup> tree, and other mostly interactive constructions that use custom data structures to support secure indexing.

The list of recently proposed Order-Preserving Encryption (OPE) schemes includes [AKSX04; OSC04; BCLO09; BCO11; TYM14; KS14; Woz+13; KAK10; Ker15; KAK13; XYH12; XY12; LW12; EWSG04; LW13; Zha+22; SYSC21; YK21]. Popa, Li, and Zeldovich [PLZ13] present a nice analysis of these schemes and they are the first to show that using a stateful scheme one can achieve the ideal security guarantees for OPE. In addition, there are a number of Order-Revealing Encryption (ORE) schemes [Bon+15; CLWW16; LW16; Cas+18; Cas+18; BZ16; Haa+17; ELL18; LWZ19; Lv+21] and related protocols [ZRL21] that have been proposed. After the introduction of OPE / ORE as the means of protecting range indices, a line of attacks [CGPR15; DDC16; NKW15; LWHZ21] have emerged.



The second group of approaches assumes an outsourced setting where the client may have to communicate with the server during query processing [RACY16; KT19; BPP17; Dem+16]. We would like to point out that there are some other methods that can be used to run range queries on encrypted data that use different types of schemes and techniques. CryptDB [PRZB11] is a seminal work in this direction offering computations over encrypted data. Arx [PBP19] provides strictly stronger security guarantees by using only semantically secure primitives. Seabed [Pap+16] uses an additively symmetric homomorphic encryption scheme for aggregates and certain filter queries. Samanthula, Jiang, and Bertino [SJB14] offer a method to verify and apply a predicate (a junction of conditions) using garbled circuits or homomorphic encryption without revealing the predicate itself. SisoSPIR [IKLO16] presents a mechanism to build an oblivious index tree such that neither party learns the pass taken. See [BPP17] and [LCZ17] for an overview of other methods. We also note a work on theoretical analysis of ORE security [JPS21].

### 3.2 Range query security in a persistent model

For the persistent adversary setting, we group the related secure databases, engines, and indices into two categories (i) systems that are oblivious or volume-hiding and do not require **Trusted Execution Environment (TEE)**, and (ii) constructions that rely on **TEE** (usually, Intel **SGX**). In this section, where relevant, we aim to compare the related work to our own point and range query solution,  $\mathcal{E}$ psolute [Bog+21], see Chapter 5. *We claim that  $\mathcal{E}$ psolute is the most secure and practical range- and point-query engine in the outsourced database model, that protects both **Access Pattern (AP)** and **Communication Volume (CV)** using **Differential Privacy**, while not relying on **TEE**, linear scan or padding result size to the maximum.*

### 3.2.1 Obliviousness and volume-hiding without enclave

This category is the most relevant to  $\mathcal{E}_{\text{psolute}}$ , wherein the systems provide either or both **AP** and **CV** protection without relying on **TEE**. **Crypte** [Roy+20] is a recent end-to-end system executing “**DP** programs”. **Crypte** has a different model than  $\mathcal{E}_{\text{psolute}}$  in that it assumes two non-colluding servers, an adversarial querying user (the analyst), and it uses **DP** to protect the privacy of an individual in the database, which includes volume-hiding for aggregate queries. **Crypte** also does not consider oblivious execution and attacks against the **AP**. **Shrinkwrap** [Bat+18] (and its predecessor **SMCQL** [Bat+17]) is an excellent system designed for complex queries over federated and distributed data sources. In **Shrinkwrap**, **AP** protection is achieved by using oblivious operators (linear scan and sort) and **CV** is concealed by adding fake records to intermediate results with **DP**. Padding the result to the maximum size first and doing a linear scan over it afterwards to “shrink” it using **DP**, is much more expensive than in  $\mathcal{E}_{\text{psolute}}$ , however. In addition, in processing a query, the worker nodes are performing an  $O(n \log n)$  cost oblivious sorting, where  $n$  is the maximum result size (whole table for range query), since they are designed to answer more general complex queries. **SEAL** [DPPS20] offers adjustable **AP** and **CV** leakages, up to specific bits of leakage. **SEAL** builds on top of **Logarithmic-SRC** [Dem+16], splits storage into multiple **ORAMs** to adjust **AP**, and pads results size to a power of 2 to adjust **CV**.  $\mathcal{E}_{\text{psolute}}$ , on the other hand, fully hides the **AP** and uses **DP** with its guarantees to pad the result size. **PINED-RQ** [Sah+18] samples Laplacian noise right in the **B+** tree index tree, adding fake and removing real pointers according to the sample. Unlike  $\mathcal{E}_{\text{psolute}}$ , **PINED-RQ** allows false negatives (i.e., result records not included in the answer), and does not protect against **AP** leakage. On the theoretical side, Chan et al. [CCMS19] (followed by Beimel, Nissim, and Zaheri [BNZ19]) treat the **AP** itself as something to protect with **DP**. [CCMS19] introduces a notion of differential

obliviousness that is admittedly weaker than the full obliviousness used in  $\mathcal{E}$ psolute. Most importantly, [CCMS19] ensures differential privacy with respect to the ORAM only, while  $\mathcal{E}$ psolute ensures DP with respect to the entire view of the adversary.

### 3.2.2 Enclave-based solutions

Works in this category use trusted execution environment (usually, SGX enclave). These works are primarily concerned with the AP protection for both trusted and untrusted memory, unlike  $\mathcal{E}$ psolute which also protects CV. Cipherbase [Ara+13] was a pioneer introducing the idea of using TEE (FPGA at that time) to assist with RDBMS security. HardIDX [Fuh+17] simply puts the B+ tree in the enclave, while StealthDB [VGG19] symmetrically encrypts all records and brings them in the enclave one at a time for processing. EnclaveDB [PVC18] assumes somewhat unrealistic 192 GB enclave and puts the entire database in it. ObliDB [EZ19] and Opaque [Zhe+17] assume fully oblivious enclave memory (not available as of today) and devise algorithms that use this fully trusted portion to obliviously execute common RDBMS operators, like filters and joins. Oblix [Mis+18] provides a multimap that is oblivious both in and out of the enclave. HybrIDX claims protection against both AP and CV leakages, but unlike  $\mathcal{E}$ psolute it only obfuscates them.  $\mathcal{E}$ psolute offers an indistinguishability guarantee for AP and a DP guarantee for CV, while HybrIDX hides the exact result size and only obfuscates the AP. Lastly, Hermetic [XPHF19] takes on the SGX side-channel attacks, including AP. It provides oblivious primitives, however, it only offers protection against software and not physical attacks (e.g., it trusts a hypervisor to disable interrupts).

## 3.3 $k$ NN query security in a snapshot model

In this section, where relevant, we aim to compare the related work to our own  $k$ NN query solution, *k-anon* [BKOZ22], see Chapter 6.

A work immediately related to ours is QuickN by Wang, Hou, and Li [WHL20]. QuickN offers an adaptation of nearest-neighbor search algorithm in conventional tree data structures (i.e., R-trees) to well-established Order-Preserving Encryption (OPE) schemes. Unlike our solution that involves a novel property-preserving encryption scheme specifically designed for high-dimensional vectors, QuickN encrypts each vector dimension separately with OPE. [WHL20] includes the experiments with attacks against their solution (attack by Grubbs et al. [Gru+17]) and report a high degree of protection against them (at most 3.7% matching rate for the inference attacks against coordinates represented as pairs of longitude and latitude).

QuickN approach of applying OPE to an R-tree, however, has some disadvantages. First, an ideal stateless OPE has been shown inferior ([BCO11]) to its counterpart, an Order-Revealing Encryption (ORE) in which the comparison over ciphertexts is defined explicitly.<sup>1</sup> An ORE, in turn, can have a varying level of security, with the higher security level translating into lower comparison performance [BKR19]. In QuickN, an R-tree-based nearest-neighbor algorithm involves a very high number of comparisons, linear in data dimensionality. With the cost of comparison no longer negligible, the overhead of a query over 2D or 3D is already high, saying nothing of 768-dimensional vectors that our work targets. Second, QuickN protocol is not single-round (i.e., it takes two roundtrips per query) and it returns a large number of false positives even for a minimal  $k$  (4 000 false positives for  $10^6$  dataset and  $k = 1$ ).

Wong et al. [WCKM09] propose a novel scheme, ASPE, that preserves a special type of scalar product. Namely, Asymmetric Scalar-Product-Preserving Encryption (ASPE) scheme preserves the scalar product of a database point and a query point, but not the product of a database point with itself or another database point. ASPE is then naturally integrated in existing  $k$ NN algorithms that use such asymmetric

---

<sup>1</sup>[WHL20] uses mOPE [PLZ13] which is an interactive protocol and not a traditional lightweight stateless OPE like [BCLO09]. Since mOPE is an ideal (though stateful) OPE, Wang, Hou, and Li [WHL20] do not include OPE leakage in their security definition.

scalar product in their indices. The work of Wong et al. [WCKM09] is similar to *k-anon* in that we also apply a property-preserving encryption scheme to existing *k*NN algorithms. ASPE is different in that it preserves a scalar product while we preserve an  $L_2$  distance comparison, and ASPE has been broken in [Y LX13] (although the attack is a chosen plaintext attack, i.e., one cannot decrypt a random ciphertext).

Other works either target different aspects of query security, like integrity and soundness of results [Yu+21; Cui+20], or involve mechanisms other than property-preserving encryption [LLLT19; YPBV16; ZXT13; LSP15; ESJ14; YPBV14; KSS13; HXRC11; PBP10; Ghi+08; QA08; MCA07].

## Chapter 4

# Range queries in the snapshot model

Order-Preserving Encryption (OPE) and Order-Revealing Encryption (ORE) are two important encryption schemes that have been proposed in the area of query evaluation over encrypted data. These schemes can provide very efficient query execution but at the same time may leak some information to adversaries. In this chapter, we present the first comprehensive comparison among a number of important OPE and ORE schemes using a framework that we developed. We evaluate protocols that are based on these schemes as well. We analyze and compare them both theoretically and experimentally and measure their performance over database indexing and query evaluation techniques using not only execution time but also I/O performance and usage of cryptographic primitive operations. Our comparison reveals some interesting insights concerning the relative security and performance of these approaches in database settings. Furthermore, we propose a number of improvements for some of these scheme and protocols. Finally, we provide a number of suggestions and recommendations that can be valuable to database researchers and users.

*Some of the following sections were paraphrased or taken verbatim from the following published work.*

[BKR19] **Dmytro Bogatov**, George Kollios, and Leonid 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](https://doi.org/10.14778/3324301.3324309)

## 4.1 Introduction

Order-Preserving Encryption (OPE) was proposed by Agrawal et al. [AKSX04] in their seminal paper. The main idea is to “encrypt” numerical values into ciphertexts that have the same order as the original plaintexts. This is a very useful primitive since it allows a database system to make comparisons between ciphertexts and get the same results as if it had operated on plaintexts. A scheme was proposed in [AKSX04] but no security analysis was given.

Boldyreva et al. [BCLO09] were the first to treat OPE schemes from a cryptographic point of view, providing security models and rigorous analysis. The ideal functionality of such a scheme is to leak only the order of the plaintexts and nothing more. However, it was shown by Boldyreva et al. [BCLO09] that the ideal functionality is not achievable if the scheme is *stateless* and *immutable*. In order to achieve the ideal functionality, Popa, Li, and Zeldovich [PLZ13] proposed a mutable scheme that constructs a binary tree on plaintexts and uses paths as ciphertexts. This tree is the encrypted full state of the dataset, and once an insertion or a deletion rebalances the tree, multiple ciphertexts get mutated. Kerschbaum [Ker15] proposed an improvement on this scheme that also hides the frequency of each plaintext (how many times a given value appears).

Furthermore, in order to improve the security of these schemes, Boneh et al. [Bon+15] proposed to generalize OPE to Order-Revealing Encryption (ORE). In ORE, ciphertexts have no particular order and look more like typical semantically secure encryptions. The database system has a special comparison function that can be used to compare two ciphertexts. These schemes are more secure than OPE schemes, although they still leak some information, and in general are more expensive to compute. Since these schemes leak some information, a number of recent works considered attacks on systems that may use these schemes [IKK12; IKK14; NKW15;

GRS17; KKNO16; CGPR15; DDC16; LMP18; Bin+18; WZ18]. Most of these attacks assume the attacker possesses *auxiliary information* and no other protections are available.

OPE / ORE schemes can be used with almost no changes to the underlying database engine. To provide greater security, a number of more complex protocols for protecting data in outsourced databases have been proposed. These constructions are often interactive, rely on custom data structures and are optimized for certain tasks, such as range queries. Naturally, the more secure the protocol is, the larger performance overhead it incurs. The most secure of these — Oblivious Random Access Machine (ORAM) based protocol — provides strong, well-understood, cryptographic privacy guarantees with no information leakage.

Applications that can benefit from such schemes and protocols include cloud access security brokers and financial and banking applications. Indeed, a number of commercial brokers including Skyhigh Networks<sup>1</sup> and CipherCloud<sup>2</sup> have been using some form of OPE or ORE schemes in their systems. In addition, financial institutions may be able to encrypt their data using the aforementioned schemes in order to provide another layer of security, assuming that the performance overhead is acceptable. For many of these applications the auxiliary information that is needed for the attacks mentioned above is either unavailable or difficult to get.

Currently, it is a very challenging task for users to choose an appropriate data privacy approach for their application, because the security and performance tradeoff is not well understood. Both security and performance of every approach need to be thoroughly evaluated. Characterizing security benefits of different approaches remains an open problem, unlikely to be solved in the immediate future. However, it is possible to evaluate the performance of each approach, so as to enable better-

---

<sup>1</sup><https://www.skyhighnetworks.com/>

<sup>2</sup><https://www.ciphercloud.com/>



informed decisions about whether the improved performance of some schemes is worth the uncertainty about the security they achieve.

We emphasize that it is not trivial to evaluate the performance of these schemes. Many of the papers presenting the above approaches provide only a theoretical treatment and concentrate more on the security definitions and analysis and less on the performance. Some of these constructions have not been even implemented properly. Furthermore, even though the main target of these schemes and protocols are database applications, most of them have not been evaluated in database settings.

To address this problem, in this paper we design a new framework that allows for systematic and extensive comparison of **OPE** and **ORE** schemes and secure range query protocols in the context of database applications. We employ these schemes in database indexing techniques (i.e. B<sup>+</sup> trees) and query protocols and we report various costs including I/O complexity.

The main contribution of this work is to present an experimental evaluation using both real and synthetic datasets using our new framework that tracks not only time but also primitive usage, I/O complexity, and communication cost. In the process, we present improvements for some of the schemes that make them more efficient and/or more secure. To make understanding of these schemes easier for the reader, we present the main ideas behind these constructions, discuss their security definitions and leakage profiles, and provide an analysis of implementation challenges for each one.

## 4.2 Security Perspective

Each scheme and protocol we analyze has its own security definition, which captures different leakage levels. We attempt to unify these definitions and analyze them under

a common framework. We also attempt to assess relative security of these definitions and analyze their leakages.

In this work we mostly consider the snapshot model, where the attacker can observe all the database contents at one time instant. Note that this excludes timing attacks such as measuring encryption time. All security definitions of the schemes and protocols that we discuss here are based on this model. Also, the snapshot attacker is the most common attacker that we face today [BPP17]. The idea is that a hacker or an insider can steal the entire encrypted database and all its contents at some point in time.

Beyond the snapshot model, it is also possible to consider a stronger adversary who can track communication volume and data access patterns in real time. Approaches that help protect against such an attacker include **ORAM** for protection against access pattern leakage and differential privacy for protection against communication volume leakage. Although this model is not a primary target of this paper, our benchmark includes a protocol (Section 4.4.5) that is secure in this setting to show the cost of adding such protection.

We wanted to specifically comment on a work of Grubbs et al. [Gru+17], which demonstrates a series of attacks against **OPE** and **ORE** schemes. The attacks can be very successful, but they depend on certain prerequisites. First, all attacks assume the existence of a well-correlated auxiliary dataset. Second, the binomial attack, which works against a “perfectly secure frequency-hiding scheme”, reliably recovers only high-frequency elements. Finally, the attacks are specifically devastating against encrypted strings (e.g. first and last names) as opposed to numerical data, and we also do not recommend using **OPE** / **ORE** for strings (see Section 4.2.1). One of the conclusions of our work is that security is negatively correlated with performance and it is up to a practitioner to trade off security and performance constraints.

### 4.2.1 A note on variable-length inputs

A generic **OPE** / **ORE** scheme accepts bit-strings of any length as inputs, and treats them as numbers or processes them bit-by-bit. We warn against supplying raw bytes of variable length (e.g. encoded strings) to **OPE** and **ORE** schemes, as such an approach will introduce both performance and security challenges.

From the performance standpoint, the complexity of **OPE** / **ORE** schemes usually depends on the input length at least linearly (see Table 4.1). 32-bit numbers already introduce a noticeable overhead for some (usually more secure) schemes, and supplying arbitrary-length inputs may worsen performance by at least an order of magnitude.

Security of such a construction will be minimal as most schemes leak some information about the magnitude of the difference, and longer inputs will naturally be treated as larger numbers. Thus, the difference between long and short inputs will be apparent. We refer to the work of Grubbs et al. [Gru+17] as they have a practically supported discussion of security consequences of using **OPE** / **ORE** with arbitrary strings.

On the other hand, other protocols in our benchmark can usually handle variable-length inputs as long as they fit into a single block for the underlying block cipher.

## 4.3 **OPE** and **ORE** Schemes

An **Order-Revealing Encryption** scheme is a triple of polynomial-time algorithms **KGEN**, **ENC** and **CMP**. **KGEN** generates a key of parameterized length (the  $\lambda$  parameter). **ENC** takes a numerical input (as a bit string) and produces a ciphertext. **CMP** takes two ciphertexts generated by the scheme and outputs whether the first plaintext was strictly less than the second. Note that being able to check this condition is enough to apply all other comparison operators ( $<$ ,  $\leq$ ,  $=$ ,  $\geq$ ,  $>$ ). Also note

that an **ORE** scheme does not include a decryption algorithm, because one can simply append a symmetric encryption of the plaintext to the produced ciphertext and use it for decryption.<sup>3</sup> An **Order-Preserving Encryption (OPE)** scheme is a particular case of an **ORE** scheme where ciphertexts are numerical and thus CMP routine is trivial (the numerical order of ciphertexts is the same as underlying plaintexts). **OPE** may optionally include a decryption algorithm, since appending a symmetric ciphertext is no longer possible.

Both **OPE** and **ORE** schemes by definition allow to totally order the ciphertexts. This is their inherent leakage (by design) and all the **OPE / ORE** security definitions account for this and possibly additional leakage.

We proceed by describing and analyzing the **OPE / ORE** schemes we have benchmarked. All plaintexts are assumed to be 32-bit signed integers, or  $n$ -bit inputs in complexity analysis. **OPE** ciphertexts are assumed to be 64-bit signed integers.

From here, we will use the term **ORE** to refer to both **OPE** and **ORE**, unless explicitly stated otherwise. Each scheme has its own subsection where the first part is the construction overview followed by security discussion, and the second part is our theoretical and experimental analysis.

#### 4.3.1 BCLO **OPE** [BCLO09]

The **OPE** scheme by Boldyreva et al. [BCLO09] was the first **OPE** scheme that provided formal security guarantees and was used in one of the first database systems that executes queries over encrypted data (CryptDB [PRZB11]). The core principle of their construction is the natural connection between a random order-preserving function and the hypergeometric probability distribution.

---

<sup>3</sup>Given the secret key, it is possible to decrypt a ciphertext by doing binary search on the plaintext domain: encrypting known values and comparing them against the target ciphertext, until the target plaintext is found. However, this would require  $\mathcal{O}(\log |\mathcal{D}|)$  encryption and comparison operations.

The encryption algorithm works by splitting the domain into two parts according to a value sampled from the [Hypergeometric Probability Distribution \(HG\)](#), and splitting the range in half recursively. When the domain size contains a single element, the corresponding ciphertext is sampled uniformly from the current range.

All pseudo-random decisions are made by an internal [PRG](#) (referred to as [TAPE-GEN](#) in [\[BCLO09\]](#)). This way not only the algorithm is deterministic, but also decryption is possible. The decryption procedure takes the same “path” of splitting domain and range, and when the domain size reaches one, the only value left is the original plaintext.

## Security

This scheme is [POPF-CCA](#) secure [\[BCLO09\]](#), meaning that it is as secure as the underlying ideal object — randomly sampled order-preserving function from a certain domain to a certain range. For practical values of the parameters, [Boldyreva, Chenette, and O’Neill](#) [\[BCO11\]](#) showed that the distance between the plaintexts can be approximated to an accuracy of about the square root of the domain size. In other words, approximately, half of the bits (the most significant) are leaked. [Grubbs et al.](#) [\[Gru+17\]](#) showed that this leakage allows to almost entirely decrypt the ciphertexts (given auxiliary data with a similar distribution) and encrypting strings (rather than numbers) with this scheme is especially dangerous (see [Section 4.2.1](#)).

## Analysis and implementation challenges

Efficient sampling from the hypergeometric distribution is a challenge by itself. Authors suggest using a randomized yet exact (not approximate) Fortran algorithm by [Kachitvichyanukul and Schmeiser](#) [\[KS88\]](#). It should be noted that the algorithm relies on infinite precision floating-point numbers, which most regular frameworks do not have. The security consequences of finite precision computations is actually an open

question. The complexity of this randomized algorithm is hard to analyze; however, we empirically verified that its running time is no worse than linear in the input bit length. The authors also suggest a different algorithm for smaller inputs [Wal77].

On average, encryption and decryption algorithms make  $n$  calls to **HG**, which in turn consumes entropy generated by the internal **PRG**. The entropy, and thus the number of calls to **PRG**, needed for one **HG** run is hard to analyze theoretically. However, we derived this number experimentally (see Section 4.5).

BCLO has been implemented in numerous languages and has been deployed in a number of secure systems. We add C# implementation to the list, and recommend using a library that supports infinite precision floating-point numbers when building the hypergeometric sampler.

### 4.3.2 CLWW ORE [CLWW16]

The **ORE** scheme by Chenette et al. [CLWW16], which authors call “Practical ORE”, is the first efficient **ORE** implementation based on **PRFs**.

On encryption, the plaintext is split into  $n$  values in the following way. For each bit, a value is this bit concatenated with all more significant bits. This value is given to a keyed **PRF** and the result is numerically added to the next less significant bit. This resulting list of  $n$  elements is the ciphertext.

The comparison routine traverses two lists in-order looking for the case when one value is greater than the other by exactly one, revealing location and value of the first differing bit. If no such index exists, the plaintexts are equal.

## Security

A generic **ORE** security definition was introduced along with the scheme [CLWW16]. **ORE** leakage is more clearly quantified than in **OPE**. The definition says that the scheme is secure with a leakage  $\mathcal{L}(\cdot)$  if there exists an algorithm (simulator) that has

access to the leakage function and can generate output indistinguishable from the one generated by the real scheme. This scheme satisfies **ORE** security definition with the leakage  $\mathcal{L}(\cdot)$  of the location and value of the first differing bit of every pair of plaintexts. Note that the most significant differing bit also leaks the approximate distance between two values.

### Analysis and implementation challenges

On encryption the algorithm makes  $n$  calls to **PRF** and the comparison procedure does not use any cryptographic primitives. Ciphertext is a list of length  $n$ , where each element is an output of a **PRF** modulo 3. The authors claim that the ciphertext’s size is  $n \log_2 3$ , just 1.6 times larger than the plaintext’s size. While this may be true for large enough  $n$  if ternary encoding is used, we found that in practice the ciphertext size is still  $2n$ .  $1.6n$  for 32-bit words is 51.2 bits, which will have to occupy one 64-bit word, or two 32-bit words, therefore resulting in  $2n$  anyway.

#### 4.3.3 Lewi-Wu **ORE** [LW16]

Lewi and Wu [LW16] presented an improved version of the CLWW scheme [CLWW16] which leaks strictly less.

The novel idea was to use the “left / right framework” in which two ciphertexts get generated — left and right. The right ciphertexts are semantically secure, so an adversary will learn nothing from them. Comparison is only defined between the left ciphertext of one plaintext and the right ciphertext of another plaintext.

The approach is to split the plaintext into blocks of  $d$  bits. The ciphertext is computed block-wise. For the right side, the algorithm compares the given block value to all  $2^d$  possible block values; each comparison result is added (modulo 3) to a **PRF** of the previous blocks. All  $2^d$  comparison results go into the right ciphertext. The left side, which is shorter, is produced in such a way as to reveal the correct

comparison result. This way the location of the differing bit inside the block is hidden, but the location of the first differing block is revealed.

## Security

This scheme satisfies the **ORE** security definition introduced by Chenette et al. [CLWW16] with the leakage  $\mathcal{L}(\cdot)$  of the location of the first differing *block*. This property allows a practitioner to set performance-security tradeoff by tuning the block size. Left / right framework is particularly useful in a  $B^+$  tree since it is possible to store only one (semantically secure) side of a ciphertext in the structure (see Section 4.4.1).

## Analysis and implementation challenges

Let  $n$  be the size of input in bits (for example, 32) and  $d$  be the number of bits per block (for example, 2).

Left encryption loops  $\frac{n}{d}$  times making one **PRP** call and two **PRF** calls each iteration. Right encryption loops  $\frac{n}{d}2^d$  times making one **PRP** call, one hash call and two **PRF** calls each iteration. Comparison makes  $\frac{n}{d}$  calls to hash at worst and half of that number on average. Please note that the complexity of right encryption is exponential in the block size. In the Table 4.1 the **PRP** usage is linear due to our improvement. The ciphertext size is no longer negligible —  $\frac{n}{d}(\lambda + n + 2^{d+1}) + \lambda$ , for  $\lambda$  being **PRF** output size.

The implementation details of this approach raise an interesting security question. Although the authors suggest using 3-rounds Feistel networks [SK96] for **PRP** and use it in their implementation, it may not be secure for small input sizes. Feistel networks security depends on the input size [HR10] — exponential in the input size. However, the typical input for **PRP** in their scheme is 2–8 bits, thus even exponential number is small.



We have considered multiple **PRP** implementations to use instead of the Feistel networks. Because the domain size is small (from  $2^2$  to  $2^8$  elements), we have decided to build a **PRP** by simply using the key as an index into the space of all possible permutations on the domain, where a permutation is obtained from the key via Knuth shuffle (this approach was mentioned in [MRS09]). Another important aspect of the implementation is that for each block we need to compute the permutation of all the values inside the block. This operation applied many times can be expensive. To address this, we propose to generate a **PRP** table once for the whole block and then use this table when one needs to compute the location of an element of permutation. This can reduce the **PRP** usage (indeed, we observe a reduction from 80 to 32 calls in our case). We evaluate this improved approach in our experimental section.

#### 4.3.4 CLOZ ORE [Cas+18]

Cash et al. [Cas+18] introduced a new **ORE** scheme that provably leaks less than any previous scheme. The idea is to use Chenette et al. [CLWW16] construction (see Section 4.3.2), but permute the list of **PRF** outputs. The original order of those outputs is not necessary, as one can simply find a pair that differs by one. This is not enough to reduce leakage, however, since an adversary can count how many elements two ciphertexts have in common.

To address this problem, the authors define a new primitive they call a **Property-Preserving Hash (PPH)**. A **PPH** as defined and used in [Cas+18], allows one to expose a property (specifically  $y \stackrel{?}{=} x + 1$ ) of two (numerical) elements such that nothing else is leaked. In particular, the outputs are randomized, so same element hashed twice will have different hashes. Please refer to the original paper [Cas+18] for formal correctness and security definitions.

Equipped with the **PPH** primitive, the algorithm “hashes” the elements of the ciphertexts before outputting them. Due to security of **PPH**, the adversary would

not be able to count how many elements two ciphertexts have in common, thus, would not be able to tell the location of differing bit.

## Security

The strong side of the scheme is its security. The scheme leaks  $\mathcal{L}(\cdot)$  an *equality pattern* of the most-significant differing bits (satisfying Chenette et al. [CLWW16] definition). As defined in [Cas+18], the intuition behind equality pattern is that for any triple of plaintexts  $m_1, m_2, m_3$ , it leaks whether  $m_2$  differs from  $m_1$  before  $m_3$  does. We do not know of any attacks against this construction (partially because no implementation exists yet, see next subsection), but it is inherently vulnerable to frequency attacks that apply to all frequency-revealing ORE schemes (see Section 4.2).

## Analysis and implementation challenges

On encryption, the scheme makes  $n$  calls to PRF,  $n$  calls to PPH HASH and one call to PRP. Comparison is more expensive, as the scheme makes  $n^2$  calls to PPH TEST.

The scheme has two limitations that make it impractical. The first one is the square number of calls to PPH, which is around 1024 for a single comparison.

The second problem is the PPH itself. Authors suggest a construction based on bilinear maps. The hash of an argument is an element of a group, and the test algorithm is computing a pairing. This operation is very expensive — order of magnitude more expensive than any other primitive we have implemented for other schemes.

We have implemented this scheme in C++ using the PBC library [Lyn18] to empirically assess schemes’s performance, and on our machine (see Section 4.5), a single comparison takes 1.9 seconds on average. Although we have produced the first (correct and secure) real implementation of this scheme in C++, it is infeasible to use it in the benchmark (it will take years to complete a single run). Therefore, for the purposes of our benchmark, we implemented a “fake” version of PPH — correct,

but insecure, which does not use pairings. Consequently, in our analysis we did not benchmark the speed of the scheme, but measured all other data.

#### 4.3.5 FH-OPE [Ker15]

Frequency-hiding OPE by Kerschbaum [Ker15] is a stateful scheme that hides the frequency of the plaintexts, so the adversary is unable to construct a frequency histogram.

This scheme is stateful, which means that the client needs to keep a data structure and update it with every encryption and decryption. The data structure is a binary search tree where the node’s value is the plaintext and node’s position in a tree is the ciphertext. For example, consider the range  $[1, 128]$ . Any plaintext that happens to arrive first (for example, 6), will be the root, and thus the ciphertext is 64. Then any plaintext smaller than the root, say 3, will become the left child of the root, and will produce the ciphertext 32. To encrypt a value, the algorithm traverses the tree until it finds a spot for the new plaintext, or finds the same plaintext. If the same plaintext is found, the traversal pseudo-randomly passes to the left or right child, up to the leaf. This way, the invariant of the tree — intervals of the same plaintexts do not overlap — is maintained. The ciphertext generated from the new node’s position is returned.

Due to randomized ciphertexts, the comparison algorithm is more complicated than in the regular deterministic OPE. To properly compare ciphertexts, the algorithm needs to know the boundaries — the minimum and maximum ciphertexts for a particular plaintext. The client is responsible for traversing the tree to find the plaintext for the ciphertext and then minimum and maximum ciphertext values. Having these values, the comparison is trivial — equality is a check that the value is within the boundaries, and other comparison operators are similar.

Authors have designed a number of heuristics to minimize the state size, however, these are mostly about compacting the tree and the result depends highly on the tree content. In our analysis, we consider the worst case performance without the use of heuristics. In our experimental evaluation, however, we did implement compaction.

## Security

The security of the scheme relies on the large range size to domain size ratio. Authors recommend at least 6 times longer ciphertexts than the plaintexts in bit-length, which means ciphertexts should be 192-bit numbers that are not commonly supported. It is possible to operate over arbitrary-length numbers, but the performance overhead would be substantial. We did a quick micro-benchmark in C# and the overhead of using `BigInteger` is 15–20 times for basic arithmetic operations.

This scheme satisfies IND-FAOCPA definition (introduced along with the scheme [Ker15]), meaning that it does not leak the equality or relative distance between the plaintexts. This definition has been criticized in [MRS18], who claim that the definition is imprecise and propose an enhanced definition along with a small change to construction to satisfy this new definition. Both schemes leak the insertion order, because it affects the tree structure. We do not know of any attacks against this leakage, but it does not mean they cannot exist. Grubbs et al. [Gru+17] describe an attack against this scheme (binomial attack), but it applies to any perfectly secure (leaking only total order) frequency-hiding OPE.

## Analysis and implementation challenges

If the binary tree grows in only one direction, at some point it will be impossible to generate another ciphertext. In this case, the tree has to be rebalanced. This procedure will invalidate all ciphertexts already generated. This property makes the scheme difficult to use in some protocols since they usually rely on the ciphertexts on

the server being always valid. The authors explicitly mention that the scheme works under the assumption of uniform input. However, the rebalancing will be caused by insertion of just 65 consecutive input elements for 64-bit integer range.

The scheme makes one tree traversal on encryption and decryption. Comparison is trickier as it requires one traversal to get the plaintext, and two traversals for minimum and maximum ciphertexts. We understand that it is possible to get these values in fewer than three traversals, but we did not optimize the scheme for the analysis and evaluation.

For practitioners we note that the stateful nature of the scheme implies that the client storage is no longer negligible as the state grows proportionally to the number of encryptions. We also note that implementing compaction extensions will affect code complexity and performance. Finally, we stress again that some non-uniform inputs can break the scheme by causing all ciphertexts to be invalid. It is up to the users of the scheme to ensure uniformity of the input, which poses serious restrictions on the usage of the scheme.

#### 4.4 Secure Range Query Protocols

We proceed by describing and analyzing the range query protocols we have chosen. For the purpose of this paper, a secure range-query protocol is defined as a client-server communication involving construction and search stages. Communication occurs between a client, who owns some sensitive data, and an honest server, who securely stores it. In construction stage, a client sends the server the encrypted datapoints (index-value tuples) and the server stores them in some internal data structure. In search stage, a client asks the server for a range (usually specifying it with encrypted endpoints) and the server returns a set of encrypted records matching the query. Note that the server may interact with the client during both stages (e.g.

Table 4.1: [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,  $\lambda$  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 (on top of total order)
	Encryption	Comparison		
[BCLO09]	$\approx n$ (41) <b>HG</b>	none	$2n$ (64)	$\approx$ <b>top half of the bits</b>
[CLWW16]	$n$ (32) PRF	none	$2n$ (64)	<b>Most-significant differing bit</b>
[LW16]	$2^{n/d}$ (32) <b>PRP</b> $2^{n/d} (2^d + 1)$ (160) PRF $\frac{n}{d} 2^d$ (64) Hash	$\frac{n}{2d}$ (9) Hash	$\frac{n}{d} (\lambda + n + 2^{d+1}) + \lambda$ (2816)	Most-significant differing block
[Cas+18]	$n$ (32) PRF $n$ (32) PPH 1 PRP	$n^2$ (1046) <b>PPH</b>	$n \cdot h$ (4096)	Equality pattern of the most-significant differing bit
[Ker15]	1 Traversal	3 Traversals	$3 \cdot n \cdot N$ (86842)	Insertion order

ask the client to sort a small list of ciphertexts). Also note that we do not allow batch insertions as it would limit the use cases (e.g. client may require interactive one-by-one insertions).

The first protocol is a family of constructions where a data structure (B+ tree in this case) uses **ORE** schemes internally. Then, we present alternative solutions with varying performance and security profiles, not relying on **ORE**. Finally, we introduce two baseline solutions we will use in the benchmark — one that achieves the best performance and the other that achieves the maximal security.

#### 4.4.1 Range query protocol from **ORE**

So far we have analyzed **OPE** and **ORE** schemes without much context. One of the best uses of an **ORE** is within a secure protocol. In this section we provide a construction of a search protocol built with a B+ tree working on top of an **ORE** scheme and analyze its security and performance.

The general idea is to consider some data structure that is optimized for range queries, and to modify it to change all comparison operators to **ORE** scheme’s **CMP** calls. This way the data structure can operate only on ciphertexts. Performance overhead would be that of using the **ORE** scheme’s **CMP** routine instead of a plain comparison. Space overhead would be that of storing ciphertexts instead of plaintexts.

In this paper, we have implemented a typical B+ tree [BM70] (with a proper deletion algorithm [Jan95]) as a data structure.

For protocols, we also analyze the **I/O** performance and the communication cost. In particular, we are interested in the expected number of **I/O** requests the server would have made to the secondary storage, and the number and size of messages parties would have exchanged.

The relative performance of the B+ tree depends only on the page capacity (the longer the ciphertexts, the smaller the branching factor). Therefore, the query com-

plexity is  $\mathcal{O}(\log_B(N/B) + r/B)$ , where  $B$  is the number of records (ciphertexts) in a block,  $N$  is the number of records (ciphertexts) in the tree and  $r$  is the number of records (ciphertexts) in the result (none for insertions).

Communication amount of the protocol is relatively small as its insertions and queries require at most one round trip.

### Security

The leakage of this protocol consists of leakage of the underlying **ORE** scheme plus whatever information about insertion order is available in the  $B^+$  tree. Please note that Lewi-Wu [LW16] **ORE** is particularly well-suited in this construction with its left / right framework, because only the semantically secure side of the ciphertext is stored in the structure. In this case, the **ORE** leakage becomes only the total order and the security of the protocol is comparable with other non-**ORE** constructions.

#### 4.4.2 Kerschbaum-Tueno [KT19]

Kerschbaum and Tueno [KT19] proposed a new data structure, which satisfies their own definitions of security (IND-CPA-DS) and efficiency (search operation has polylogarithmic running time and linear space complexity).

In short, the idea is to maintain a (circular) array of symmetrically encrypted ciphertexts in order. On insertion, the array is rotated around a uniformly sampled offset to hide the location of the smallest element. Client interactively performs a binary search requesting an element, decrypting it and deciding which way to go.

### Security

Authors prove that this construction is IND-CPA-DS secure (defined in the same paper [KT19]). The definition assumes an array data structure and therefore serves specifically this construction (as opposed to being generic). It provably hides the



frequency due to semantic encryption and hides the location of the first element due to random rotations. Leakage-wise this construction is strictly better than B<sup>+</sup> tree with ORE — they both leak total order, but [KT19] hides distance information and smallest / largest elements. Specifically, for all pairs of consecutive elements  $e_i$  and  $e_{i+1}$  it is revealed that  $e_{i+1} \geq e_i$  except for one pair of smallest and largest elements in the set.

### Analysis and implementation challenges

Insertions are I/O-heavy because they involve rotation of the whole data structure. All records will be read and written, thus the complexity is  $\mathcal{O}(N/B)$ . Searches are faster since they involve logarithmic number of blocks. The first few blocks can be cached and the last substantial number of requests during the binary search will target a small number of blocks. The complexity is then  $\mathcal{O}(\log_2 N/B)$ .

Communication volume is small as well. Insertion requires  $\log_2 N$  messages from each side. Searches require double that number because separate protocol is run for both endpoints.

The data structure is linear in size, and the client storage is always small. Sizes of messages are also small as only a single ciphertext is usually transferred.

For practitioners we have a few points. The construction in the original paper [KT19] contains a typo as  $m$  and  $m'$  must be swapped in the insertion algorithm. Also, we have found some rare edge cases; when duplicate elements span over the modulo, the algorithm may not return the correct answer. Both inconsistencies can be fixed however. This protocol is not optimized for I/O operations for insertions, and thus would be better suited for batch uploads.

### 4.4.3 POPE [RACY16]

Roche et al. [RACY16] presented a protocol, direct improvement over mOPE [PLZ13], which is especially suitable for large number of insertions and small number of queries. The construction is heavily based on buffer trees [Arg03] to support fast insertion and lazy sorting.

The idea is to maintain a POPE tree on the server and have the client manipulate that tree. POPE tree is similar to B-tree, in that the nodes have multiple children and nodes are sorted on each level. Each node has an ordered list of *labels* of size  $L$  and an unbounded unsorted set of encrypted data called buffer. Parameter  $L$  controls the list size, the leaf's buffer size, and the size of client's working set. The insertion procedure simply adds an encrypted piece of data to the root's buffer, thus we do not concentrate on insertion analysis in this section.

The query procedure is more complex. To answer a query, the server interacts with the client to split the tree according to the query endpoints. On a high level, for each endpoint the buffers are cleared (content pushed down to leaves), and nodes in the paths are split. After that, answering a query means replying with all ciphertexts in all buffers between the two endpoint leaves.

The authors provide cost analysis of their construction. Search operations are expected to require  $\mathcal{O}(\log_L n)$  rounds. It must be noted that the first queries will require many more rounds, since large buffers must be sorted.

### Security

This construction satisfies the security definition of frequency-hiding partial order-preserving (FH-POP) protocol (introduced in the paper [RACY16]). According to [RACY16, Theorem 3], after  $n$  insertions and  $m$  queries with local storage of size  $L$ , where  $mL \in o(n)$ , the POPE scheme is frequency-hiding partial order-preserving with

$\Omega\left(\frac{n^2}{mL \log_L n} - n\right)$  incomparable pairs of elements. Simply put, the construction leaks pairwise order of a *bounded* number of elements. Aside from this, the construction provably hides the frequency (i.e., equality) of the elements.

### Analysis and implementation challenges

In our analysis we count each request-response communication as a round. This is different from [RACY16] where they use *streaming* a number of elements as a single round. The rationale for our approach is that if we allow persistent channels additionally to messages, then any protocol can open a channel for each operation. Thus, we do not allow channels for all protocols in our analysis.

Also, as noted by the authors, if  $L = n^\epsilon$  for  $0 < \epsilon < 1$ , then the amortized costs become  $\mathcal{O}(1)$ . While this is true, in our analysis the choice of  $L$  depends on the storage volume block size for I/O optimizations, instead of the client's volatile storage capacity. Thus, the costs remain logarithmic.

Search bandwidth depends heavily on the current state of the tree. When the tree is completely unsorted (the first query), all elements of the tree will be transferred to split the large root, then possibly internal node will have to be split requiring sending of  $\frac{N}{L}$  elements, and so on, thus  $\mathcal{O}(N + r)$ . When the tree is completely sorted (after a large number of uniform queries), the bandwidth will be similar to that of a standard B+ tree —  $\mathcal{O}(L \log_L N + r)$ . The average case is hard to compute; however, authors prove an upper bound on bandwidth after  $n$  insertions and  $m$  queries —  $\mathcal{O}(mL \log_L n + n \log_L m + n \log_L(\lg n))$ .

POPE tree is not optimized for I/O the way B-tree is. Search complexity is hard to analyze as is bandwidth complexity. In the worst-case (first query), all blocks need to be accessed  $\mathcal{O}\left(\frac{N}{B} + \frac{r}{B}\right)$ . In the best-case all nodes occupy exactly one block and I/O complexity is the same as with B+ tree  $\mathcal{O}\left(\log_L \frac{N}{B} + \frac{r}{B}\right)$ . The average case is in

between and matters get worse as the node is not guaranteed to occupy a single block due to the buffers of arbitrary size.

Client’s persistent storage is negligibly small — it stores the encryption key. Volatile storage is bounded by  $L$ .

For practitioners we present a number of things to consider. Buffer within one node is unsorted, so in the worst-case,  $L$ -sized chunks remain unordered. Due to this property, the query result may contain up to  $2(L - 1)$  extra entries, which the client will have to discard from the response.

The first query after a large number of insertions will result in client sorting the whole  $N$  elements, and thus, POPE has different performance for cold and warm start. Also, even to navigate an already structured tree, the server has to send to the client the whole  $L$  elements and ask where to go on all levels.

Furthermore, [RACY16] does not stress the fact that after alternating insertions and queries, it may happen that some intermediate buffers are not empty, thus returning buffers between endpoints must include intermediate buffers as well. The consequence is that the whole subtree is traversed between paths to endpoints, unlike the B+ tree case where only leaves are involved.

Finally, POPE tree is not optimized for I/O operations. Even if  $L$  is chosen so that the node fits in the block, only leaves and only after some number of searches will optimally fit in blocks. Arbitrary sized buffers of intermediate nodes and the lack of underflow requirement do not allow for I/O optimization.

#### 4.4.4 Logarithmic-BRC [Dem+16]

Demertzis et al. [Dem+16] introduced a novel protocol called “Logarithmic-BRC” whose I/O complexity depends only on the result size, regardless of the database size. The core primitive for their construction is a **Searchable Symmetric Encryption** (SSE) scheme. An SSE scheme is a server-client protocol in which the server stores a

specially encrypted keywords-to-documents map, and a client can query documents with keywords while the server learns neither keywords nor the documents. Note that the map stores short document identifiers instead of the actual documents, and we will use the term “documents” to mean “document identifiers” or “record IDs” in this section.

The construction treats record values as documents and index ranges as keywords so that records can be retrieved by the ranges that include them. Specifically, a client builds a virtual binary tree over the domain of indices and assigns each record a set of keywords, which is the path from that record to the root. This way, the root keyword is associated with all documents and the leaf keyword is associated with only one record.

Upon query, a client computes a cover — a set of nodes whose sub-trees cover the requested range. A client sends these keywords to the SSE server, which returns encrypted documents — result values. Of the several covering techniques suggested in the protocol [Dem+16] we have chosen the Best Range Cover (BRC), because it results in fewest nodes and does not return false-positives. Kiayias et al. [KPTZ13] have proven that the worst-case number of nodes for domain of size  $N$  is  $\mathcal{O}(\log N)$  and presented an efficient BRC algorithm.

## Security

In a snapshot setting, this construction’s security is that of the SSE. We have used [Cas+13] and [Cas+14] SSE schemes; their leakage in a snapshot setting is the database size and at most some initialization parameters. Thus, the security of these schemes is high enough to call them *fully hiding* in our setting. Additional access pattern leakage comes up during queries; exact implications of this leakage remain an open research problem but it is known that it can be harmful [KKNO16].

## Analysis and implementation challenges

Communication involves a client sending at worst  $\log_2 N$  keywords and server responding with the exact result.

For each keyword in the query set, server will query the SSE scheme, which will return  $r$  documents. Therefore, server’s I/O complexity is that of SSE.

Demertzis et al. [Dem+16] have used [Cas+13] SSE scheme in their implementation, but we have found it slow in terms of I/O. Instead, we have implemented an improved scheme [Cas+14], which directly addresses I/O optimization.

Both SSE schemes’ I/O complexity is linear with the result size  $r$ . [Cas+14] scheme makes at most one I/O per result document in the worst-case and there are extensions to significantly improve I/O complexity. We have implemented the **pack** extension, which packs documents in blocks to fit the I/O pages. We note that this extension can dramatically reduce the I/Os (see Section 4.5.3 and Figure 4.8).

Logarithmic-BRC is very scalable as its performance does not depend on total data size and only degrades with the result size. Storage overhead, however, is significant. Each record is associated with the whole path in the binary tree —  $\log_2 N$  nodes (keywords). The storage complexity is therefore  $\mathcal{O}(N \log N)$ , and the overhead is then a factor of  $\log N$ .

Updates, while addressed in the original protocol, are not very practical in this construction. Authors suggest using bulk-loading for updates, maintaining merge trees, and requiring the client to do a merge once in a while. The I/O complexity of such approach is unclear. In our implementation we perform the construction stage only in batch mode, and thus do not include it in the analysis. We also emphasize that the update routine was not implemented for evaluation in the original paper.

#### 4.4.5 The two extremes

To put the aforementioned protocols in a context we introduce the baselines — an efficient and insecure construction we will refer to as *no encryption* and maximal security protocol we refer to as *ORAM*.

##### No encryption

This protocol is a regular B+ tree [BM70] without any ORE in it. It is the construction one can expect to see in almost any general-purpose database.

In terms of security it provides no guarantees — all data is in the clear. In terms of efficiency it is optimal. B+ tree data structure is optimal in I/O usage, indices inside nodes are smallest possible (integers) and there is no overhead in comparing elements inside the nodes as opposed to working with ORE ciphertexts.

##### ORAM

Oblivious Random Access Machine (ORAM) is a construction that additionally to semantic security of a snapshot setting (see Section 4.2) provably hides the access pattern — a sequence of reads and writes to particular memory locations. With ORAM an adversary would not be able to recognize a series of accesses to the same location and will not differentiate reads versus writes. ORAM was introduced by Goldreich and Ostrovsky [GO96] who also proved its lower bound (strengthened in [LN18]) — logarithmic overhead per request. A number of efficient ORAM constructions were designed (see [CXL16] for a good survey) and we use the state-of-the-art construction, PathORAM [Ste+13].

A generic ORAM server responds to read and write requests for a particular address. In our baseline protocol we store B+ tree nodes in ORAM. A client works with the tree as it normally would except each time it needs to access a node, it communicates with ORAM.

In terms of security this protocol is fully hiding in the snapshot model and provably hides the access pattern. We note that one can improve security even further by adding noise to the result obscuring communication volume. We also note that a practitioner can use a similar protocol with **ORAM** replaced with a trivial data store and have the tree nodes encrypted. It would be fully hiding in a snapshot setting, but we prefer the baseline that covers more than only the snapshot model.

In terms of performance this construction incurs some noticeable overhead. Regardless of specific **ORAM** being used, each access incurs at least logarithmic overhead according to lower bounds [GO96]. Combined with logarithmic complexity of the B+ tree itself, the complexity, both I/O and communication, is  $\mathcal{O}(\log^2 N)$ . We found that PathORAM has good I/O performance, as its internal tree structure translates into good cache affinity. Unlike in other protocols in our benchmark, **ORAM** client does most of the computational work. While the server only makes I/O requests, the client handles encryption, shuffling, and request logic.

We present this protocol as a baseline solution in terms of security over efficiency. We have not implemented stand-alone PathORAM, but rather a simulator which correctly reports I/O, communication and primitive usage. Surprisingly, we found that **ORAM** protocol’s overhead, although higher than in **ORE**-based protocols, is in-line with the most secure protocols in our benchmark.

## 4.5 Evaluation

All experiments were conducted on a single machine. We use macOS 10.14.2 with 8-Core 3.2 GHz Intel Xeon W processor, 32 GB DDR4 ECC main memory and 1 TB **SSD** disk. The main code is written in C# and runs on .NET Core 2.1.3.



Table 4.2: [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	<b>Same as ORE</b>	1 2 / 177 B	1 2 / 342 B
[KT19]	$\frac{N}{B}$ <b>494 requests</b>	$\log_2 \frac{N}{B} + \frac{r}{B}$ 17 requests	<b>Total order</b>	$\log_2 N$ 40 / 671 B	$\log_2 N$ 86 / 1 453 B
[RACY16] warm	1	$\log_L \frac{N}{B} + \frac{r}{B}$ 300 requests	<b>Partial order</b>	1	$\log_L N$ 914 / 347 kB
[RACY16] cold	1 request	$\frac{N}{B}$ <b>2175 requests</b>	Fully hiding	2 / 32 B	$N$ <b>498K / 9 MB</b>
[Dem+16]	—	$r$ <b>40 requests</b>	Same as SSE	—	$\log_2 N$ 2 / 391 B
ORAM	$\log^2 \frac{N}{B}$ <b>31 requests</b>	$\log_2 \frac{N}{B} (\log_B \frac{N}{B} + \frac{r}{B})$ <b>185 requests</b>	Fully hiding (access pattern)	$\log^2 \frac{N}{B}$ <b>143 / 18 kB</b>	$\log^2 \frac{N}{B}$ <b>490 / 63 kB</b>

## Interactive website

Additionally to making our source code, compiled binaries and Docker images available, we want to let researchers interactively run small-sized simulations. We host a website<sup>4</sup> where one can select a protocol (including baselines, CLOZ [Cas+18] and both SSE schemes), cache size and policy and I/O page parameter; supply one’s own data and query sets, and run the simulations. Simulations are run one at a time and usually complete within seconds. The user is then able to view the result — tables, plots, values and raw logs, which we used to build plots for this paper. Input size on the website is limited for practical purposes and users are encouraged to run arbitrary-size simulations using our binaries or Docker images.

### 4.5.1 Implementation

We have implemented most of the primitives, data structures, and constructions ourselves. For some primitives and all schemes we provided the first open-sourced cross-platform C# implementation. We note that neither primitives, nor schemes are production-ready; however, we believe they can be used in research projects and prototypes. We also emphasize that the B+ tree implementation we are using, although our own with instrumentation in it, is not custom in any way, but rather standard as defined in the original paper [BM70] with deletion algorithm by [Jan95].

This software project (22K lines of code, third of which are tests) is documented and tested (over 97% coverage). All code including primitives, data structures, schemes, protocols, simulation logic, benchmarks, build scripts and tests is published on GitHub<sup>5</sup> under CC BY-NC 4.0 license. Additionally, we have published parts of the project as stand-alone .NET Core (nuGet) packages, and we host a web-server where users can run simulations for small inputs (see previous subsection).

---

<sup>4</sup><https://ore.dbogatov.org/>

<sup>5</sup><https://github.com/dbogatov/ore-benchmark>

## Primitives

All schemes and protocols use the same primitives, most of which we implemented ourselves. All primitives rely on the default .NET Core AES implementation. .NET Core uses platform-specific implementation of AES, thus leverages AES-NI CPU instruction. In our project all key sizes are 128 bits, as is AES block size.

We implemented AES-based PRG, which uses AES [Dwo+01] in CTR mode [Dwo01] and caches unused entropy (as suggested in [Hou04]). For PRF, since we need only 128-bit inputs and outputs, we used one application of AES [KL14, Proposition 3.27]. For symmetric encryption we use AES with a random initialization vector in CBC mode [KL14, Section 3.6.2]. For hash we use default .NET Core SHA-2 implementation. For PRP, we implemented unbalanced Feistel networks [SK96] for large inputs and Knuth shuffle [Knu16] for small inputs. Please see the README of project’s repository<sup>6</sup> for low-level details.

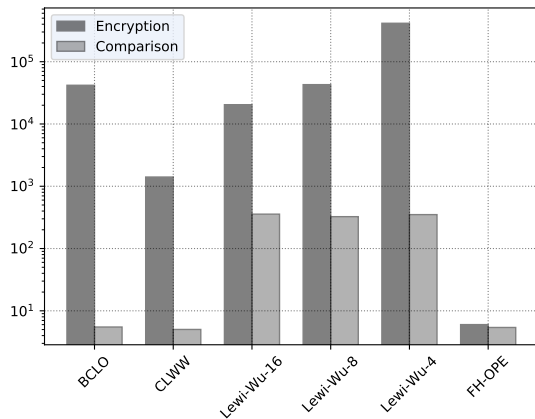


Figure 4-1: Schemes benchmark (time in microseconds, log scale). Lewi-Wu parameter is the number of blocks.

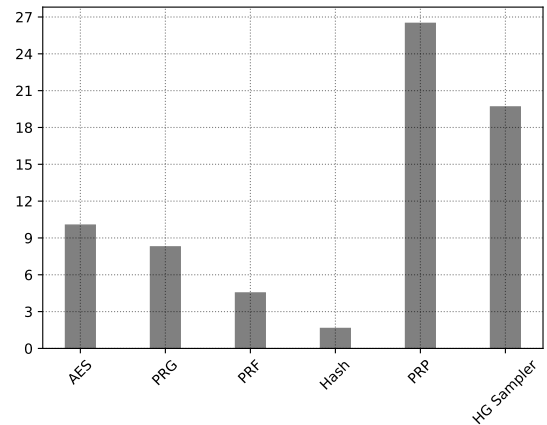


Figure 4-2: Primitives benchmark (time in microseconds)

<sup>6</sup><https://github.com/dbogatov/ore-benchmark>

## Schemes and protocols

We implemented schemes and protocols precisely as in the original papers. When we found problems or improvements, we described them in implementation challenges notes, but did not alter the original designs in our code, unless explicitly stated. Each **ORE** scheme implements a `C#` interface; thus our own implementation of B+ tree operates on a generic **ORE**. For the *no encryption* baseline, we have a stub implementation of the interface, which has identity functions for encryption and decryption. It is important to note that all schemes and protocols use exclusively our implementations of primitives. Thus we rule out the possible bias of one primitive implementation being faster than the other.

## Simulations

We have four types of simulations.

Protocol simulation runs both protocol stages — construction and search — on supplied data for all protocols including all schemes coupled with B+ tree. In this simulation we measure the primitive usage, number of **ORE** scheme operations (when applies), communication volume and size, and the number of I/O requests. We intentionally do not measure elapsed time, since it would be extremely inaccurate in this setting — simulation and measurement routines take substantial fraction of time.

Scheme simulation runs all five **ORE** schemes and tracks only the primitive usage.

The scheme benchmark, however, is designed to track time. We use Benchmark.NET [NET18] to ensure that the reported time is accurate. This tool handles issues like cold / warm start, elevating process' priority, and performing enough runs to draw statistically sound conclusions. This benchmark reports elapsed time up to nanoseconds for all four schemes (excluding CLOZ [Cas+18]) and their variants.

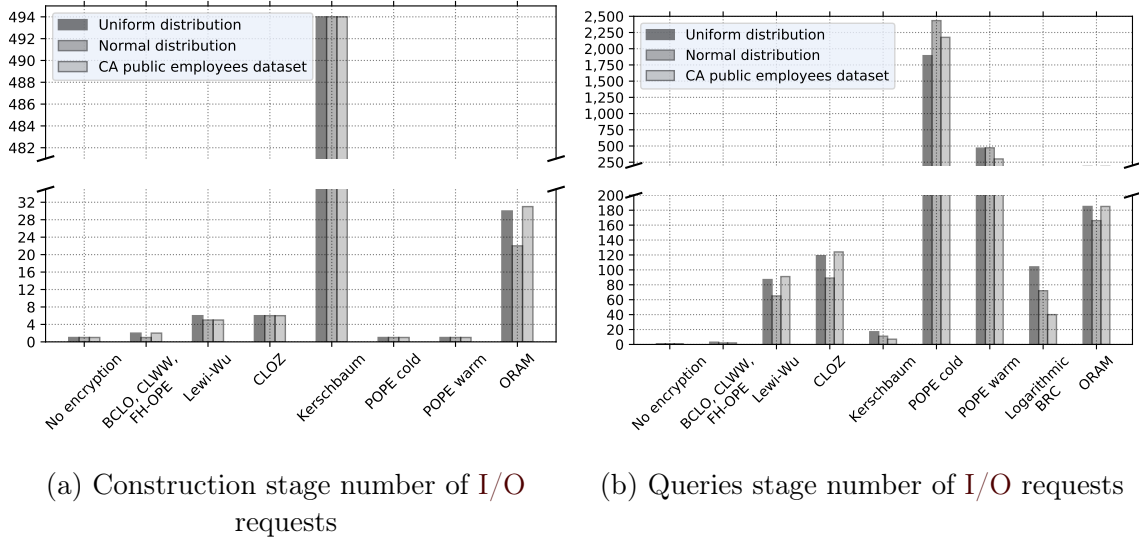


Figure 4-3: Number of I/O requests for different protocols and data distributions

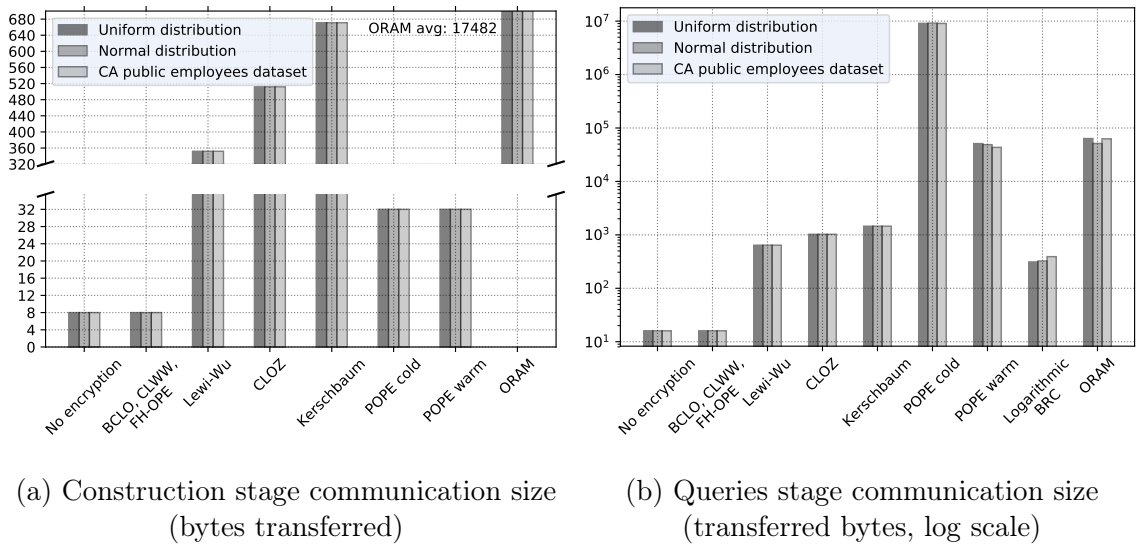


Figure 4-4: Communication size for different protocols and data distributions

Finally, primitive benchmark uses the same tool, but compares the primitives. We use it to compare different implementations of primitives (e.g. Feistel PRP vs pre-generated permutation) and to approximate time consumption of the schemes and protocols based on primitive usage.

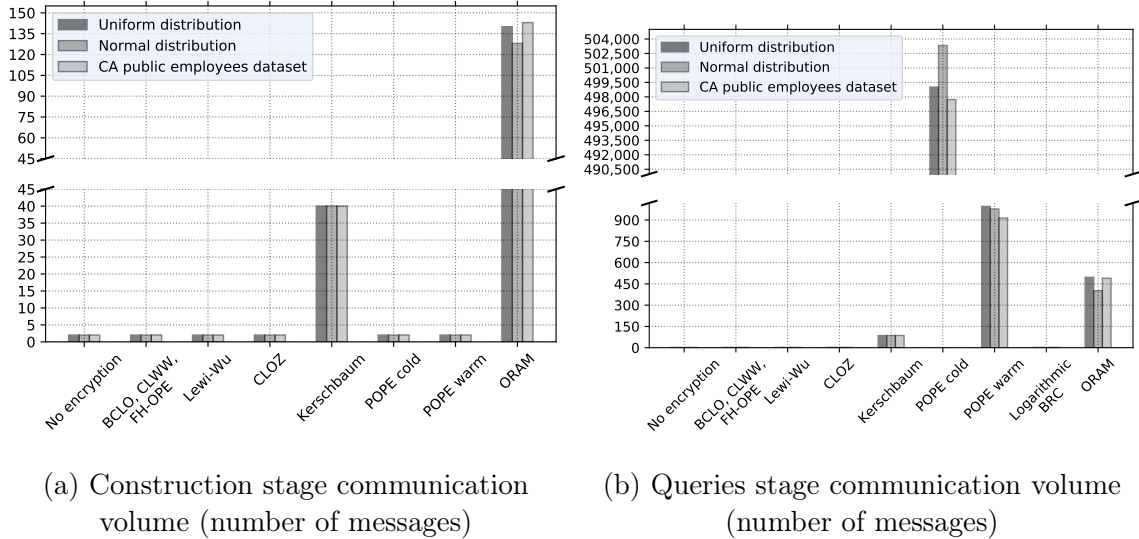


Figure 4-5: Communication volume for different protocols and data distributions

## 4.5.2 Setup

For our simulations, we have used three datasets. Two synthetic distributions, that are uniform (range is third of data size) and normal (standard deviation is 0.1 of data size). The real dataset is California public employees salaries (“total pay and benefits” column) [Tra17]. Synthetic datasets and subsets of the real dataset are generated pseudo-randomly. Queries are generated uniformly at random with a range as a percentage of data size.

## 4.5.3 Results

### Primitive usage by schemes

In Table 4.1 we show the simulation-derived values of each OPE and ORE scheme’s primitive usage. Each scheme is given 1 000 data points of each dataset. First, the scheme encrypts each data point, then decrypts each ciphertext and then performs five comparisons (all possible types) pairwise. This micro-simulation is repeated 100 times. Resulting values for primitive usage are averaged for each scheme. State

and ciphertext sizes are calculated after each operation and the values are averaged. Please note that the simulated values are consistent with the theoretical calculations.

### Benchmarks of schemes and primitives

Using the Benchmark.NET tool [NET18], we have accurately tracked the performance of the schemes and primitives running of different parameters (see Figures 4.1 and 4.2). ORE schemes benchmark setup is the same as in primitive usage simulation Section 4.5.3. Primitives were given randomly generated byte inputs and keys of different sizes (e.g. PRP of 2 to 32 bits). Benchmark.NET decides how many times to run the routine to get statistically sound results. For example, large variance results in more runs. To improve the accuracy, each run is compiled in release mode as a separate project and runs in a separate process with the highest priority.

Please note the logarithmic scale of the schemes' performances. FH-OPE is fast since it does not perform CPU-heavy operations and works in main memory. Lewi-Wu performance degrades exponentially with the increase of block size mainly due to exponential number of PRF executions and the performance of PRP degrading exponentially. Note also that Lewi-Wu comparison takes noticeable time due to Hash primitive usage.

In the primitives benchmark, it is clear that most primitives use AES under the hood. PRG and PRF take less than AES because they do not include the initialization vector generation needed for symmetric encryption. PRP is implemented as a Knuth shuffle [Knu16] and its complexity is exponential in the input bit length. Input size of 2 bits is shown on Figure 4.1. PRG does not discard the entropy generated by AES cycle, so one AES cycle can supply four 32-bit integers. PRP generates the permutation table once and does not regenerate it if the same key and number of bits are supplied.

## Protocols

In this experiment we have run each protocol with each of the three datasets. Dataset sizes are 247 000 (bounded by California Employees dataset size) and the number of queries is 1 000. Queries are generated uniformly at random with a fixed range — 0.5 % of data size. The cache size is fixed to 128 blocks, and the B+ tree branching factor as well as block sizes for other protocols are set such that the page size is 4 KiB. The values we are measuring are the number of I/O operations, communication volume, and size for both construction and query stages.

See Table 4.2 for the snapshot for particular distribution (CA employees). Figures 4.3 to 4.5 shows all values we tracked for all protocols and distributions. Values for ORE based protocols are averaged. Being “cold” in our simulations means executing the first query and being “warm” means the first query has been previously executed. This difference makes sense only for POPE as its first query incurs disproportionately large overhead by design.

Note that all ORE based protocols behave the same except when ciphertext size matters. Thus, since BCLO, CLWW and FH-OPE have the same ciphertext size, they create B+ trees with the same page capacity and have the same number of I/Os for different operations. Lewi-Wu and CLOZ schemes have relatively large ciphertexts and thus induce larger traffic (see Figure 4.4a) and smaller B+ tree branching factor resulting in greater number of I/O requests (see Figure 4.3b). Kerschbaum protocol requires high number of I/O requests during construction since it needs to insert an element into the arbitrary place in an array and rotate the data structure on a disk.

POPE suffers huge penalty on the first query (see Figures 4.3b, 4.4b and 4.5b) since it reads and sends all blocks to the client for sorting. POPE performance improves as more queries are executed.



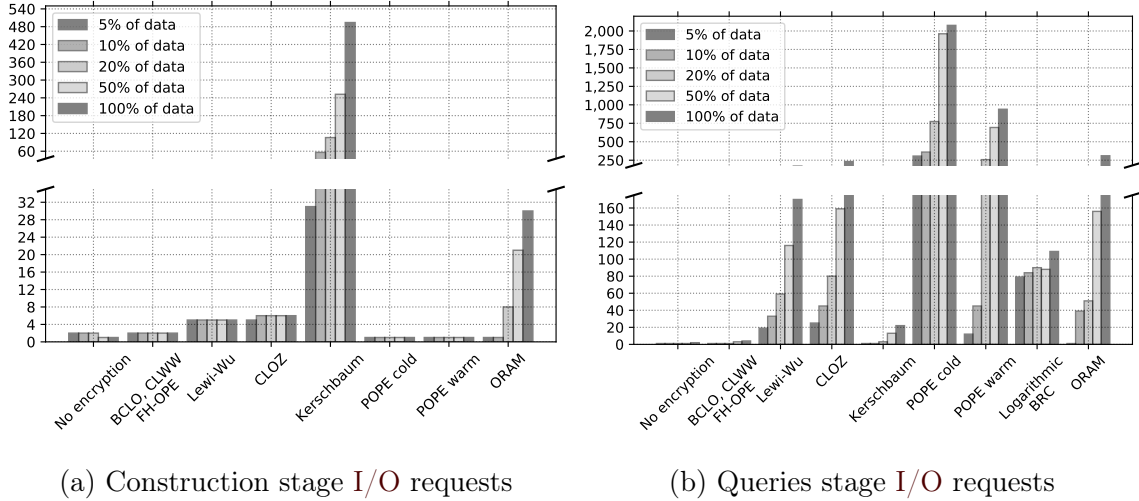


Figure 4-6: Scalability: number of I/O requests

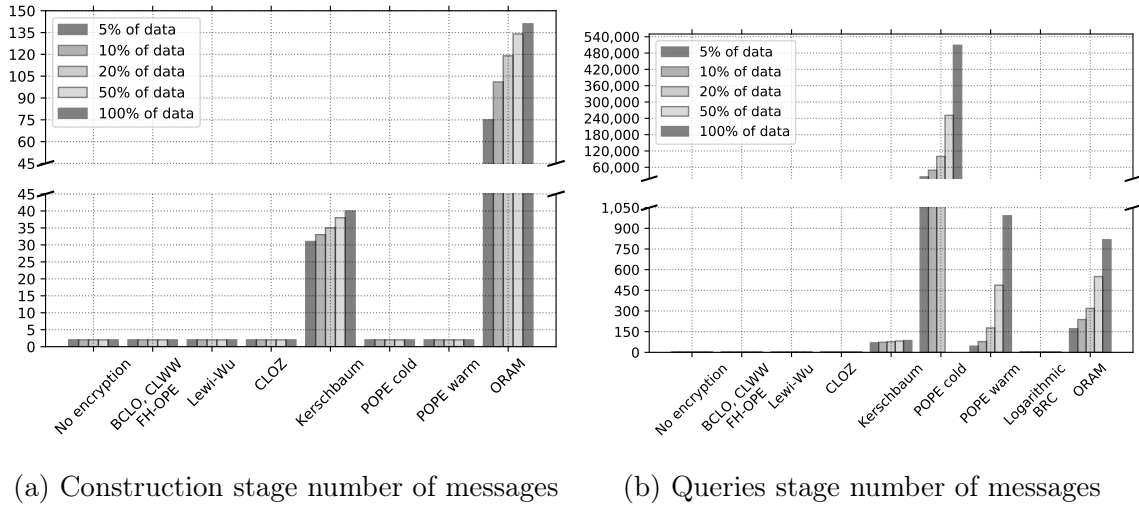


Figure 4-7: Scalability: communication volume

Logarithmic-BRC does not support interactive insertions and thus its construction stage is not benchmarked. Otherwise it is the most performant of all non-ORE protocols. Note, however, that its performance depends on the result size, not data size.

As expected, ORAM performs worse than the ORE-based protocols, but its performance is in-line with the non-ORE protocols. It may seem that ORAM does

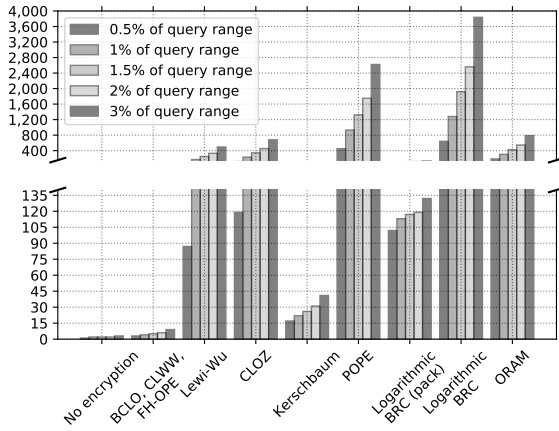


Figure 4-8: Performance for different query sizes

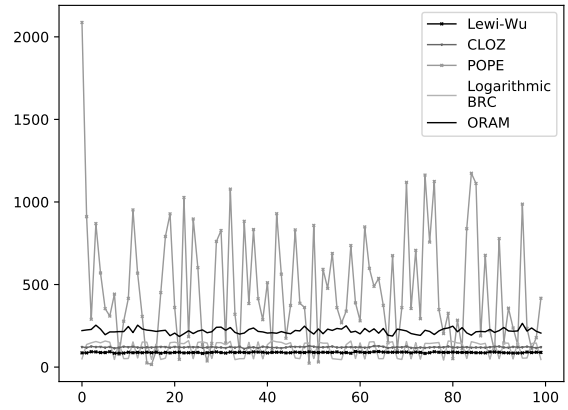


Figure 4-9: Performance over time (queries)

especially bad in construction communication (Figures 4-4b and 4-5b), but it is only because POPE has a shortcut in construction. This “debt” is being payed off during queries (Figure 4-4b).

Note that the values do not vary a lot among different data distributions except for I/O requests. I/O performance depends on the result size for queries, and is therefore more sensitive to data distribution.

Also note that using an ORE scheme with relatively small ciphertext in B+ tree does not add any substantial I/O overhead (see “No encryption”).

On Figure 4-8 it is clear that query performance does not depend substantially on the query size, except for Logarithmic-BRC, for which the relation is linear. Note that Logarithmic-BRC with optimally configured pack extension shows almost no growth. This is because for large ranges BRC will return the higher nodes (keywords matching many documents), which are optimally packed in I/O pages. As query range doubles, higher nodes are involved increasing the chance that requested keywords have their documents packed.

Figures 4-6 and 4-7 show Table 4.2 asymptotic values. The simulation was run for uniform dataset of 247 000 records (hundred percent), 1 000 queries, 0.5 % query

range and 128 blocks cache size. Kerschbaum construction I/Os and cold POPE query values grow linearly with inputs, while the other protocols grow logarithmically, square-logarithmically, or do not grow.

Figure 4.9 shows how the performance of protocols fluctuates as queries are processed. Note that POPE and Logarithmic-BRC fluctuate the most (which is, in general, undesirable), and POPE is the only protocol where cold versus warm makes a difference.

## 4.6 Remarks and conclusion

Having done theoretical and practical evaluations of the protocols, we have found that primitive usage is a much better performance measure than the plain time measurements. When it comes to practical use, the observed time of a query execution is a mix of a number of factors and I/O requests can slow the system down dramatically.

ORE-based B<sup>+</sup> tree protocol is provably I/O optimal and can potentially be extended by using another data structure with ORE. Its security/performance trade off is tunable by choosing and parametrizing the underlying ORE scheme. Each scheme we considered has its own unique advantages and drawbacks. BCLO [BCLO09] is the least secure scheme in the benchmark, but is stateless and produces numerical ciphertexts, so it may be used in the databases without any modifications. Frequency-hiding OPE [Ker15] also has this property, hides the frequency of the ciphertexts, but is stateful and requires uniform input. Lewi-Wu [LW16] is easily customizable in terms of tuning performance to security ratio, and it offers the security benefits of left / right framework — particularly useful for B<sup>+</sup> tree. CLWW [CLWW16] provides weaker security guarantees but is the fastest scheme in the benchmark.

Kerschbaum protocol [KT19] offers semantically secure ciphertexts, hiding the location of the smallest and largest of them, and has a simple implementation. The protocol is well-suited for bulk insertions and scales well.

POPE [RACY16] offers a “deferred” B+ tree implementation. By deferring the sorting of its ciphertexts, POPE remains more secure for the small number of queries. POPE has the fastest insertion routine and does not reveal the order of most of its ciphertexts. It will be more performant for the systems where there are a lot more insertions than queries. We would also recommend to “warm up” the structure to avoid a substantial delay upon the first query.

Logarithmic-BRC is a perfect choice for huge datasets where query result size is limited. It is the only protocol with substantial space overhead, but it offers scalability and perfect (in a snapshot setting) security, and a carefully chosen and configured SSE scheme ensures that I/O grows slowly as a function of result size.

ORAM has shown the most interesting result. Its performance is not only adequate, but also in-line with the other even less secure protocols. With this empirical result, we expect more interest in ORAM research, possibly discovering tighter bounds, faster constructions and efficient ways to use the schemes. The performance of ORAM gives an upper bound on the acceptable performance level of less secure (access pattern revealing) protocols, as practitioners will choose ORAM over both less secure and less performant solutions.

We found our framework to be a powerful tool for analyzing the protocols, and we hope developers of new protocols will contribute implementations and evaluate them.

An important future work is to understand better the meaning of the different leakage profiles and their implications. Furthermore, another direction is to try to improve the performance of the most secure schemes (e.g. [Cas+18]).

## Chapter 5

# Range queries in the persistent model

To protect the data in the outsourced database systems, various cryptographic techniques are used to ensure data privacy, while allowing efficient querying. A rich collection of attacks on such systems has emerged. Even with strong cryptography, just communication volume or access pattern is enough for an adversary to succeed.

In this chapter we present a model for differentially private outsourced database system and a concrete construction,  $\mathcal{E}$ solute, that provably conceals the aforementioned leakages, while remaining efficient and scalable. In our solution, differential privacy is preserved at the record level even against an untrusted server that controls data and queries.  $\mathcal{E}$ solute combines **Oblivious Random Access Machine** and differentially private sanitizers to create a generic and efficient construction.

We go further and present a set of improvements to bring the solution to efficiency and practicality necessary for real-world adoption. We describe the way to parallelize the operations, minimize the amount of noise, and reduce the number of network requests, while preserving the privacy guarantees. We have run an extensive set of experiments, dozens of servers processing up to 10 million records, and compiled a detailed result analysis proving the efficiency and scalability of our solution. While providing strong security and privacy guarantees we are less than an order of magnitude slower than range query execution of a non-secure plain-text optimized RDBMS like MySQL and PostgreSQL.

*Some of the following sections were paraphrased or taken verbatim from the following published work.*

[Bog+21] **Dmytro Bogatov**, Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O’Neill. “ $\epsilon$ solute: 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](https://doi.org/10.1145/3460120.3484786)

## 5.1 Introduction

Secure outsourced database systems aim at helping organizations outsource their data to untrusted third parties, without compromising data confidentiality or query efficiency. The main idea is to encrypt the data records before uploading them to an untrusted server along with an index data structure that governs which encrypted records to retrieve for each query. While strong cryptographic tools can be used for this task, existing implementations such as CryptDB [PRZB11], Cipherbase [Ara+13], StealthDB [VGG19] and TrustedDB [BS13] try to optimize performance but do not provide strong security guarantees when answering queries. Indeed, a series of works [HMCK12; IKK14; CGPR15; NKW15; KKNO16; Bin+18; GRS17; IKK12; LMP18] demonstrate that these systems are vulnerable to a variety of reconstruction attacks. That is, an adversary can fully reconstruct the distribution of the records over the domain of the indexed attribute. This weakness is prominently due to the *access pattern leakage*: the adversary can tell if the same encrypted record is returned on different queries.

More recently, [KKNO16; KPT20; LMP18; GLMP18; GJW19] showed that reconstruction attacks are possible even if the systems employ heavyweight cryptographic techniques that hide the access patterns, such as homomorphic encryption [Gen10; Vai11] or Oblivious Random Access Machine (ORAM) [Gol87; GO96], because they

leak the size of the result set of a query to the server (this is referred to as *communication volume leakage*). Thus, even some recent systems that provide stronger security guarantees like OblIDB [EZ19], Opaque [Zhe+17] and Oblix [Mis+18] are susceptible to these attacks. This also means that no outsourced database system can be both optimally efficient and privacy-preserving: secure outsourced database systems should not return the exact number of records required to answer a query.

We take the next step towards designing secure outsourced database systems by presenting novel constructions that strike a provable balance between efficiency and privacy. First, to combat the access pattern leakage, we integrate a layer of **ORAM** storage in our construction. Then, we bound the communication volume leakage by utilizing the notion of **Differential Privacy (DP)** [DMNS06]. Specifically, instead of returning the exact number of records per query, we only reveal perturbed query answer sizes by adding random encrypted records to the result so that the communication volume leakage is bounded. Our construction guarantees privacy of any single record in the database which is necessary in datasets with stringent privacy requirements. In a medical **HIPAA**-compliant setting, for example, disclosing that a patient exists in a database with a rare diagnosis correlating with age may be enough to reveal a particular individual.

The resulting mechanism achieves the required level of privacy, but implemented naïvely the construction is prohibitively slow. We make the solution practical by limiting the amount of noise and the number of network roundtrips while preserving the privacy guarantees. We go further and present a way to parallelize the construction, which requires adapting noise-generation algorithms to maintain differential privacy requirements.

Using our system, we have run an extensive set of experiments over cloud machines, utilizing large datasets — that range up to 10 million records — and queries

of different sizes, and we report our experimental results on efficiency and scalability. We compare against best possible solutions in terms of efficiency (conventional non-secure outsourced database systems on unencrypted data) and against an approach that provides optimal security (retrieves the full table from the cloud or runs the entire query obliviously with maximal padding). We report that our solution is very competitive against both baselines. Our performance is comparable to that of unsecured plain-text optimized database systems (like MySQL and PostgreSQL): while providing strong security and privacy guarantees, we are only 4 to 8 times slower in a typical setting. Compared with the optimally secure solution, a linear scan (downloading all the records), we are 18 times faster in a typical setting and even faster as database sizes scale up.

To summarize, our contributions in this work are as follows:

- We present a new model for a *differentially private* outsourced database system, **CDP-ODB**, its security definition, query types, and efficiency measures. In our model, the adversarial honest-but-curious server cannot see the record values, access patterns, or exact communication volume.
- We describe a novel construction,  $\mathcal{E}$ solute, that satisfies the proposed security definition, and provide detailed algorithms for both range and point query types. In particular, to conceal the access pattern and communication volume leakages, we provide a secure storage construction, utilizing a combination of **Oblivious Random Access Machine** [Gol87; GO96] and differentially private sanitization [BLR13]. Towards this, we maintain an index structure to know how many and which objects we need to retrieve. This index can be stored locally for better efficiency (in all our experiments this is the case), but crucially, it can also be outsourced to the adversarial server and retrieved on-the-fly for each query.



- We improve our generic construction to enable parallelization within a query. The core idea is to split the storage among multiple **ORAMs**, but this requires tailoring the overhead required for differential privacy proportionally to the number of **ORAMs**, in order to ensure privacy. We present practical improvements and optimization techniques that dramatically reduce the amount of fetched noise and the number of network roundtrips.
- Finally, we provide and open-source a high-quality C++ implementation of our system. We have run an extensive set of experiments on both synthetic and real datasets to empirically assess the efficiency of our construction and the impact of our improvements. We compare our solutions to the naïve approach (linear scan downloading all data every query), oblivious processing and maximal padding solution (Shrinkwrap [Bat+18]), and to a non-secure regular **RDBMS** (PostgreSQL and MySQL), and we show that our system is very competitive.

## 5.2 Differentially private outsourced database systems

In this section we present our model, *differentially private outsourced database system*, **CDP-ODB**, its security definition, query types and efficiency measures. It is an extension of the outsourced database model in Section 1.1.1.

### 5.2.1 Adversarial model

We consider an honest-but-curious polynomial time adversary that attempts to breach differential privacy with respect to the input database  $\mathcal{D}$ . We observe later in Section 5.2.1 that it is impossible to completely hide the number of records returned on each query without essentially returning all the database records on each query. This, in turn, means that different query sequences may be distinguished, and, furthermore, that differential privacy may not be preserved if the query sequence depends on the

content of the database records. We hence, only require the protection of differential privacy with respect to every fixed query sequence. Furthermore, we relax to computational differential privacy (following [MPRV09]).

In the following definition, the notation  $\text{VIEW}_{\Pi}(\mathcal{D}, q_1, \dots, q_m)$  denotes the view of the server  $\mathcal{S}$  in the execution of protocol  $\Pi$  in answering queries  $q_1, \dots, q_m$  with the underlying database  $\mathcal{D}$ .

**Definition 5.2.1.** *We say that an outsourced database system  $\Pi$  is  $(\epsilon, \delta)$ -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  $q_1, \dots, q_m \in \mathcal{Q}^m$  where  $m = \text{poly}(\lambda)$ ,*

$$\Pr[\mathcal{A}(1^\lambda, \text{VIEW}_{\Pi}(\mathcal{D}, q_1, \dots, q_m)) = 1] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(1^\lambda, \text{VIEW}_{\Pi}(\mathcal{D}', q_1, \dots, q_m)) = 1] + \delta + \text{negl}(\lambda),$$

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

**Remark 5.2.1** (Informal). *We note that security and differential privacy in this model imply protection against communication volume and access pattern leakages and thus prevent a range of attacks, such as [CGPR15; NKW15; KKNO16].*

### On impossibility of adaptive queries

Non-adaptivity in our **CDP-ODB** definition does not reflect a deficiency of our specific protocol but rather an inherent source of leakage when the queries may depend on the decrypted data. Consider an adaptive **CDP-ODB** definition that does not fix the query sequence  $q_1, \dots, q_m$  in advance but instead an arbitrary (efficient) user  $\mathcal{U}$  chooses them during the protocol execution with  $\mathcal{S}$ . As before, we ask that the  $\mathcal{S}$ 's view is **DP** on neighboring databases for every such  $\mathcal{U}$ . We observe that this definition cannot possibly be satisfied by *any* outsourced database system without unacceptable efficiency overhead. Note that non-adaptivity here does not imply that

the client knows all the queries in advance, but rather can choose them at any time (e.g., depending on external circumstances) as long as they do not depend on true answers to prior queries.

To see this, consider two neighboring databases  $\mathcal{D}, \mathcal{D}'$ . Database  $\mathcal{D}$  has 1 record with `key = 0` and  $\mathcal{D}'$  has none. Furthermore, both have 50 records with `key = 50` and 100 records with `key = 100`. User  $\mathcal{U}$  queries first for the records with `key = 0`, and then if there is a record with `key = 0` it queries for the records with `key = 50`, otherwise for the records with `key = 100`. Clearly, an efficient outsourced database system cannot return nearly as many records when `key = 50` versus `key = 100` here. Hence, this allows distinguishing  $\mathcal{D}, \mathcal{D}'$  with probability almost 1.

To give a concrete scenario, 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), and if there is a record, she queries the senior patients next (range query, `age  $\geq$  65`), otherwise she queries the general population (resulting in more records). We leave it open to meaningfully strengthen our definition while avoiding such impossibility results, and we defer the formal proof to future work.

### 5.2.2 Query types

In this work we are concerned with the following query types:

**Range queries** Here we assume a total ordering on  $\mathcal{X}$ . A query  $q_{[a,b]}$  is associated with an interval  $[a, b]$  for  $1 \leq a \leq b \leq N$  such that  $q_{[a,b]}(c) = 1$  iff  $c \in [a, b]$  for all  $c \in \mathcal{X}$ . The equivalent SQL query is:

```
SELECT * FROM table WHERE attribute BETWEEN a AND b;
```

**Algorithm 1**  $\mathcal{E}$ psolute protocol.  $\text{ORAM}(\cdot)$  denotes an execution of  $\text{ORAM}$  protocol (Section 2.2), where  $\mathcal{U}$  plays the role of the client.  $\text{ORAM}$  protocol client and server states are implicit.  $S \setminus T$  represents a set of valid record IDs  $S$  that are not in the true result set  $T$ .

$\Pi_{\text{setup}}$	
1: <b>User</b> $\mathcal{U}$	<b>Server</b> $\mathcal{S}$
2: <b>Input:</b> $\mathcal{D}$	<b>Input:</b> $\emptyset$
3: $\mathcal{I} \leftarrow \text{CREATEINDEX}(\mathcal{D})$	
4: $\mathbf{y} = (\mathbf{w}, r_i^{\text{ID}}, r_i) \Big _{i=1}^n$	
5:	$\xleftrightarrow{\text{ORAM}(\mathbf{y})}$
6: $\mathcal{DS} \leftarrow A(\text{SK}_1, \dots, \text{SK}_N)$	$\xrightarrow{\mathcal{DS}}$
7: <b>Output:</b> $\mathcal{I}$	<b>Output:</b> $\mathcal{DS}$
$\Pi_{\text{query}}$	
1: <b>User</b> $\mathcal{U}$	<b>Server</b> $\mathcal{S}$
2: <b>Input:</b> $q, \mathcal{I}$	<b>Input:</b> $\mathcal{DS}$
3: $T \leftarrow \text{LOOKUP}(\mathcal{I}, q)$	$\xrightarrow{q}$
4: $\mathbf{y}_{\text{true}} = (\mathbf{r}, r_i^{\text{ID}}, \perp) \Big _{i \in T}$	$\xleftarrow{c}$
5: $\mathbf{y}_{\text{noise}} = (\mathbf{r}, S \setminus T, \perp) \Big _1^{c- T }$	
6: $R$	$\xleftrightarrow{\text{ORAM}(\mathbf{y}_{\text{true}} \parallel \mathbf{y}_{\text{noise}})}$
7: <b>Output:</b> $R$	<b>Output:</b> $\emptyset$

**Point queries** Here  $\mathcal{X}$  is arbitrary and a query predicate  $q_a$  is associated with an element  $a \in \mathcal{X}$  such that  $q_a(b) = 1$  iff  $a = b$ . In an ordered domain, point queries are degenerate range queries. The equivalent  $\text{SQL}$  query is:

```
SELECT * FROM table WHERE attribute = a;
```

### 5.2.3 Measuring Efficiency

We define two basic efficiency measures for a  $\text{CDP-ODB}$ .

**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. Specifically, we say that an outsourced database system has *storage efficiency* of  $(a_1, a_2)$  if the following holds. Fix any  $\mathcal{D} = \{(r_1, r_1^{\text{ID}}, \text{SK}_1), \dots, (r_n, r_n^{\text{ID}}, \text{SK}_n)\}$  and let  $n_1 = \sum_{i=1}^n |r_i|$ . Let  $\mathcal{S}_{\text{state}}$  be an output of  $\mathcal{S}$  on a run of  $\Pi_{\text{setup}}$  where  $\mathcal{U}$  has input  $\mathcal{D}$ , and let  $n_2 = |\mathcal{S}_{\text{state}}|$ . Then  $n_2 \leq a_1 n_1 + a_2$ .

**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. Specifically, we say that an outsourced database system has *communication efficiency* of  $(a_1, a_2)$  if the following holds. Fix any  $q$  and  $\mathcal{DS}$  output by  $\Pi_{\text{setup}}$ , let  $\mathcal{U}$  and  $\mathcal{S}$  execute  $\Pi_{\text{query}}$  where  $\mathcal{U}$  has inputs  $q$ , and output  $R$ , and  $\mathcal{S}$  has input  $\mathcal{DS}$ . Let  $m_1$  be the amount of data in bits transferred between  $\mathcal{U}$  and  $\mathcal{S}$  during the execution of  $\Pi_{\text{query}}$ , and let  $m_2 = |R|$ . Then  $m_2 \leq a_1 m_1 + a_2$ .

Note that  $a_1 \geq 1$  and  $a_2 \geq 0$  for both measures. We say that an outsourced database system is *optimally storage efficient* (resp., *optimally communication efficient*) if it has storage (resp., communication) efficiency of  $(1, 0)$ .

### 5.3 $\mathcal{E}$ psolute

In this section we present a construction,  $\mathcal{E}$ psolute, that satisfies the security definition in Section 5.2, detailing algorithms for both range and point query types. We also provide efficiency guarantees for approximate and pure DP versions of  $\mathcal{E}$ psolute.

#### 5.3.1 General construction

Let  $\mathcal{Q}$  be a collection of queries. We are interested in building a differentially private outsourced database system for  $\mathcal{Q}$ , called  $\mathcal{E}$ psolute. Our solution will use these building blocks.

- A  $(\eta_1, \eta_2)$ -ORAM protocol  $\text{ORAM}(\cdot)$ .
- An  $(\epsilon, \delta, \alpha, \beta)$ -differentially private sanitizer  $(A, B)$  for  $\mathcal{Q}$  and negligible  $\beta$ , which satisfies the non-negative noise guarantee from Remark 2.3.1.
- A pair of algorithms  $\text{CREATEINDEX}$  and  $\text{LOOKUP}$ .  $\text{CREATEINDEX}$  consumes  $\mathcal{D}$  and produces an index data structure  $\mathcal{I}$  that maps a search key  $\text{SK}$  to a list of record IDs  $r^{\text{ID}}$  corresponding to the given search key.  $\text{LOOKUP}$  consumes  $\mathcal{I}$  and  $q$  and returns a list  $T = r_1^{\text{ID}}, \dots, r_{|T|}^{\text{ID}}$  of record IDs matching the supplied query.

Our protocol  $\Pi = (\Pi_{\text{setup}}, \Pi_{\text{query}})$  of  $\mathcal{E}$ psolute works as shown in Algorithm 1. Hereafter, we reference lines in Algorithm 1. See Figure 5.1 for a schematic description of the protocol.

**Setup protocol**  $\Pi_{\text{setup}}$  Let  $\mathcal{U}$ 's input be a database  $\mathcal{D} = \{(r_1, r_1^{\text{ID}}, \text{SK}_1), \dots, (r_n, r_n^{\text{ID}}, \text{SK}_n)\}$  (line 2).  $\mathcal{U}$  creates an index  $\mathcal{I}$  mapping search keys to record IDs corresponding to these keys (line 3).  $\mathcal{U}$  sends over the records to  $\mathcal{S}$  by executing the **ORAM** protocol on the specified sequence (lines 4 to 5).  $\mathcal{U}$  generates a **DP** structure  $\mathcal{DS}$  over the search keys using sanitizer  $A$ , and sends  $\mathcal{DS}$  over to  $\mathcal{S}$  (line 6). The output of  $\mathcal{U}$  is  $\mathcal{I}$  and of  $\mathcal{S}$  is  $\mathcal{DS}$ ; final **ORAM** states of  $\mathcal{S}$  and  $\mathcal{U}$  are implicit, including encryption key  $K$  (line 7).

**Query protocol**  $\Pi_{\text{query}}$   $\mathcal{U}$  starts with a query  $q$  and index  $\mathcal{I}$ ,  $\mathcal{S}$  starts with a **DP** structure  $\mathcal{DS}$ . One can think of these inputs as outputs of  $\Pi_{\text{setup}}$  (line 2).  $\mathcal{U}$  immediately sends the query to  $\mathcal{S}$ , which uses the sanitizer  $B$  to compute the total number of requests  $c$ , while  $\mathcal{U}$  uses index  $\mathcal{I}$  to derive the true indices of the records the query  $q$  targets (line 3).  $\mathcal{U}$  receives  $c$  from  $\mathcal{S}$  and prepares two **ORAM** sequences:  $\mathbf{y}_{\text{true}}$  for real records retrieval, and  $\mathbf{y}_{\text{noise}}$  to pad the number of requests to  $c$  to perturb

the communication volume.  $\mathbf{y}_{\text{noise}}$  includes valid non-repeating record IDs that are not part of the true result set  $T$  (lines 4 to 5).  $\mathcal{U}$  fetches the records, both real and fake, from  $\mathcal{S}$  using the **ORAM** protocol (line 6). The output of  $\mathcal{U}$  is the filtered set of records requested by the query  $q$ ; final **ORAM** states of  $\mathcal{S}$  and  $\mathcal{U}$  are implicit (line 7).

The protocols for point and range queries only differ in sanitizer implementations, see Sections 5.3.5 and 5.3.6. Note above that in any execution of  $\Pi_{\text{query}}$  we have  $c \geq q(\mathcal{D})$  with overwhelming probability  $1 - \beta$  (by using sanitizers satisfying Remark 2.3.1), and thus the protocol is well-defined and its accuracy is  $1 - \beta$ . Also note that the **DP** parameter  $\delta$  is lower-bounded by  $\beta$  because sampling negative noise, however improbable, violates privacy, and therefore the final construction is  $(\epsilon, \beta)$ -**DP**.

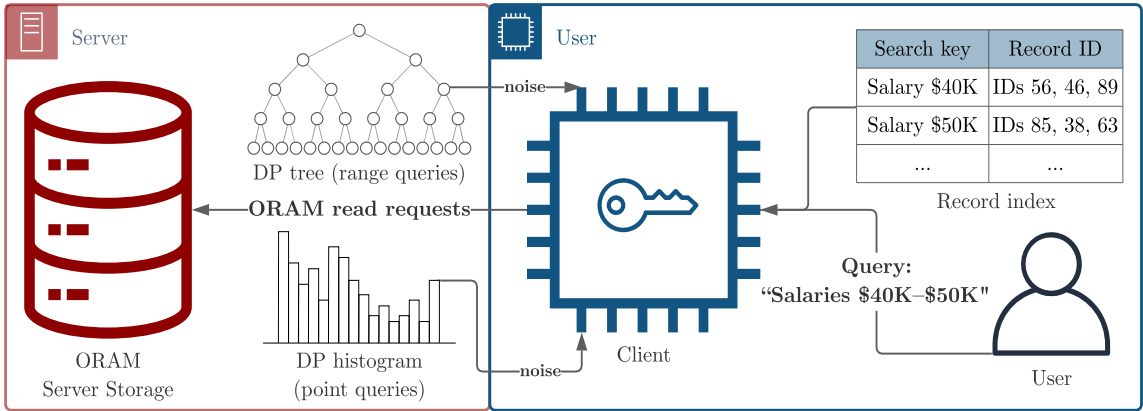


Figure 5.1:  $\mathcal{E}$ psolute construction

### 5.3.2 Security

**Theorem 5.3.1.**  *$\mathcal{E}$ psolute is  $(\beta \cdot m)$ -wrong and  $(\epsilon, \delta)$ -**CDP-ODB** where the negligible term is  $\text{negl}(\lambda) = 2 \cdot \eta_2$ .*

*Proof.* We consider a sequence of views

$$\text{VIEW}_1 \rightarrow \text{VIEW}_2 \rightarrow \text{VIEW}_3 \rightarrow \text{VIEW}_4 .$$

$\text{VIEW}_1$  is  $\text{VIEW}_{\Pi}(\mathcal{D}, q_1, \dots, q_m)$ .  $\text{VIEW}_2$  is produced only from  $\mathcal{DS} \leftarrow A(\text{SK}_1, \dots, \text{SK}_N)$ . Namely, compute  $c_i \leftarrow A(\mathcal{DS}, q_i)$  for all  $i$  and run **ORAM** simulator on  $\sum_i c_i$ .

By **ORAM** security,

$$\Pr [\mathcal{A}(\text{VIEW}_1)] - \Pr [\mathcal{A}(\text{VIEW}_2)] \leq \eta_2 .$$

$\text{VIEW}_3$  is produced similarly but  $\mathcal{DS} \leftarrow A(\text{SK}'_1, \dots, \text{SK}'_N)$  instead. Note that the  $c_i$  are simply post-processing on  $\mathcal{DS}$  via  $B$  so

$$\Pr [\mathcal{A}(\text{VIEW}_2)] = \exp(\epsilon) \cdot \Pr [\mathcal{A}(\text{VIEW}_3)] + \delta .$$

$\text{VIEW}_4 = \text{VIEW}_\Pi(\mathcal{D}', q_1, \dots, q_m)$ . It follows by **ORAM** security

$$\Pr [\mathcal{A}(\text{VIEW}_3)] - \Pr [\mathcal{A}(\text{VIEW}_4)] \leq \eta_2 .$$

Putting this all together completes the proof.  $\square$

### 5.3.3 Efficiency

For an **ORAM** with communication efficiency  $(a_1, a_2)$  and an  $(\alpha, \beta)$ -differentially private sanitizer, the  $\mathcal{E}$ psolute communication efficiency is  $(a_1, a_2 \cdot \alpha)$ . The efficiency metrics demonstrate how the total storage or communication volume (the number of stored or transferred bits) changes additively and multiplicatively as the functions of data size  $n$  and domain  $N$ . We therefore have the following corollaries for the efficiency of the system in the cases of approximate and pure differential privacy.

**Corollary 5.3.2.**  *$\mathcal{E}$ psolute is an outsourced database system with storage efficiency  $(\mathcal{O}(1), 0)$ . Depending on the query type, assume it offers the following communication efficiency.*

**Range queries**  $(\mathcal{O}(\log n), \mathcal{O}(2^{\log^* N} \log n))$

**Point queries**  $(\mathcal{O}(\log n), \mathcal{O}(\log n))$

*Then, there is a negligible  $\delta$  such that  $\mathcal{E}$ psolute satisfies  $(\epsilon, \delta)$ -differential privacy for some  $\epsilon$ .<sup>1</sup>*

---

<sup>1</sup>Note that the existence of  $\epsilon$  in this setting implies that the probability of an adversary breaking the **DP** guarantees is bounded by it.



*Proof.* By using **ORAM**, we store only the original data once and hence, we get optimal storage efficiency.

The communication efficiency depends on the upper bound of the error for each sanitizer when  $\delta > 0$ , as described in Section 2.3.1 and Remark 2.3.1. The most efficient **ORAM** protocol to date has  $\mathcal{O}(\log n)$  communication overhead (see Section 2.2).  $\square$

**Corollary 5.3.3.**  *$\mathcal{E}$ psolute is an outsourced database system with storage efficiency  $(\mathcal{O}(1), 0)$ . Depending on the query type, assume it offers the following communication efficiency.*

**Range queries**  $(\mathcal{O}(\log n), \mathcal{O}(\log N \log n))$

**Point queries**  $(\mathcal{O}(\log n), \mathcal{O}(\log N \log n))$

*Then,  $\mathcal{E}$ psolute satisfies  $\epsilon$ -differential privacy for some  $\epsilon$ .*

*Proof.* Similarly, we derive the proof by considering the use of **ORAM** and the upper bound of the error for each sanitizer when  $\delta = 0$  in Section 2.3.1.  $\square$

### 5.3.4 Extending to multiple attributes

We will now describe how  $\mathcal{E}$ psolute supports multiple indexed attributes and what the privacy and performance implications are. The naïve way is to simply duplicate the entire stack of states of  $\mathcal{U}$  and  $\mathcal{S}$ , and during the query use the states whose attribute the query targets. However,  $\mathcal{E}$ psolute design allows to keep the most expensive part of the state — the **ORAM** state — shared for all attributes and both types of queries. Specifically, the index  $\mathcal{I}$  and **DP** structure  $\mathcal{DS}$  are generated per attribute and query type, while  $\mathcal{U}$  and  $\mathcal{S}$  **ORAM** states are generated once. This design is practical since  $\mathcal{DS}$  is tiny and index  $\mathcal{I}$  is relatively small compared to **ORAM** states, see Section 5.5.

We note that in case the indices grow large in number, it is practical to outsource them to the adversarial server using **ORAM** and download only the ones needed for each query. In terms of privacy, the solution is equivalent to operating different  $\mathcal{E}$ psolute instances because **ORAM** hides the values of records and access patterns

entirely. Due to Theorem 2.3.1 for non-disjoint datasets, the total privacy budget of the multi-attribute system will be the sum of individual budgets for each attribute / index.

Next, we choose two DP sanitizers for our system, for point and for range queries, and calculate the  $\alpha$  values to make them output positive values with high probability, consistent with Remark 2.3.1.

### 5.3.5 $\epsilon$ psolute for point queries

For point queries, we use the LPA method as the sanitizer to ensure pure differential privacy. Specifically, for every histogram bin, we draw noise from the Laplace distribution with mean  $\alpha_p$  and scale  $\lambda = 1/\epsilon$ . To satisfy Remark 2.3.1, we have to set  $\alpha_p$  such that if values are drawn from  $\text{LAPLACE}(\alpha_p, 1/\epsilon)$  at least as many times as the number of bins  $N$ , they are all positive with high probability  $1 - \beta$ , for negligible  $\beta$ .

We can compute the exact minimum required value of  $\alpha_p$  in order to ensure drawing positive values with high probability by using the CDF of the Laplace distribution. Specifically,  $\alpha_p$  should be equal to the minimum value that satisfies the following inequality.

$$\left(1 - \frac{1}{2}e^{-\alpha_p \cdot \epsilon}\right)^N \leq 1 - \beta$$

which is equivalent to

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

### 5.3.6 $\epsilon$ psolute for range queries

For range queries, we implement the aggregate tree method as the sanitizer. Specifically, we build a complete  $k$ -ary tree on the domain, for a given  $k$ . A leaf node holds the number of records falling into each bin plus some noise. A parent node holds sum

**Algorithm 2** Parallel  $\mathcal{E}$ psolute for  $\Pi_\gamma$ , extends Algorithm 1.  $H$  is a random hash function  $H : \{0, 1\}^* \rightarrow \{1, \dots, m\}$ .  $\gamma$  and  $\tilde{k}_0$  are computed as in Section 5.4.2.

$\Pi_{\text{setup}}$  of  $\Pi_\gamma$

1: <b>User</b> $\mathcal{U}$	<b>Server</b> $\mathcal{S}$
2: <b>Input:</b> $\mathcal{D}$	<b>Input:</b> $\emptyset$
3: $\mathcal{I} \leftarrow \text{CREATEINDEX}(\mathcal{D}, m)$	
..... <b>for</b> $j \in \{1, \dots, m\}$ <b>do</b> (in parallel).....	
4: $\langle \bar{r}, r^{\text{ID}} \rangle$ s.t. $H(r^{\text{ID}}) = j$	
5: $\mathbf{y} = \langle (\mathbf{w}, r^{\text{ID}}, \bar{r}) \rangle$	$\xleftarrow{\text{ORAM}_j(\mathbf{y})}$
..... <b>endfor</b> .....	
6: $\mathcal{DS} \leftarrow A(\text{SK}_1, \dots, \text{SK}_N)$	$\xrightarrow{\mathcal{DS}}$
7: <b>Output:</b> $\mathcal{I}$	<b>Output:</b> $\mathcal{DS}$

$\Pi_{\text{query}}$  of  $\Pi_\gamma$

1: <b>User</b> $\mathcal{U}$	<b>Server</b> $\mathcal{S}$	
2: <b>Input:</b> $q, \mathcal{I}$	<b>Input:</b> $\mathcal{DS}$	
3: $T_1, \dots, T_m \leftarrow \text{LOOKUP}(I, q)$	$\xrightarrow{q}$	$k \leftarrow B(\mathcal{DS}, q)$
4:	$\xleftarrow{c}$	$c \leftarrow (1 + \gamma) \frac{\tilde{k}_0}{m}$
..... <b>for</b> $j \in \{1, \dots, m\}$ <b>do</b> (in parallel).....		
5: $\mathbf{y}_{\text{true}} = (\mathbf{r}, r_i^{\text{ID}}, \perp) \Big _{i \in T_j}$		
6: $\mathbf{y}_{\text{noise}} = (\mathbf{r}, S \setminus T_j, \perp) \Big _1^{c -  T_j }$	$\xleftarrow{\text{ORAM}_j(\mathbf{y}_{\text{true}} \parallel \mathbf{y}_{\text{noise}})}$	$R_j$
..... <b>endfor</b> .....		
7: <b>Output:</b> $R_j \Big _{j=1}^m$	<b>Output:</b> $\emptyset$	

of the leaf values in the range covered by this node, plus noise. Every time a query is issued, we find the minimum number of nodes that cover the range, and determine

the required number of returned records by summing these node values. Then, we ask the server to retrieve the records in the range, plus to retrieve multiple random records so that the total number of retrieved records matches the required number of returned records.

The noise per node is drawn from the Laplace distribution with mean  $\alpha_h$  and scale  $\lambda = \frac{\log_k N}{\epsilon}$ . Consistent with Remark 2.3.1, we determine the mean value  $\alpha_h$  in order to avoid drawing negative values with high probability. We have to set  $\alpha_h$  such that if values are drawn from  $\text{LAPLACE}\left(\alpha_h, \frac{\log_k N}{\epsilon}\right)$  at least as many times as the number of nodes in the tree, they are all positive with high probability  $1 - \beta$ , for negligible  $\beta$ .

Again, we can compute the exact minimum required value of  $\alpha_h$  in order to ensure drawing positive values with high probability by using the CDF of the Laplace distribution. Specifically,  $\alpha_h$  should be equal to the minimum value that satisfies the following inequality.

$$\left(1 - \frac{1}{2}e^{-\frac{\alpha_h \cdot \epsilon}{\log_k N}}\right)^{\text{nodes}} \leq 1 - \beta$$

which is equivalent to

$$\alpha_h = \left\lceil -\frac{\ln(2 - 2^{\text{nodes}\sqrt{1-\beta}}) \cdot \log_k N}{\epsilon} \right\rceil \quad (5.1)$$

where  $\text{nodes} = \frac{k^{\lceil \log_k(k-1) + \log_k N - 1 \rceil} - 1}{k-1} + N$  is the total number of tree nodes.

## 5.4 An efficient Parallel $\mathcal{E}$ psolute

While the previously described scheme is a secure and correct CDP-ODB, a single-threaded implementation may be prohibitively slow in practice. To bring the performance closer to real-world requirements, we need to be able to scale the algorithm horizontally. In this section, we describe an upgrade of  $\mathcal{E}$ psolute — a scalable parallel solution.

We suggest two variants of parallel  $\mathcal{E}$ psolute protocol. Both of them work by operating  $m$  ORAMs and randomly assigning to each of them  $n/m$  database records. For each query, we utilize the index  $\mathcal{I}$  to find the required records from the corresponding ORAMs. For each ORAM, we execute a separate thread to retrieve the records. The threads work in parallel and there is no need for locking, since each ORAM works independently from the rest. We present two methods that differ in the way they build and store DP structure  $\mathcal{DS}$ , and hence the number of ORAM requests they make.

#### 5.4.1 No- $\gamma$ -method: DP structure per ORAM

In  $\Pi_{\text{no-}\gamma}$ , for each ORAM / subset of the dataset, we build a DP index the same way as described in Section 5.3. We note that Theorem 2.3.1 for disjoint datasets applies to this construction: the privacy budget  $\epsilon$  for the construction is the largest (least private) among the  $\epsilon$ 's of the DP indices for each ORAM / subset of the dataset.

The communication efficiency changes because (i) we essentially add  $m$  record subsets in order to answer a query, each having at most  $\alpha$  extra random records, and (ii) each ORAM holds fewer records than before, resulting in a tree of height  $\log \frac{n}{m}$ .

However, we cannot expect that the records required for each query are equally distributed among the different ORAMs in order to reduce the multiplicative communication cost from  $\log n$  to  $\frac{\log n}{m}$ . Instead, we need to bound the worst case scenario which is represented by the maximum number of records from any ORAM that is required to answer a query. This can be computed as follows.

Let  $X_j$  be 1 if a record for answering query  $q$  is in a specific ORAM $_j$ , and 0 otherwise. Due to the random assignment of records to ORAMs,  $\Pr[X_j = 1] = 1/m$ . Assume that we need  $k_0$  records in order to answer query  $q$ . The maximum number of records from ORAM $_j$  in order to answer  $q$  is bounded as follows.

$$\Pr \left[ \sum_{i=1}^{k_0} X_i > (1 + \gamma) \frac{k_0}{m} \right] \leq \exp \left( -\frac{k_0 \gamma^2}{3m} \right) \quad (5.2)$$

Finally, we need to determine the value of  $\gamma$  such that  $\exp \left( -\frac{k_0 \gamma^2}{3m} \right)$  is smaller than the value  $\beta$ . Thus,  $\gamma = \sqrt{\frac{-3m \log \beta}{k_0}}$ . The communication efficiency for each query type is described in the following corollary.

**Corollary 5.4.1.** *Let  $\Pi_{\text{no-}\gamma}$  be an outsourced database system with storage efficiency  $(\mathcal{O}(1), 0)$ . Depending on the query type,  $\Pi_{\text{no-}\gamma}$  offers the following communication efficiency.*

**Range queries**  $\left( \mathcal{O} \left( \left( 1 + \sqrt{\frac{-3m \log \beta}{k_0}} \right) \log \frac{n}{m} \right), \mathcal{O} \left( \frac{\log^{1.5} N}{\epsilon} m \log n \right) \right)$

**Point queries**  $\left( \mathcal{O} \left( \left( 1 + \sqrt{\frac{-3m \log \beta}{k_0}} \right) \log \frac{n}{m} \right), \mathcal{O} \left( \frac{\log N}{\epsilon} m \log n \right) \right)$

*Then,  $\Pi_{\text{no-}\gamma}$  satisfies  $\epsilon$ -differential privacy for some  $\epsilon$ .*

In our experiments, we set  $m$  as a constant depending on the infrastructure. However, if  $m$  is set as  $\mathcal{O}(\log n)$ , the total communication overhead of the construction will still exceed the lower-bound presented in [LSY20].

#### 5.4.2 $\gamma$ -method: shared DP structure

In  $\Pi_\gamma$ , we maintain a single shared DP structure  $\mathcal{DS}$ . When a query is issued, we must ensure that the number of records retrieved from every ORAM is the same. As such, depending on the required noisy number of records  $\tilde{k}_0$ , we need to retrieve at most  $(1 + \gamma) \frac{\tilde{k}_0}{m}$  records from each ORAM, see Equation (5.2), for  $\gamma = \sqrt{\frac{-3m \log \beta}{\tilde{k}_0}}$ . Setting  $\tilde{k}_0 = k_0 + \frac{\log^{1.5} N}{\epsilon}$  for range queries and  $\tilde{k}_0 = k_0 + \frac{\log N}{\epsilon}$  for point queries, the communication efficiency is as follows.

**Corollary 5.4.2.** *Let  $\Pi_\gamma$  be an outsourced database system with storage efficiency  $(\mathcal{O}(1), 0)$ . Depending on the query type,  $\Pi_\gamma$  offers the following communication efficiency.*

**Range queries**  $\left( \mathcal{O} \left( \left( 1 + \sqrt{\frac{-3m \log \beta}{k_0 + \frac{\log^{1.5} N}{\epsilon}}} \right) \log \frac{n}{m} \left( 1 + \frac{\log^{1.5} N}{\epsilon} \right) \right), 0 \right)$

**Point queries**  $\left( \mathcal{O} \left( \left( 1 + \sqrt{\frac{-3m \log \beta}{k_0 + \frac{\log N}{\epsilon}}} \right) \log \frac{n}{m} \left( 1 + \frac{\log N}{\epsilon} \right) \right), 0 \right)$

Then,  $\Pi_\gamma$  satisfies  $\epsilon$ -differential privacy for some  $\epsilon$ .

$\Pi_\gamma$  is depicted in Algorithm 2. There are a few extensions to the subroutines and notation from Algorithm 1. CREATEINDEX and LOOKUP now build and query the index which maps a search key to a pair — the record ID and the ORAM ID (1 to  $m$ ) which stores the record. Lines 4 to 5 of Algorithm 2  $\Pi_{\text{setup}}$  repeat for each ORAM and operate on the records partitioned for the given ORAM using hash function H on the record ID. A shared DP structure is created with the sanitizer A (line 6). In Algorithm 2  $\Pi_{\text{query}}$ , the total number of ORAM requests is computed once (line 4). Lines 5 to 6 repeat for each ORAM and operate on the subset of records stored in the given ORAM. Note that  $\mathcal{U}$  and  $\mathcal{S}$  implicitly maintain  $m$  ORAM states, and the algorithm uses the (A, B) sanitizer defined in Section 5.3.

Note that we guarantee privacy and access pattern protection on a record level. Each ORAM gets accessed at least once (much more than once for a typical query) thus the existence of a particular result record in a particular ORAM is hidden.

### 5.4.3 Practical improvements

Here we describe the optimizations aimed at bringing the construction's performance to the real-world demands.

#### ORAM request batching

We have noticed that although the entire set of ORAM requests for each query is known in advance, the requests are still executed sequentially. To address this inefficiency, we have designed a way to combine the requests in a batch and reduce the number of network requests to the bare minimum. We have implemented this method

over PathORAM, which we use for the  $(\eta_1, \eta_2)$ -ORAM protocol, but the idea applies to most tree-based ORAMs (similar to [CLT16]).

Our optimization utilizes the fact that all PathORAM leaf IDs are known in advance and paths in a tree-based storage share the buckets close to the root. The core idea is to read all paths first, processes the requests and then write all paths back. This way the client makes a single `read` request, which is executed much faster than many small requests. Requests are then processed in main memory, including re-encryptions. Finally, the client executes the `write` requests using remapped leaves as a single operation, saving again compared to sequential execution.

This optimization provides up to **8 times** performance boost in our experiments. We note that the gains in speed and I/O overhead are achieved at the expense of main memory, which is not an issue given that the memory is released after a batch, and our experiments confirm that. The security guarantees of PathORAM are maintained with this optimization, since the security proof in [Ste+13, Section 3.6] still holds. Randomized encryption, statistically independent remapping of leaves, and stash processing do not change.

### Lightweight ORAM servers

We have found in our experiments that naïve increase of the number of CPU cores and gigabytes of memory does not translate into linear performance improvement after some threshold. Investigating the observation we have found that the  $\mathcal{E}$ bsolute protocol, executing parallel ORAM protocols, is highly intensive with respect to main memory access, cryptographic operations and network usage. The bottleneck is the hardware — we have confirmed that on a single machine the memory and network are saturated quickly preventing the linear scaling.

To address the problem, we split the user party  $\mathcal{U}$  into multiple lightweight machines that are connected locally to each other and reside in a single trust domain



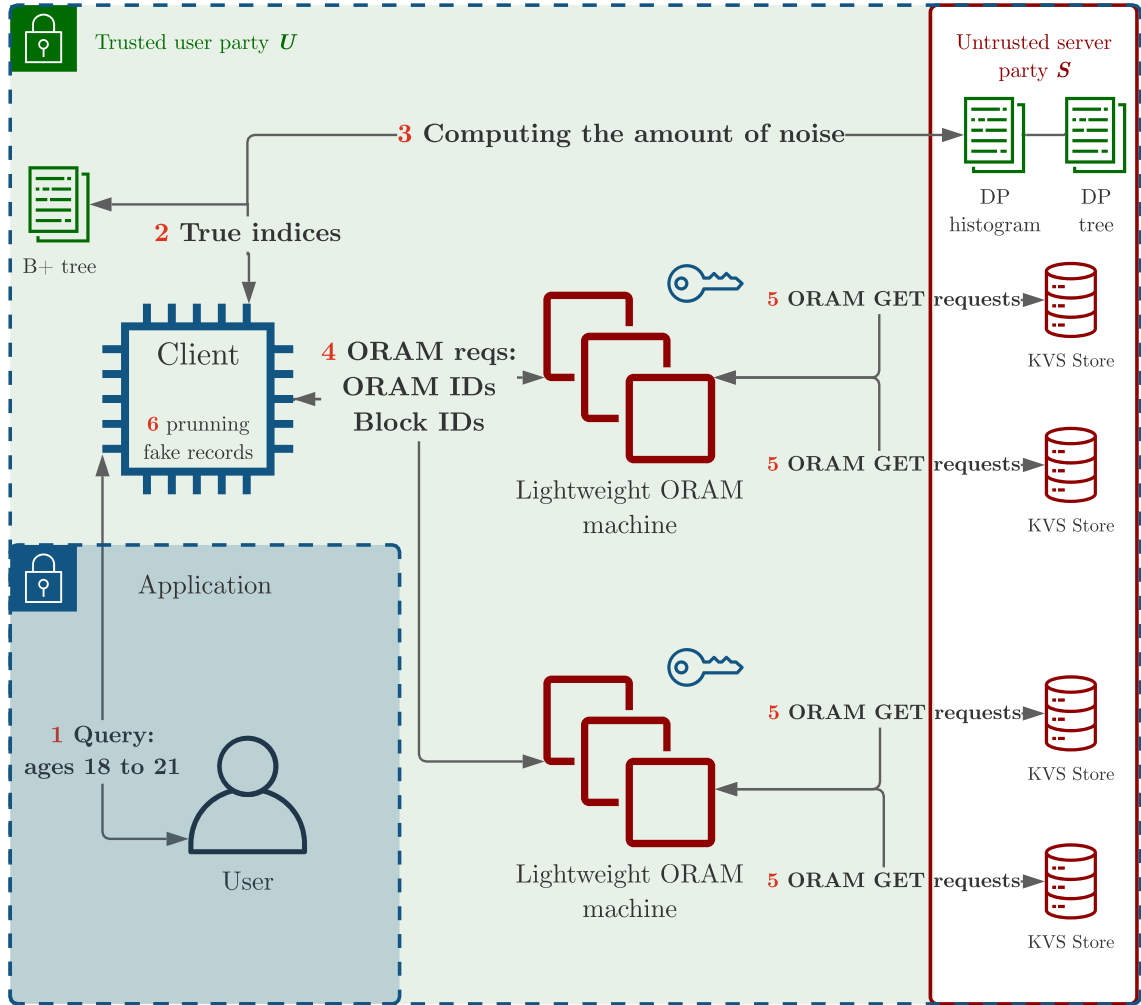


Figure 5.2: Parallel  $\mathcal{E}$ psolute construction. A *user* sends a query to  $\mathcal{U}$  modeled as the *client* machine, which uses local *data index* and *DP structures* to prepare a set of *ORAM* requests, which are sent to respective *ORAM machines*. These machines execute the *ORAM* protocol against the *untrusted storage* of  $\mathcal{S}$ .

(e.g., same data center). Specifically, we maintain a *client machine* that receives user requests and prepares *ORAM* read requests, and up to  $m$  lightweight *ORAM machines*, whose only job is to run the *ORAM* protocols in parallel. See Figure 5.2 for the schematic representation of the architecture. We emphasize that  $\mathcal{U}$  is still a single party, therefore, the security and correctness guarantees remain valid.

The benefit of this approach is that each of the lightweight machines has its own hardware stack. Communication overhead among  $\mathcal{U}$  machines is negligible compared to the one between  $\mathcal{U}$  and  $\mathcal{S}$ . The approach is also flexible: it is possible to use up to  $m$  ORAM machines and the machines do not have to be identical. Our experiments show that when the same number of CPU cores and amount of memory are consumed the efficiency gain is up to **5 times**.

## 5.5 Experimental Evaluation

We have implemented our solution as a modular client-server application in C++. We open-sourced all components of the software set: PathORAM<sup>2</sup> and B+ tree<sup>3</sup> implementations and the main query executor<sup>4</sup>. We provide PathORAM and B+ tree components as C++ libraries to be used in other projects; the code is documented, benchmarked and tested (228 tests covering 100% of the code). We have also published our datasets and query sets.<sup>5</sup>

For cryptographic primitives, we used OpenSSL library (version 1.1.1i). For symmetric encryption in ORAM we have used AES in CBC mode [Dwo+01; Dwo01] with a 256-bits key (i.e.,  $\eta_2 = 2^{-256}$ ), for the hash algorithm H used to partition records among ORAMs we have used SHA-256 algorithm [ST15]. Aggregate tree fanout  $k$  is 16, proven to be optimal in [QYL13].

We designed our experiments to answer the following questions:

**Question-1** How practical is our system compared to the most efficient and most private real-world solutions?

**Question-2** How practical is the storage overhead?

---

<sup>2</sup><https://github.com/epsolute/path-oram>

<sup>3</sup><https://github.com/epsolute/b-plus-tree>

<sup>4</sup><https://github.com/epsolute/epsolute>

<sup>5</sup><http://csr.bu.edu/dp-oram/>

**Question-3** How different inputs and parameters of the system affect its performance?

**Question-4** How well does the system scale?

**Question-5** What improvements do our optimizations provide?

**Question-6** What is the impact of supporting multiple attributes?

For **Question-1** we have run the default setting using conventional RDBMS (MySQL and PostgreSQL), Linear Scan approach and Shrinkwrap [Bat+18]. To target **Question-2**, we measured the exact storage used by the client and the server for different data, record and domain sizes. To answer **Question-3**, we ran a default setting and then varied all parameters and inputs, one at a time. For **Question-4** we gradually added CPUs, ORAM servers and KVS instances and observed the rate of improvement in performance. To target **Question-5** we have run the default setting with our optimizations toggled. Lastly, for **Question-6** we have used two datasets to construct two indices and then queried each of the attributes.

### 5.5.1 Data sets

We used two real and one synthetic datasets — California public pay pension database 2019 [Tra19] (referred to as “CA employees”), Public Use Microdata Sample from US Census 2018 [US 18] (referred to as “PUMS”) and synthetic uniform dataset. We have used salary / wages columns of the real datasets, and the numbers in the uniform set also represent salaries. The NULL and empty values were dropped.

We created three versions of each dataset —  $10^5$ ,  $10^6$  and  $10^7$  records each. For uniform dataset, we simply generated the target number of entries. For PUMS dataset, we picked the states whose number of records most closely matches the target sizes (Louisiana for  $10^5$ , California for  $10^6$  and the entire US for  $10^7$ ). Uniform dataset

was also generated for different domain sizes — number of distinct values for the record. For CA employees dataset, the set contains 260 277 records, so we contracted it and expanded in the following way. For contraction we uniformly randomly sampled  $10^5$  records. For expansion, we computed the histogram of the original dataset and sampled values uniformly within the bins.

Each of the datasets has a number of corresponding query sets. Each query set has a selectivity or range size, and is sampled either uniformly or following the dataset distribution (using its CDF).

### 5.5.2 Default setting

The default setting uses the  $\Pi_\gamma$  from Section 5.4 and lightweight ORAM machines from Section 5.4.3 and Figure 5.2. We choose the  $\Pi_\gamma$  because it outperforms  $\Pi_{\text{no-}\gamma}$  in all experiments (see **Question-4** in Section 5.5.5). In the setting, there are 64 Redis services (8 services per one Redis server VM), 8 ORAM machines communicating with 8 Redis services each, and the client, which communicates with these 8 ORAM machines. We have empirically found this configuration optimal for the compute nodes and network that we used in the experiments. ORAM and Redis servers run on GCP n1-standard-16 VMs (Ubuntu 18.04), in regions us-east4 and us-east1 respectively. Client machine runs n1-highmem-16 VM in the same region as ORAM machines. The ping time between the regions (i.e. between trusted and untrusted zones) is 12 ms and the effective bandwidth is 150 MB/s. Ping within a region is negligible.

Default DP parameters are  $\epsilon = \ln(2) \approx 0.693$  and  $\beta = 2^{-20}$ , which are consistent with the other DP applications in the literature [Hsu+14]. Buckets number is set as the largest power of  $k = 16$  that is no greater than the domain of the dataset  $N$ .

Default dataset is a uniform dataset of  $10^6$  records with domain size  $10^4$ , and uniformly sampled queries with selectivity 0.5%. Default record size is 4 KiB.

### 5.5.3 Experiment stages

Each experiment includes running 100 queries such that the overhead is measured from loading query endpoints into memory to receiving the exact and whole query response from all **ORAM** machines. The output of an experiment is, among other things, the overhead (in milliseconds), the number of real and noisy records fetched and communication volume averaged per query.

### 5.5.4 RDBMS, Linear Scan and Shrinkwrap

On top of varying the parameters, we have run similar workloads using alternative mechanisms — extremes representing highest performance or highest privacy. Unless stated otherwise, the client and the server are in the trusted and untrusted regions respectively, with the network configuration as in Section 5.5.2.

#### Relational databases

Conventional **RDBMS** represents the most efficient and least private and secure solution in our set. While MySQL and PostgreSQL offer some encryption options and no differential privacy, for our experiments we turned off security features for maximal performance. We have run queries against MySQL and PostgreSQL varying data and record sizes. We used **n1-standard-32 GCP VMs** in **us-east1** region, running MySQL version 14.14 and PostgreSQL version 10.14.

#### Linear Scan

Linear scan is a primitive mechanism that keeps all records encrypted on the server then downloads, decrypts and scans the entire database to answer every query. This method is trivially correct, private and secure, albeit not very efficient. There are **RDBMS** solutions, which, when configured for maximum privacy, exhibit linear scan

behavior (e.g., MS-SQL Always Encrypted with Randomized Encryption<sup>6</sup> and Oracle Column Transparent Data Encryption<sup>7</sup>). For a fair comparison we make the linear scan even more efficient by allowing it to download data via parallel threads matching the number of threads and bytes per request to that of our solution. Although linear scan is wasteful in the amount of data it downloads and processes, compared to our solution it has a benefit of not executing an **ORAM** protocol with its logarithmic overhead and network communication in both directions.

### Shrinkwrap

Shrinkwrap [Bat+18] is a construction that answers federated SQL queries hiding both access pattern and communication volume. Using the EMP-toolkit [WMK16] and the code Shrinkwrap authors shared with us, we implemented a prototype that only answers range queries. This part of Shrinkwrap amounts to making a scan over the input marking the records satisfying the range, sorting the input, and then revealing the result set plus **DP** noise to the client. For the latter part we have adapted Shrinkwrap’s Truncated Laplace Mechanism [Bat+18, Definition 4] to hierarchical method [QYL13] in order to be able to answer an unbounded number of all possible range queries. We have emulated the outsourced database setting by using two `n1-standard-32` servers in different regions (12 ms ping and 150 MB/s bandwidth) executing the algorithm in a circuit model (the faster option per Shrinkwrap experiments) and then revealing the result to the trusted client. We note that although the complexity of a Shrinkwrap query is  $\mathcal{O}(n \log n)$  due to the sorting step, its functionality is richer as it supports more relational operators, like **JOIN**, **GROUP BY** and aggregation. We also note that since MySQL, PostgreSQL and Shrinkwrap are not

---

<sup>6</sup><https://docs.microsoft.com/sql/relational-databases/security/encryption/always-encrypted-database-engine>

<sup>7</sup><https://docs.oracle.com/database/121/ASOAG/introduction-to-transparent-data-encryption.htm>

parallelized within the query, experiments using more CPUs do not yield higher performance.

### 5.5.5 Results and Observations

After running the experiments, we have made the following observations. Note that we report results based on the default setting.

- $\mathcal{E}$ psolute is efficient compared to a strawman approach, RDBMS and Shrinkwrap: it is three orders of magnitude faster than Shrinkwrap, 18 times faster than the scan and only 4–8 times slower than a conventional database. In fact, for different queries, datasets, and record sizes, our system is much faster than the linear scan, as we show next.
- $\mathcal{E}$ psolute’s client storage requirements are very practical: client size is just below 30 MB while the size of the offloaded data is over 400 times larger.
- $\mathcal{E}$ psolute scales predictably with the change in its parameters: data size affects performance logarithmically, record size — linearly, and privacy budget  $\epsilon$  — exponentially.
- $\mathcal{E}$ psolute is scalable: using  $\Pi_\gamma$  with the lightweight ORAM machines, the increase in the number of threads translates into linear performance boost.
- The optimizations proposed in Section 5.4.3 provide up to an order of magnitude performance gain.
- $\mathcal{E}$ psolute efficiently supports multiple indexed attributes. The overhead and the client storage increase slightly due to a lower privacy budget and extra local indices.

For the purposes of reproducibility we have put the log traces of all our experiments along with the instructions on how to run them on a publicly available page

[epsolute.org](http://epsolute.org). Unless stated otherwise, the scale in the figures is linear and the  $x$ -axis is categorical.

**Question-1: against RDBMS, Linear Scan and Shrinkwrap**

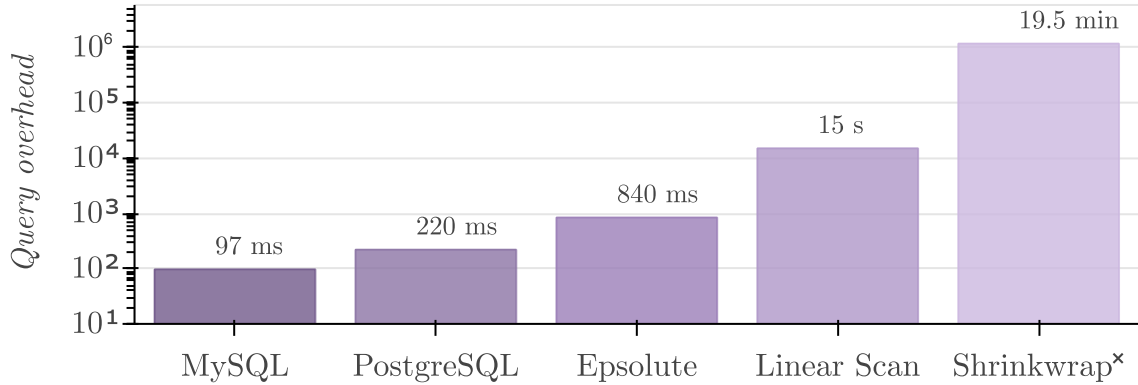


Figure 5-3: Different range-query mechanisms, logarithmic scale. Default setting:  $10^6$  4 KiB uniformly-sampled records with the range  $10^4$ .

The first experiment we have run using  $\mathcal{E}$ psolute is the default setting in which we observed the query overhead of **840 ms**. To put this number in perspective, we compare  $\mathcal{E}$ psolute to conventional relational databases, the linear scan and Shrinkwrap.

For the default setting, MySQL and PostgreSQL, configured for no privacy and maximum performance, complete in 97 ms and 220 ms respectively, which is just **8 to 4 times** faster than  $\mathcal{E}$ psolute, see Figure 5-3. Conventional RDBMS uses efficient indices (B+ trees) to locate requested records and sends them over without noise and encryption, and it does so using less hardware resources. In our experiments RDBMS performance is linearly correlated with the result and record sizes.

Linear scan experiments demonstrate the practicality of  $\mathcal{E}$ psolute compared to a trivial “download everything every time” approach, see Figure 5-4. Linear scan’s overhead is  $\mathcal{O}(n)$  regardless of the queries, while  $\mathcal{E}$ psolute’s overhead is  $\mathcal{O}(\log n)$  times the result size. According to our experiments,  $\mathcal{E}$ psolute eclipses the linear scan at 4 KiB, 64 threads and only *ten thousand records* (both mechanisms complete in



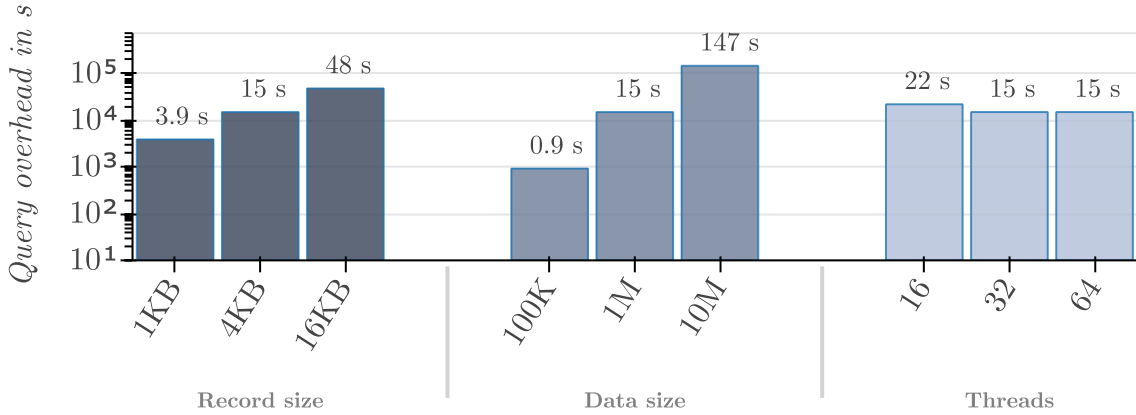


Figure 5·4: Linear scan performance, logarithmic scale. The experiments are run for the default setting of  $10^6$  records of size 4 KiB and 64 threads, with one of the three parameters varying.

about 120 ms). For a default setting (at a million records), the difference is **18 times**, see Figure 5·4.

Because Shrinkwrap sorts the input obliviously in a circuit model, it then incurs  $\mathcal{O}(n \log n)$  comparisons, each resulting in multiple circuit gates, which is much more expensive than the linear scan. Unlike linear scan, however, Shrinkwrap does not require much client memory as the client merely coordinates the query. While Shrinkwrap supports richer set of relational operators, for range queries alone  $\mathcal{E}$ psolute is **three orders of magnitude** faster.

### Question-2: storage

While  $\mathcal{E}$ psolute storage efficiency is near-optimal ( $\mathcal{O}(1), 0$ ), it is important to observe the absolute values. Index  $\mathcal{I}$  is implemented as a B+ tree with fanout 200 and occupancy 70%, and its size, therefore, is roughly  $5.7n$  bytes. Most of the **ORAM** client storage is the PathORAM stash with its size chosen in a way to bound failure probability to about  $\eta_1 = 2^{-32}$  (see [Ste+13, Theorem 1]). In Table 5.1, we present  $\mathcal{E}$ psolute storage usage for the parameters that affect it — data, record and domain sizes. We measured the sizes of the index  $\mathcal{I}$ , DP structure  $\mathcal{DS}$ , and **ORAM** client

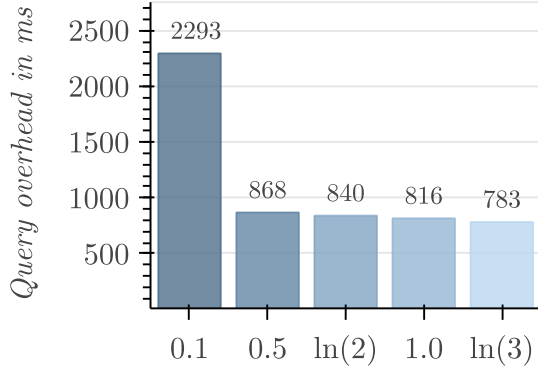
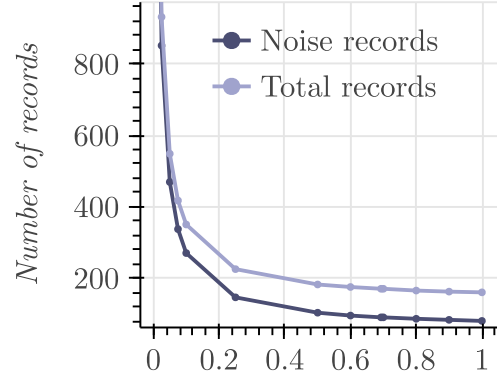
$n \backslash$ Record		1 KiB		4 KiB		16 KiB	
$10^5$		400 KiB	400 B	400 KiB	102 KiB	400 KiB	1.6 MB
		<b>396 MB</b>	4.6 MB	<b>1.5 GB</b>	14 MB	<b>6.2 GB</b>	51 MB
$10^6$		3.9 MB	400 B	3.9 MB	102 KiB	3.9 MB	1.6 MB
		<b>3.2 GB</b>	15 MB	<b>12 GB</b>	25 MB	<b>48 GB</b>	62 MB
$10^7$		40 MB	400 B	40 MB	102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

Table 5.1:  $\mathcal{E}$ solute storage usage for varying data, record and domain sizes. The values are as follows. 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* indicates that the value is estimated.

and server states. Our observations are: (i) index size expectedly grows only with the data size, (ii)  $\mathcal{DS}$  is negligibly small in practice, (iii) small  $\mathcal{I}$  and  $\mathcal{DS}$  sizes imply the efficiency of supporting multiple indexed attributes, (iv)  $\mathcal{S}$  to  $\mathcal{U}$  storage size ratio varies from **85** in the smallest setting to more than **2000** in the largest, and (v) one can trade client storage for **ORAM** failure probability. We conclude that the storage requirements of  $\mathcal{E}$ solute are practical.

### Question-3: varying parameters

To measure and understand the impact of configuration parameters on the performance of our solution we have varied  $\epsilon$ , record size, data size  $n$ , domain size  $N$ , selectivities, as well as data and query distributions. The relation that is persistent throughout the experiments is that for given data and record sizes, the performance (the time to completely execute a query) is strictly proportional to the total number of records, fake and real, that are being accessed per query. Each record access goes

Figure 5-5: Privacy budget  $\epsilon$ Figure 5-6: Effect of  $\epsilon$ 

through the **ORAM** protocol, which, in turn, downloads, re-encrypts and uploads  $\mathcal{O}(\log n)$  blocks. These accesses contribute the most to the overhead and all other stages (e.g., traversing index or aggregate tree) are negligible.

**Privacy budget  $\epsilon$  and its effect** We have run the default setting for  $\epsilon = \{0.1, 0.5, \ln 2, 1.0, \ln 3\}$ .  $\epsilon$  strictly contributes to the amount of noise, which grows exponentially as  $\epsilon$  decreases, see Figure 5-5, observe sharp drop. As visualized on Figure 5-6, at high  $\epsilon$  values the noise contributes a fraction of total overhead, while at low values the noise dominates the overhead entirely.

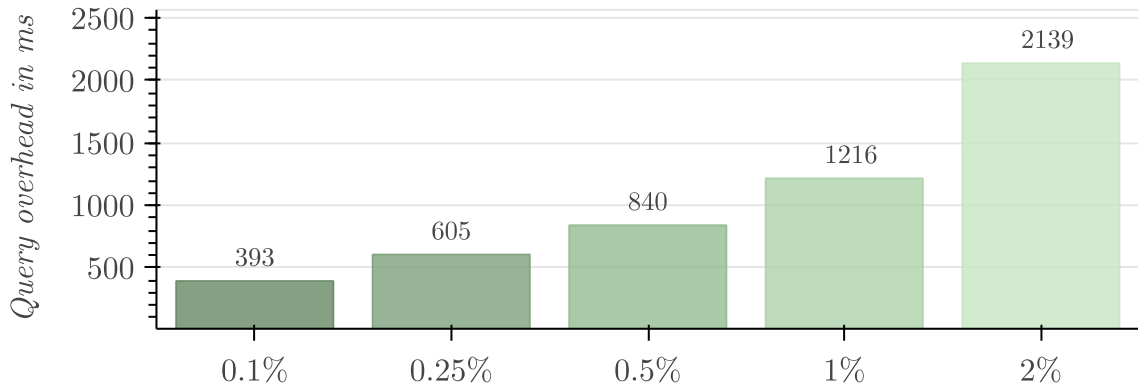


Figure 5-7: Selectivity

**Selectivity** We have ranged the selectivity from 0.1% to 2% of the total number of records, see Figure 5.7. Overhead expectedly grows with the result size. For smaller queries, and thus for lower overhead, the relation is positive, but not strictly proportional. This phenomena, observed for the experiments with low resulting per-query time, is explained by the variance among parallel threads. During each query the work is parallelized over  $m$  ORAMs and the query is completed when the *last* thread finishes. The problem, in distributed systems known as “the curse of the last reducer” [SV11], is when one thread takes disproportionately long to finish. In our case, we run 64 threads in default setting, and the delay is usually caused by a variety of factors — blocking I/O, network delay or something else running on a shared virtual CPU. This effect is noticeable when a single thread does relatively little work and small disruptions actually matter; the effect is negligible for large queries.

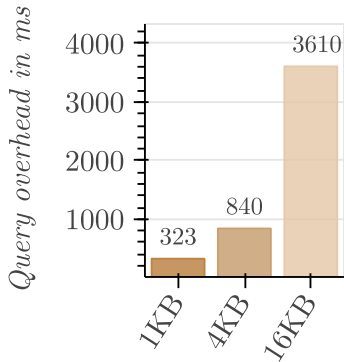


Figure 5-8: Record size

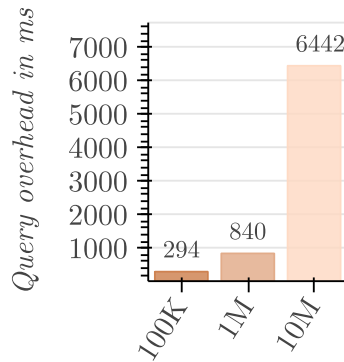


Figure 5-9: Data size

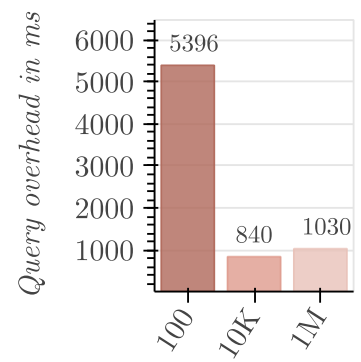


Figure 5-10: Domain size

**Record, data and domain sizes** We have tried 1 KiB, 4 KiB and 16 KiB records, see Figure 5.8. Trivially, the elapsed time is directly proportional to the record size.

We set  $n$  to  $10^5$ ,  $10^6$  and  $10^7$ , see Figure 5.9. The observed correlation of overhead against the data size is positive but non-linear, 10 times increment in  $n$  results in less than 10 times increase in time. This is explained by the ORAM overhead — when  $n$  changes, the ORAM storage gets bigger and its overhead is logarithmic.

For synthetic datasets we have set  $N$  to 100,  $10^4$  and  $10^6$ , see Figure 5·10. The results for domain size correlation are more interesting: low and high values deliver worse performance than the middle value. Small domain for a large data set means that a query often results in a high number of real records, which implies significant latency regardless of noise parameters. A sparse dataset, on the other hand, means that for a given selectivity wider domain is covered per query, resulting in more nodes in the aggregate tree contributing to the total noise value.

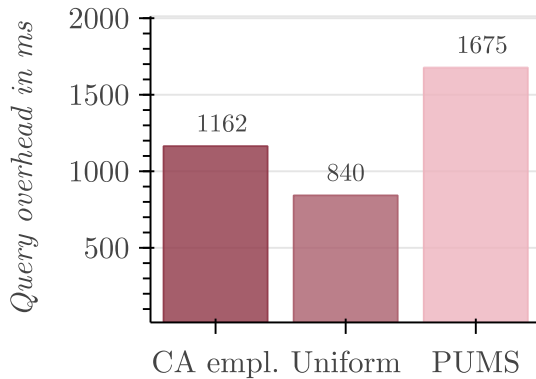


Figure 5·11: Data distribution

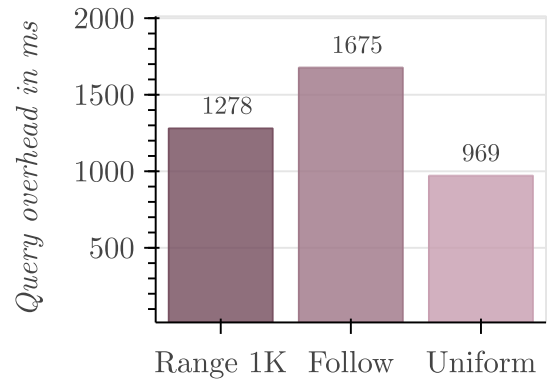
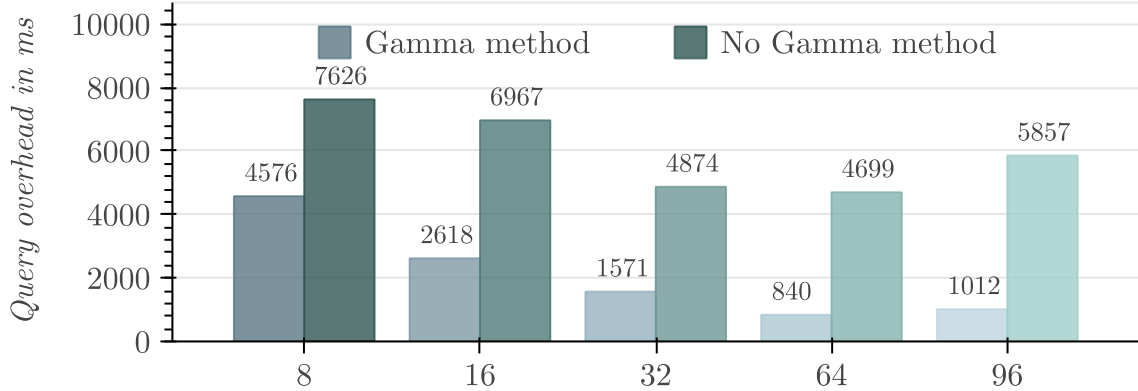


Figure 5·12: Query distribution

**Data and query distributions** Our solution performs best on the uniform data and uniform ranges, see Figures 5·11 and 5·12. Once a skew of any kind is introduced, there appear sparse and dense regions that contribute more overhead than uniform regions. Sparse regions span over wider range for a given selectivity, which results in more noise. Dense regions are likely to include more records for a given range size, which again results in more fetched records. Both real datasets are heavily skewed towards smaller values as few people have ultra-high salaries.

Figure 5.13: Scalability measurements for  $\Pi_\gamma$  and  $\Pi_{\text{no-}\gamma}$ **Question-4: scalability**

Horizontal scaling is a necessity for a practical system, this is the motivation for the parallelization in the first place. Ideally, performance should improve proportionally to the parallelization factor, number of **ORAMs** in our case,  $m$ .

For scalability experiments we run the default setting for both  $\Pi_{\text{no-}\gamma}$  and  $\Pi_\gamma$  (*no- $\gamma$ -method* and  *$\gamma$ -method* respectively) varying the number of **ORAMs**  $m$ , from 8 to 96 (maximum virtual **CPUs** on a **GCP VM**). The results are visualized on Figure 5.13. We report two positive observations: (i) the  $\gamma$ -method provides substantially better performance and storage efficiency, and (ii) when using this method the system scales linearly with the number of **ORAMs**. ( $m = 96$  is a special case because some **ORAMs** had to share a single **KVS**.)

**Question-5: optimizations benefits**

Improvement (section)	Enabled	Disabled	<b>Boost</b>
<b>ORAM batching</b> (5.4.3)	840 ms	6 978 ms	<b>8.3x</b>
Lightweight <b>ORAM machines</b> (5.4.3)	840 ms	4 484 ms	<b>5.3x</b>
Both improvements	840 ms	8 417 ms	<b>10.0x</b>

Table 5.2: Improvements over parallel  $\mathcal{E}$ psolute

Table 5.2 demonstrates the boosts our improvements provide; when combined, the speedup is up to an order of magnitude.

ORAM request batching (Section 5.4.3) makes the biggest difference. We have run the default setting with and without the batching. The overhead is substantially smaller because far fewer I/O requests are being made, which implies benefits across the full stack: download, re-encryption and upload.

Using lightweight ORAM machines (Section 5.4.3) makes a difference when scaling. In the default setting, 64 parallel threads quickly saturate the memory access and network channel, while spreading computation among nodes removes the bottleneck.

#### Question-6: multiple attributes

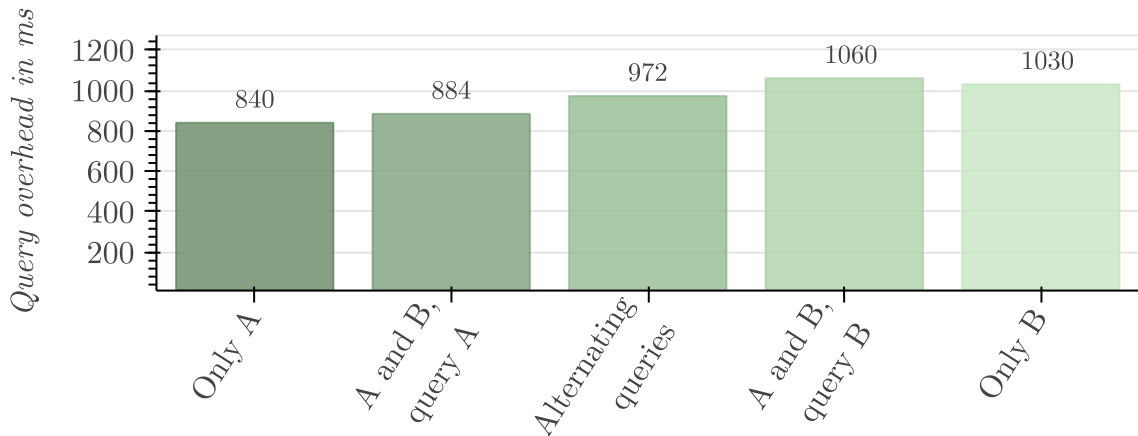


Figure 5.14: Query overhead when using multiple attributes. *Only A* and *Only B* index one attribute. *A and B* indexes both attributes and then queries one of them. *Alternating* indexes both attributes and runs half of the queries against *A* and another half against *B*.

$\mathcal{E}$ psolute supports multiple indexed attributes. In Section 5.3.4 we described that the performance implications amount to having an index  $\mathcal{I}$  and a DP structure  $\mathcal{DS}$  per attribute and sharing the privacy budget  $\epsilon$  among all attributes. As shown in Table 5.1,  $\mathcal{I}$  and  $\mathcal{DS}$  are the smallest components of the client storage. To observe

the query performance impact, we have used the default dataset with domains  $10^4$  and  $10^6$  as indexed attributes  $A$  and  $B$  respectively. We ran queries against only  $A$ , only  $B$  and against both attributes in alternating fashion. Each of the attributes used  $\epsilon = \frac{\ln 2}{2}$  to match the default privacy budget of  $\ln(2)$ .

Figure 5.14 demonstrates the query overhead of supporting multiple attributes. The principal observation is that the overhead increases only slightly due to a lower privacy budget. The client storage went up by just 9 MB, and still constitutes only 3.3% of the server storage, which is not affected by the number of indexed attributes.

## 5.6 Conclusion and Future Work

In this work, we present a system called  $\mathcal{E}$ psolute that can be used to store and retrieve encrypted records in the cloud while providing strong and provable security guarantees, and that exhibits excellent query performance for range and point queries. We use an optimized **Oblivious Random Access Machine** protocol that has been parallelized together with very efficient **DP-sanitizers** that hide both the access patterns and the exact communication volume sizes and can withstand advanced attacks that have been recently developed. We provide a prototype of the system and present an extensive evaluation over very large and diverse datasets and workloads that show excellent performance for the given security guarantees.

In our future work, we plan to investigate methods to extend our approaches to use a **Trusted Execution Environment (TEE)**, like **SGX**, in order to improve the performance even further. We will also explore a multi-user setting without the need for a shared stateful client, and enabling dynamic workloads with insertions and updates. We will also consider how adaptive and non-adaptive security models would change in the case of dynamic environments. Lastly, we plan to explore other relational operations like **JOIN** and **GROUP BY**.



## Chapter 6

# $k$ NN queries in the snapshot model

In this chapter we describe and analyze *k-anon*, a system that executes secure  $k$ NN queries in the snapshot adversary model. We study the effect of protecting the records with a type of property-preserving encryption on quality of search and efficiency of certain attacks. Specifically, we examine theoretically and practically how accuracy of both  $k$ NN search and ML-based inversion attack degrade with added security.

### 6.1 Introduction

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 point according to the distance metric. A  $k$ -nearest-neighbor ( $k$ NN) search is a subtype of a general nearest-neighbor problem where  $k$  closest objects are requested. Applications that use  $k$ NN 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).

In this chapter we propose a method and an analysis of running secure  $k$ NN queries in an outsourced database model. We model our application as a generic

document similarity search, where the server stores the embeddings of the documents and returns the  $k$  closest records to the query embedding. We propose to apply a type of property-preserving encryption over the embeddings on the server while retaining its ability to do nearest-neighbor search. Finally, we simulate an attack against the records — an ML-based inversion attack that aims to recover the set of words of a document from its embedding. Our goal is to observe and study the correlation of the security parameter (an approximation term) with search accuracy and attack efficiency.

To summarize, our contributions in this work are as follows:

- We construct *k-anon* — a secure similarity search system in snapshot adversary setting. We analyze the correlation of system’s security with search accuracy and attack efficiency, and conclude on practicality of the *k-anon* system.
- We implement a Distance Comparison Preserving Encryption (DCPE) scheme from [FGHO21] and adapt it to encrypt text embeddings. We analyze the practical aspects of DCPE scheme’s security (e.g., the effects of floating point representation) and benchmark the construction.
- We conduct a set of experiments to study the effect of the security parameter on search accuracy. We use a fine-tuned BERT model to produce embeddings for the TREC 2020 collection. Given the TREC validation set (i.e., “correct answers”) we run the search for varying security levels and report a number of ranking quality measures.
- We adapt a recent ML-based inversion attack by Song and Raghunathan [SR20] against embeddings in our setting. The attack works by training an LSTM model on pairs of sentences and their embeddings. We run this attack for varying security levels, and training on both plaintext and encrypted records.

## 6.2 Distance Comparison Preserving Encryption

A promising approach in secure  $k$ NN evaluation is using a property-preserving encryption scheme to allow the existing search algorithms to work with minimal alterations. ASPE scheme by Wong et al. [WCKM09] is a step in this direction, but their scheme has been shown insecure under a type of chosen plaintext attack in [Y LX13]. Using a common OPE scheme over vector values to encrypt the objects for the purpose of running  $k$ NN queries on them has been explored in [WHL20], but this approach incurs high overhead linear in dimensionality. See more detailed related work analysis in Section 3.3. We, therefore, need a different method — a scheme that operates over high-dimensional vectors and preserves a property that is required to answer the nearest-neighbor queries.

A classical nearest-neighbor search [WL83; CD21] simply orders the objects according to their distances from the target. It is important to note that knowing the exact distance is not required, merely the knowledge of *distance comparison* suffices (i.e.,  $x$  is closer to  $y$  than  $z$  is). An encryption scheme that preserves the distance comparison would satisfy the  $k$ NN search correctness, but not necessarily security or even practicality. First, a fully deterministic Distance Comparison Preserving Encryption (DCPE) would reveal at least the frequency of data points (i.e., how many times a point appears in the dataset). Second, even in the plaintext world the use of approximate nearest-neighbor search [ML14; Ary+98] may be preferred due to the curse of dimensionality [BGRS99; IM98] (the exact distance is less important in higher dimensions).

### 6.2.1 DCPE construction

A candidate *approximate* DCPE scheme that we adapt to our solution has been recently proposed by Fuchsbaauer et al. [FGHO21]. The scheme provides the following

guarantee on its ciphertexts

$$\begin{aligned} \forall x, y, z \in \mathbb{X} : \text{DIST}(x, y) < \text{DIST}(x, z) - \beta \\ \implies \text{DIST}(f(x), f(y)) < \text{DIST}(f(x), f(z)) \end{aligned}$$

where  $\mathbb{X} \subseteq \mathbb{R}^d$  is the set of  $d$ -dimensional vectors of real numbers,  $\text{DIST}$  is the  $L_2$  distance over elements in  $\mathbb{X}$ , and  $\beta$  is the approximation term. Parameter  $\beta$  partially defines the security of the encrypted set — the larger  $\beta$ , 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 the 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  $\beta$ , Fuchsbauer et al. [FGHO21] prove that  $\beta \approx \sqrt{\max N}$  would hide about half of the input bits, for  $\max N$  being the maximum vector length in the dataset.

---

**Algorithm 3** Distance Comparison Preserving Encryption scheme, adapted from [FGHO21, Algorithm 2].

---

KEYGEN( $1^\lambda, \mathbb{S}$ )	ENC( $(s, \mathbf{k}), \vec{m}$ )	DEC( $(s, \mathbf{k}), (\vec{c}, n)$ )
1 : $s \leftarrow \mathbb{S}$	1 : $n \leftarrow \{0, 1\}^\lambda$	1 : $\text{coins}_n \parallel \text{coins}_u \leftarrow \text{PRF}(k, n)$
2 : $\mathbf{k} \leftarrow \{0, 1\}^\lambda$	2 : $\text{coins}_n \parallel \text{coins}_u \leftarrow \text{PRF}(k, n)$	2 : $\vec{n} \leftarrow \text{NORMAL}(0, I_d; \text{coins}_n)$
3 : <b>return</b> $(s, \mathbf{k})$	3 : $\vec{n} \leftarrow \text{NORMAL}(0, I_d; \text{coins}_n)$	3 : $u \leftarrow \text{UNIFORM}(0, 1; \text{coins}_u)$
	4 : $u \leftarrow \text{UNIFORM}(0, 1; \text{coins}_u)$	4 : $x \leftarrow \frac{s\beta}{4} \cdot \sqrt[d]{u}$
	5 : $x \leftarrow \frac{s\beta}{4} \cdot \sqrt[d]{u}$	5 : $\vec{\delta} \leftarrow \frac{\vec{n}}{\ \vec{n}\ } \cdot x$
	6 : $\vec{\delta} \leftarrow \frac{\vec{n}}{\ \vec{n}\ } \cdot x$	6 : $\vec{m} \leftarrow \frac{\vec{c} - \vec{\delta}}{s}$
	7 : $\vec{c} \leftarrow s \cdot \vec{m} + \vec{\delta}$	7 : <b>return</b> $\vec{m}$
	8 : <b>return</b> $\vec{c}$	

---

Fuchsbauer et al. [FGHO21] offer an instantiation of the  $\beta$ -DCPE scheme (though not an implementation) that we have adapted to our needs and show on Algorithm 3.

The KEYGEN procedure generates a key  $\mathbf{k}$  and an amplification factor  $s$ . The key participates in generating the random coins needed to produce deterministic execution, and the amplification factor controls the magnitude of projection of a plaintext object into the ciphertext.

The ENC procedure “encrypts” an object by moving it in space in a way that makes it hard to recover its original position while its distance-comparison relative to other encrypted points is preserved. The algorithm first constructs a hypersphere of radius  $\beta$ , the approximation term, around the input point. The routine then samples a new point uniformly inside that hypersphere. Finally, that new point is projected into the ciphertext point according to the amplification factor  $s$ . Note that for each encryption the scheme generates a fresh nonce  $n$  and uses it along with the key  $\mathbf{k}$  to generate the coins for the samplers. That is, the point in the  $\beta$ -hypersphere is deterministically set from the nonce (“**number used only once**”, unique per point) and the key (one for all points), and the final ciphertext is projected the same way for all points. The DEC procedure makes the same steps in reverse, correctly setting the point in the hypersphere using the nonce and the key. See Figure 6.1 for a visual example of DCPE encryption.

### 6.2.2 DCPE security

The security of the scheme thus depends on (i) the maximum amount of amplification, (ii) the radius of the hypersphere  $\beta$ , and (iii) the entropy of the samplers. Fuchsbauer et al. [FGHO21] show that the amplification,  $s$  parameter, affects one-wayness bounds [FGHO21, Section 7.2]. The approximation term  $\beta$  affects bit-security with  $\beta \approx \sqrt{\max N}$  protecting about half of the bits. Finally, the key  $\mathbf{k}$  and nonce  $n$  sizes, the security parameter  $1^\lambda$ , and the samplers used to generate normal multivariate and uniform samples affect the specific amount of entropy used to generate a point in the hypersphere.

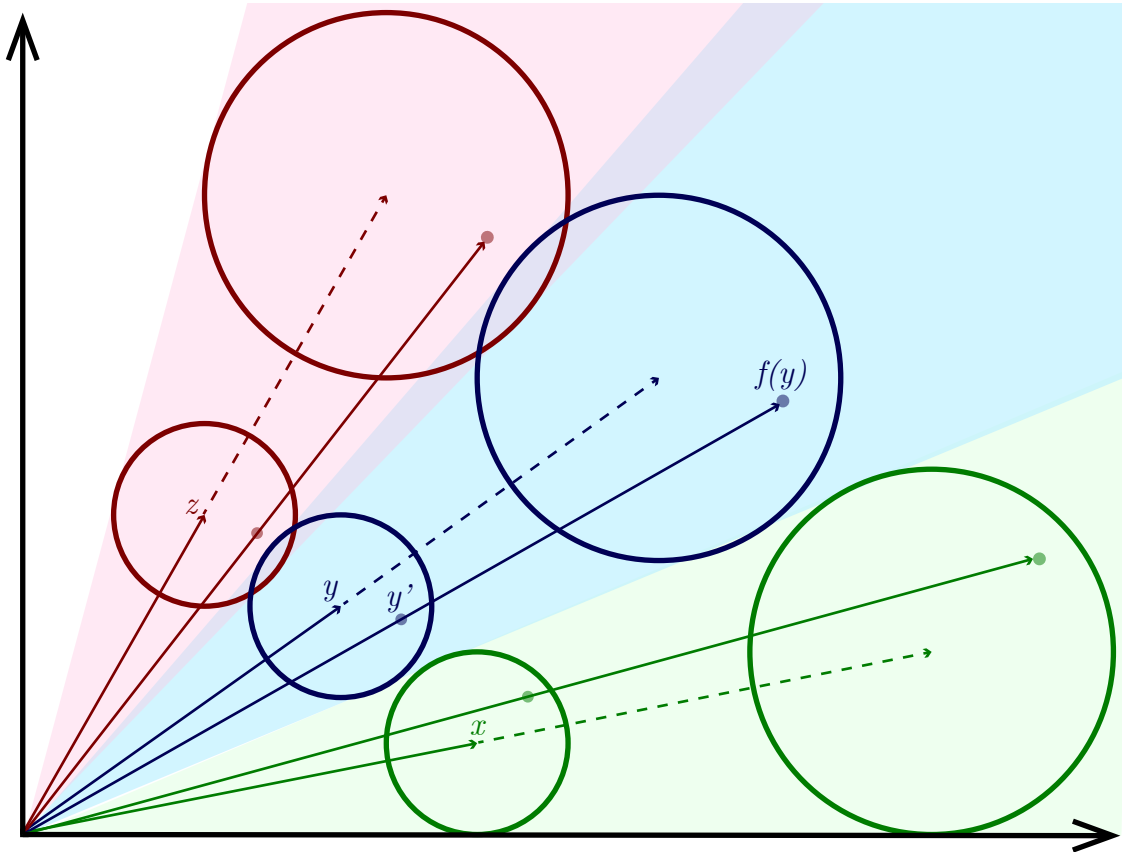


Figure 6-1: Schematic description of encryption process of DCPE, drawn to scale. In this diagram, there are two dimensions ( $d = 2$ ),  $\beta$  (the radius of a circle) is 2 units, and  $s$  (the projection magnitude, the length from the origin to the larger circle over the length to the smaller one) is 2. The encrypted point is uniformly sampled inside a  $\beta$ -sphere, then projected  $s$  times further from the origin. If two points are too close, their circles intersect, and their encryptions can be sampled in a way that breaks distance comparison. Intuitively, larger  $\beta$  implies greater ciphertext space for a point and greater security.

As the construction operates on real numbers, an open question remains on how to avoid negative side-effects of floating point numbers bit representation. Unlike integers, floating point numbers are represented in memory in a way that their precision is different depending on their value, see the IEEE 754-2019 standard [Dav19]. Simply put, the closer the value is to zero, the smaller the difference between two consecutive representable values is. For example, while the representable 32-bit IEEE 754 floating

point values range from about  $1.18 \cdot 10^{-38}$  to  $3.4 \cdot 10^{38}$ , there are only  $2^{32} \approx 4 \cdot 10^9$ , 4 billion representable discrete numbers. This, along with the rounding errors, puts some limits on how large  $s$  and  $\beta$  can be.

### 6.2.3 DCPE implementation and benchmarks

We offer the first implementation of [FGHO21]  $\beta$ -DCPE for 32-bit and 64-bit IEEE 754 numbers in C++.<sup>1</sup> The code is documented, tested and benchmarked, see Table 6.1. Observe that the difference in performance between encryption and decryption is predictably minimal, and the overhead of encryption grows slower-than-linearly with dimensionality.

Operation	Input size	Dimensions $d$	Wall-clock time
KEYGEN	N/A	N/A	1.81 ms
ENC	32-bit (float)	1	4.12 ms
		100	12.2 ms
		768	62.0 ms
	64-bit (double)	1	3.96 ms
		100	11.4 ms
		768	59.3 ms
DEC	32-bit (float)	1	3.94 ms
		100	11.6 ms
		768	62.1 ms
	64-bit (double)	1	3.96 ms
		100	11.3 ms
		768	59.6 ms

Table 6.1: DCPE implementation benchmarks

<sup>1</sup><https://github.com/private-knn/dcpe>

### 6.3 $k$ NN search accuracy

The first part of *k-anon* is the search accuracy. In this set of experiments, we embed the documents and apply  $\beta$ -DCPE to the embeddings. We use existing efficient  $k$ NN search algorithms and report ranking quality metrics for different  $\beta$ .

#### 6.3.1 Secure $k$ NN protocol

With the  $\beta$ -DCPE as a component, we can model the *k-anon* protocols similar to ORE with B+ tree ones. In the setup protocol  $\Pi_{\text{setup}}$ ,  $\mathcal{U}$  simply encrypts the entire input, one vector at a time, and sends the encrypted data over to  $\mathcal{S}$ . In the query protocol  $\Pi_{\text{query}}$ ,  $\mathcal{U}$  encrypts the query with DCPE, sends the ciphertext to  $\mathcal{S}$ , while  $\mathcal{S}$  runs a standard  $k$ NN search against the ciphertext.  $k$  encrypted vectors are then returned to  $\mathcal{U}$ , which decrypts them as the last step. These protocols are executed for a single set of secrets and DCPE parameters, including  $\beta$ .

For the choice of the dataset, we use the established information retrieval TREC 2020 test collections (MS MARCO passage retrieval collection [Baj+16]). A TREC collection consists of a set of documents, a set of topics (questions) and a corresponding set of relevance judgments (correct answers). The benefit of using a TREC dataset is being able to evaluate relevant metrics over the produced results, for example, MRR [Cra09] and nDCG [JK02]. We can then track how these metrics, along with the simpler edit distance and set difference over the result, degrade with higher security.

As the embedding mechanism, we use a custom fine-tuned BERT model. Bidirectional Encoder Representation from Transformer (BERT) is a transformer-based ML technique for Natural Language Processing. An original BERT model, published by Devlin et al. [DCLT19], has been trained on a BookCorpus [Zhu+15] (800 million words) and English Wikipedia (2.5 billion words) using 24 encoders with 16 bidi-



rectional self-attention heads (for the larger of two versions, `BERTLARGE`). BERT’s main novelty is its bidirectional nature — it processes all words in relation to each other and not one-by-one. The technology is now prevalent [RKR21], and Google uses BERT in almost every English query<sup>2</sup>.

It is common, however, to use the original BERT model as a base and do training on top. We trained a BERT-based dense retrieval model that uses BERT for representing both queries and documents and inner product for computing their similarity [Hof+21]. The parameters between both BERT models (for queries and documents) are shared. We used cross entropy loss function for training and used the standard MS MARCO [Baj+16] training set. The produced vectors by the BERT model are used in our experiments for approximate nearest neighbor indexing of documents and retrieval for queries. We used the test queries produced by the Deep Learning Track of the Text Retrieval Conference (TREC) in 2020 in our experiments.

### 6.3.2 Experimental evaluation

The actual experiment is conducted as follows. First, we produce a set of embeddings for 8.8 million documents and 200 queries from TREC 2020 dataset. We observed that the maximum length of an embedding vector is about  $\sqrt{\max N} \approx 11$  units. Second, we encrypt the record and query embeddings using DCPE and range  $\beta$  from 0 (meaning exact distance-comparison) to 50, with  $\beta = \sqrt{11} \approx 3.3$  hiding about half of the bits of input embeddings. Third, we run the nearest neighbor search on these pairs of data and queries sets using FAISS [JDJ21], a GPU-enabled library for efficient similarity search and clustering of dense vectors. Finally, we report a range of ranking quality metrics and generic  $k$ NN result metrics.

---

<sup>2</sup><https://searchengineand.com/google-bert-used-on-almost-every-english-query-342193>

## Ranking quality metrics

We report recall, Mean Reciprocal Rank (MRR) [Cra09] and Normalized Discounted Cumulative Gain (nDCG) [JK02] to assess the ranking quality with respect to TREC relevance judgments.

*Recall* is the fraction of relevant documents that the query retrieved over all relevant documents.

Mean Reciprocal Rank (MRR) is the average of reciprocal ranks of a query response, which is a multiplicative inverse of the rank of the first correct answer. MRR is defined as

$$\text{MRR} = \frac{1}{|\mathbb{Q}|} \sum_{i=1}^{|\mathbb{Q}|} \frac{1}{\text{rank}_i},$$

where  $\mathbb{Q}$  is the sample of queries and  $\text{rank}_i$  is the rank position of the first relevant document for the  $i_{\text{th}}$  query.

Normalized Discounted Cumulative Gain (nDCG) is another measure of ranking quality most used in web search engine algorithms. Its main goal is to produce a metric that would promote the following two assumptions. First, a document’s relevance implies its usefulness, and second, highly relevant documents are more useful if they have higher rank (appear earlier in the result list). nDCG is a normalized version of discounted cumulative gain, and is defined as the actual over ideal gain up to a position  $p$

$$\text{nDCG}_p = \frac{\text{DCG}_p}{\text{IDCG}_p} = \frac{\sum_{i=1}^p \frac{\text{relevance}_i}{\log_2(i+1)}}{\sum_{i=1}^{\text{REL}(p)} \frac{\text{relevance}_i}{\log_2(i+1)}},$$

where  $\text{relevance}_i$  is the graded relevance of the result at position  $i$  and  $\text{REL}(p)$  is the number of relevant documents in the corpus up to position  $p$ . Note that we use the classic definition by Järvelin and Kekäläinen [JK02], and not the one by Burges et al. [Bur+05] that puts even greater emphasis on relevance.

We measure all three metrics at certain cutoffs, meaning that if the number of returned documents is smaller than the cutoff, the missing documents are assumed to be irrelevant. The cutoff for recall is 1 000 and for **MRR** and **nDCG** is 10, as is common in the information retrieval literature.

To keep track of the isolated effect of the approximation factor on the **kNN** results, we also report the set difference and Damerau-Levenshtein distance [Lev66; Dam64] of actual and expected **kNN** results. Result distance is measured as the minimal number of insert, delete and swap operations to get one set from the other. The inclusion of swap operation makes metric penalize less the case when the search returned the correct set with only a few documents transposed. This property is especially useful for us since the approximate distance comparison preservation results exactly in this kind of small error in the output. Result difference is simply a set difference of two outputs. This metric does not penalize the wrong order of documents as long as all  $k$  relevant documents are present. For ease of exposition, we report these two metrics as the fraction of total number of returned documents. That is, for 1 000 returned documents the distance of 853 would be reported as 85.3%.

### 6.3.3 Results for varying $\beta$

We run two sets of experiments to see how these metrics change with varying the security parameter  $\beta$ . For the first set of  $\beta$  values, we ranged the parameter from 0 to 50 with increments of 1.0, see Figure 6.2. Higher values of  $\beta$  predictably degraded the search accuracy, but we wanted to see how quickly and after which values of  $\beta$  the accuracy starts to fall noticeably.

First, we notice that the result difference grows close to linearly with  $\beta$ , which means that each new security level knocks out a few correct responses from the set proportionally. Second, we see that the result distance jumps immediately to almost 100%, which means that even a tiny approximation term significantly perturbs the

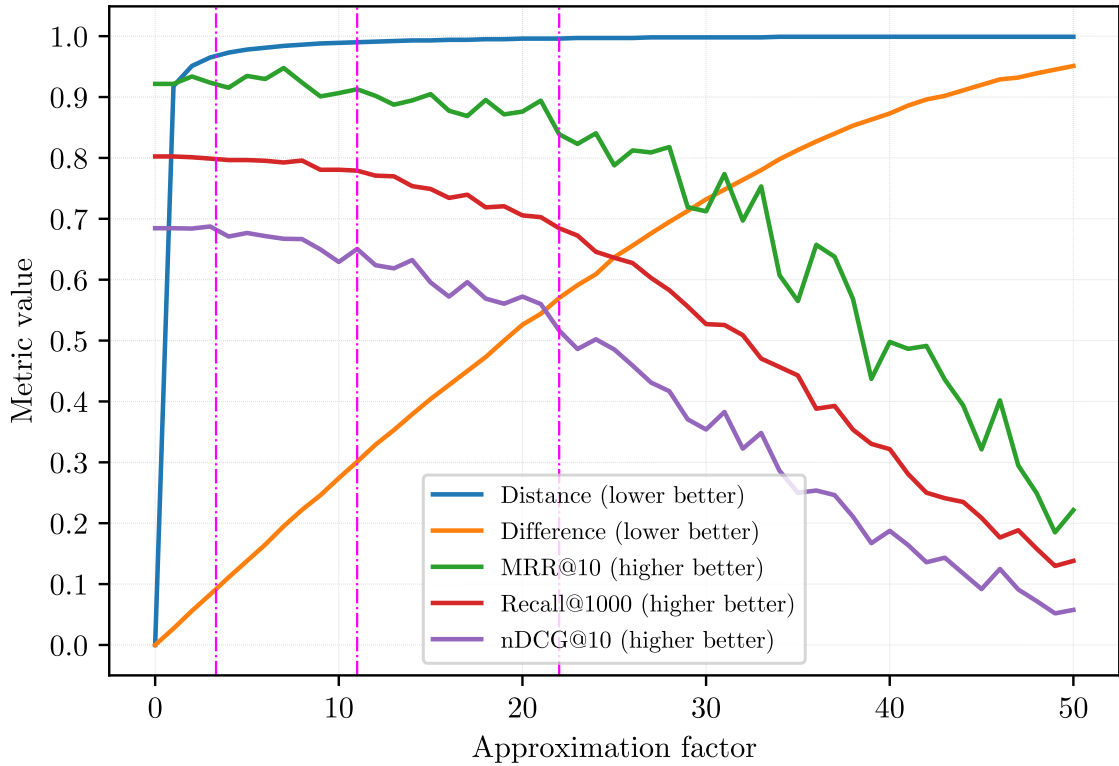


Figure 6-2: Search accuracy for  $\beta \in \{0.0, \dots, 50.0\}$

order of the responses, but given low difference between the answer sets, not the content of the result. Finally, we observe that at about  $\beta = 9$ , the TREC metrics start to fall sharply and throughout the entire range of  $\beta$  they fall in accord.

For the maximum vector length of  $\sqrt{\max N} \approx 11$ , the  $\beta$  value that hides half of the input bits is  $\beta = \sqrt{11} \approx 3.3$ . To see the metrics behavior at around this point, we ran the experiments again, now ranging  $\beta$  in finer manner, from 0.0 to 5.0 with increments of 0.1, see Figure 6-3. We confirm that for  $\beta = \sqrt{\max N} \approx 3.3$ , the values of the ranking quality metrics are sufficiently close to the plaintext values. We therefore conclude that the bit-security DCPE offers comes with a low search accuracy penalty.

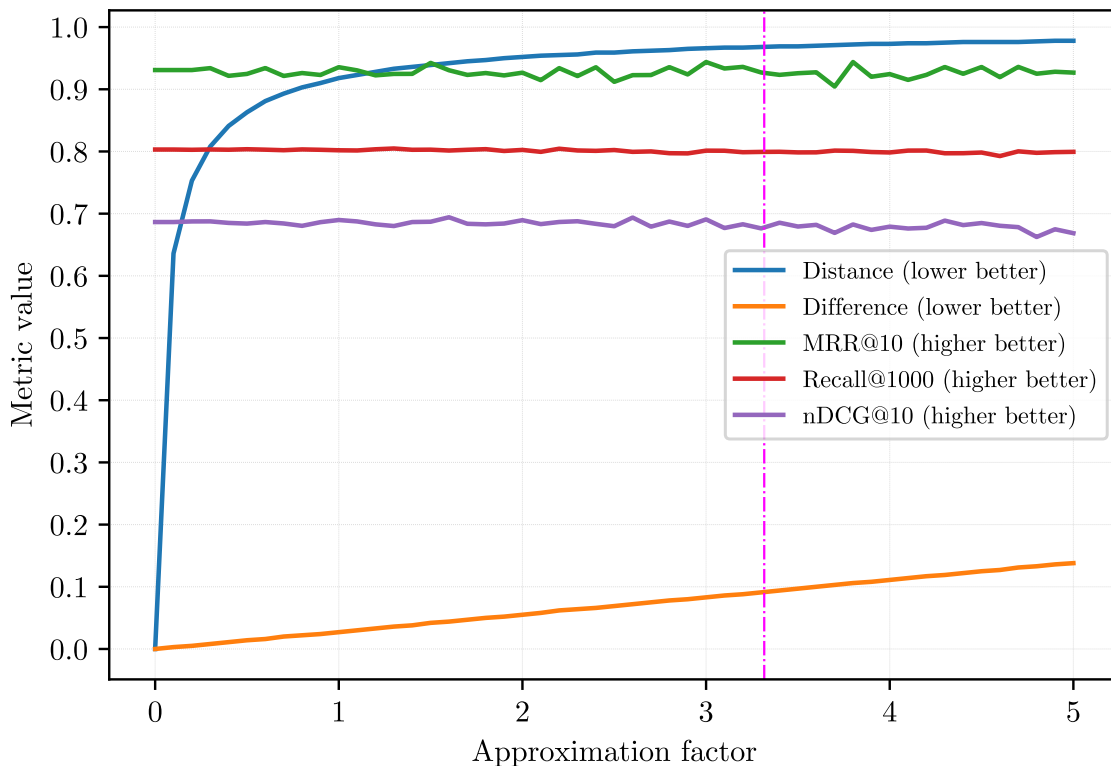


Figure 6-3: Search accuracy for  $\beta \in \{0.0, \dots, 5.0\}$ . Highlighted is  $\beta = \sqrt{\max N}$  for  $\max N \approx 11$  being the longest vector in the dataset.

## 6.4 Security against attacks

The second part of *k-anon* is the protection against attacks. In this set of experiments, we adapt a recent ML-based attack by Song and Raghunathan [SR20]. Song and Raghunathan [SR20] offer three attacks against text embeddings: (i) *the inversion attack*, which recovers a set of words from a document embedding, (ii) *the attribute inference attack* that recovers some property of the document by its embedding, such as the gender or age of its author, and (iii) *the membership inference attack*, which reveals whether a given document was or was not in the training set for the embedding model. The inversion attack can be run in two modes: white-box and black-box. The white-box attack assumes the access to an embedding model and

therefore can directly use its architecture and parameters to invert the inputs. The black-box attack relies only on being able to use the model to produce an embedding from a document, similar to how a generic embedding [API](#) would work. We use this latter black-box inversion attack in our experiments since it most closely matches the adversary capabilities in our outsourced database setting.

#### 6.4.1 Black-box model inversion attack [\[SR20\]](#)

The black-box inversion attack assumes no knowledge of the embedding model; it can only use it to produce the embeddings. In this section we follow the original notation by Song and Raghunathan [\[SR20\]](#).

The attack works by training another model  $\Upsilon$  to recognize the correlation between the set of words  $\mathcal{W}(x)$  of a document  $x$  and its embedding  $\Phi(x)$ . The attacker uses some auxiliary dataset  $\mathcal{D}_{\text{aux}}$  (same domain works predictably better than cross domain), and produces a collection of  $(\Phi(x), \mathcal{W}(x))$  for all  $x \in \mathcal{D}_{\text{aux}}$ . The adversary then trains the attack model  $\Upsilon$  to maximize  $\log \Pr_{\Upsilon}(\mathcal{W}(x) | \Phi(x))$  over dataset  $(\Phi(x), \mathcal{W}(x))$ . Finally, the adversary simply queries the model with the embedding  $\Upsilon(\Phi(x))$  and recovers the original words  $\mathcal{W}(x)$ .

Song and Raghunathan [\[SR20\]](#) offer two models for the inversion attack. The [Multi-Label Classification \(MLC\)](#) model assigns a binary label of whether a word is in the set for each word in the dictionary. The objective function is

$$\mathcal{L}_{\text{MLC}} = - \sum_{w \in \mathcal{V}} [y_w \log(\hat{y}_w) + (1 - y_w) \log(1 - \hat{y}_w)]$$

where

- $\mathcal{V}$  is a set of possible words, a dictionary,
- $\hat{y}_w = \Pr_{\Upsilon}(y_w | \Phi(x))$  is the predicted probability of word  $w$  given  $\Upsilon$  conditioned on  $\Phi(x)$ , and

- $y_w = 1$  if word  $w$  is in  $x$  and 0 otherwise.

A disadvantage of MLC model is that it predicts all words independently. Song and Raghunathan [SR20] therefore offer a more sophisticated approach based on [Wel+18]. A Multi-Set Prediction (MSP) recurrent neural network predicts the next word in the set conditioned on the embedding  $\Phi(x)$  and the up-to-date predicted set of words. The objective function is

$$\mathcal{L}_{\text{MSP}} = \sum_{i=1}^{\ell} \frac{1}{|\mathcal{W}_i|} \sum_{w \in \mathcal{W}_i} -\log \Pr_{\Upsilon}(w | \mathcal{W}_{<i}, \Phi(x))$$

where  $\mathcal{W}_{<i}$  is the set of already predicted words up to a timestamp  $i$  and  $\mathcal{W}_i$  is the set of words left to predict. According to experiments by Song and Raghunathan [SR20], MSP outperforms MLC.

#### 6.4.2 Experimental evaluation

We have contacted Song and Raghunathan [SR20] and obtained a prototype code they used to run the attack.<sup>3</sup> The model  $\Upsilon$  is implemented as a one-layer LSTM with 300 hidden units. The model is trained for 30 epochs with Adam optimizer [KB14], learning rate of  $10^{-3}$  and batch size of 256. The actual implementation uses TensorFlow [Aba+16] (version 1) and is naturally optimized for GPU training.

We run two sets of experiments corresponding to two adversarial settings. First, similar to the work of Song and Raghunathan [SR20], we assume that the adversary has a black-box access to the embedding model  $\Phi$  and can use it to train  $\Upsilon$ , and we call this experiment a *public model*. This setting corresponds to a scenario when the system uses some open embedding model with little to no alterations (i.e., Google AI Platform Training<sup>4</sup>). Second, we assume a more realistic scenario where the embedding model  $\Phi$  is not available, and the adversary can only use the system

<sup>3</sup><https://github.com/google/embedding-tests>

<sup>4</sup><https://cloud.google.com/ai-platform/training/docs/algorithms/bert>

as a whole. That is, for a given document  $x$ , the adversary can only produce the encrypted embedding  $\text{ENC}(\Phi(x))$ . We call this experiment a *private model*. In both experiments the actual outsourced database is encrypted and the difference is in the dataset  $\mathcal{D}_{\text{aux}}$  on which the adversary can train the attack model  $\Upsilon$ . In the public model experiment, the adversary trains on plaintext embeddings while in the private model environment she trains on the already encrypted embeddings.

We have used two datasets for both experiments. The first dataset is a BookCorpus [Zhu+15] that Song and Raghunathan [SR20] used. With this dataset we have been able to verify that our adapted attack implementation produces similar results to original [SR20]. The second dataset is the TREC 2020, same as in Section 6.3. With this dataset we can link together the search accuracy and attack efficiency for the same levels of security.

### Attack efficiency metrics

In line with the work of Song and Raghunathan [SR20], we define the attack efficiency metrics similar to a generic ML measurers of accuracy — precision, recall and  $F_1$  score. Precision in our setting is defined as the number of words that the model predicted and that are part of an embedded sentence (i.e., true positives) over the total number of predicted words (all positives). Recall is defined as the number of correctly predicted words (true positives) over the number of all words in the sentence (true positives plus false negatives).  $F_1$  score is then the harmonic mean of these two:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

We go a step further in this direction and given the context of the model — recovering a set of words from a sentence embedding — also track the fraction of common words (a.k.a. stop-words) in the predicted set. BERT explicitly encourages



including the stop-words in the input because their relative position matters for the context and thus embedding. However, the attack only produces an unordered set of words and not their relative position. Therefore, the *model prediction quality*, the  $F_1$  score, may seem high, but it may not imply high *attack efficiency*, because a huge fraction of predicted words are common and thus contribute little to added adversary knowledge. We define the list of stop words as pronouns, verb forms of “be”, “have” and “do”, some modal verbs, compound forms (e.g., “you’ll”), negations, articles, certain prepositions, conjunctions, adverbs and some more high-frequency words.<sup>5</sup> We also include punctuation and digits in the list, as **BERT** tokenizes these along with the rest of the words.

## Baselines

We have established two baselines between which the attack performance over encrypted inputs is assumed to lie. The first baseline is the attack on the plaintext inputs (referred to as *plaintext attack*), a replica of the original attack by Song and Raghunathan [SR20]. The second baseline is the attack on the random embeddings (referred to as *random attack*), which, counterintuitively, does not produce close to zero  $F_1$  score.

In both cases, we have trained the attack model for 30 epochs (see  $F_1$  score on Figure 6.4). We note a number of observations: (i) we have replicated the efficiency result of the black-box model inversion attack from [SR20, Table 2,  $F_1$  score, same domain,  $\mathcal{L}_{MSP}$ , **BERT**], (ii) for the plaintext attack, the  $F_1$  score stops growing after about 10<sup>th</sup> epoch, (iii) the random attack produces a far-from-negligible  $F_1$  score, (iv) **TREC** dataset is much less susceptible to the attack than the originally used BookCorpus.

---

<sup>5</sup>The list is adapted from the Snowball processing language: <http://snowball.tartarus.org/algorithms/english/stop.txt>.

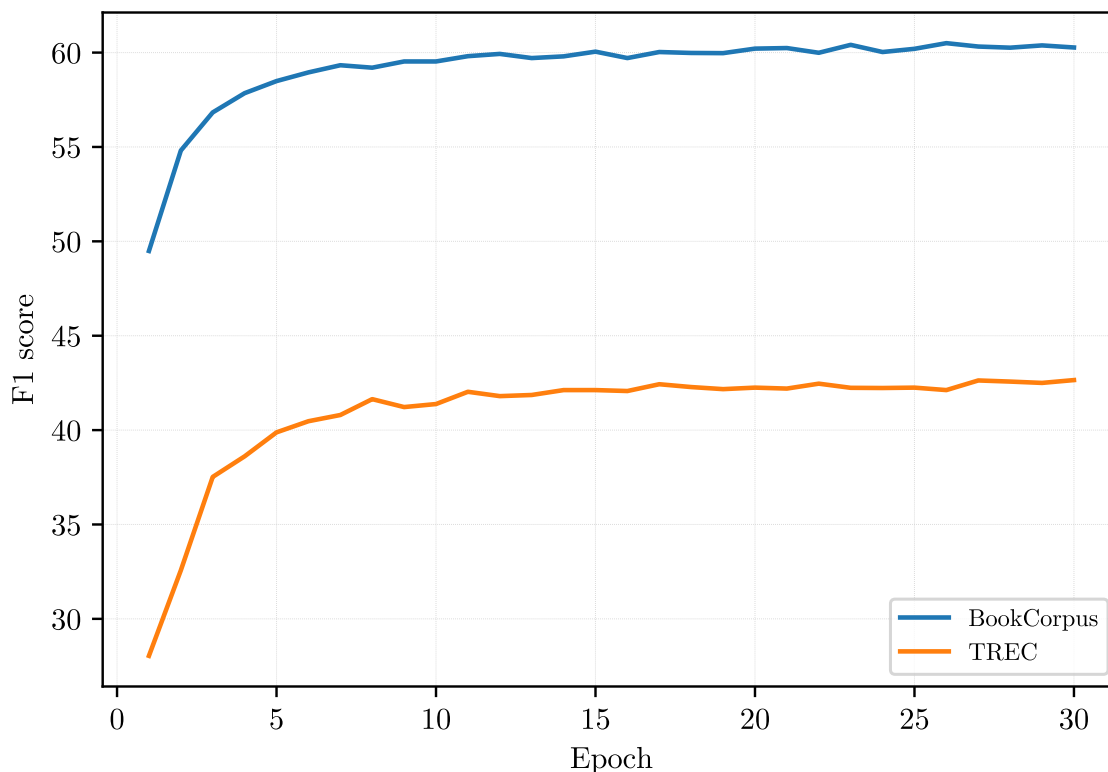


Figure 6-4: Inversion attack  $F_1$  score for different epochs and for the **BookCorpus** and **TREC** datasets.

On this latter observation, we speculate that the reason is in the size of the input document, and to the lesser extent different embedding mechanism. **BookCorpus** input document, as used in [SR20], is merely a sentence, while **TREC** document is a larger paragraph. We have tuned the maximum token sequence length (shorter sequences are padded, larger ones are truncated), but it yielded no improvement. From a purely combinatorial perspective, we conclude that the larger input loses more information while being embedded in the same-sized vector.

The case of a random attack is puzzling, as one would expect that there is no information to recover from an a-priori information-less (random) inputs. We have dived deeper than  $F_1$  score and inspected the actual words that the attack recovered

and observed that almost all of them are stop-words and punctuation. That is, the attack merely established that the input document contained a period, coma, “the” and “a”, and it happened to be right some of the time. While such inversion technically results in an  $F_1$  score as high as about 22%, it does not necessarily translate into an information leakage or a privacy violation. We therefore include an evaluation of the recovered words as part of our larger analysis.

We have run all experiments for both BookCorpus and TREC datasets, and we have noticed that although the absolute values of attack efficiency are higher for BookCorpus, all relations and correlations are the same for both datasets. We therefore report on the more relevant TREC dataset in the rest of the chapter.

### Public model

For the public model setting we have used the trained model  $\Upsilon$  from the baseline experiments and ran it against the encrypted embeddings. That is, for all documents  $x$  in  $\mathcal{D}_{\text{aux}}$ , we recovered a set of words from the encrypted embeddings  $\Upsilon(\text{ENC}(\Phi(x)))$  and compared it to the set of words in the original document  $\mathcal{W}(x)$ . We have repeated the process for different values of  $\beta \in \{1.0, \dots, 50.0\}$ , see Figure 6.5.

We observe that even for  $\beta = 0$ , the attack efficiency drops sharply compared to the plaintext baseline. We also see that the metric values drop further as  $\beta$  increases, starting slowly for small  $\beta$ , then accelerating for  $\beta$  10 to 30 and then slowing down again. Interestingly, we observe that the attack efficiency dives below the lower baseline, the random embeddings. We conclude that  $\beta \approx 27$ , which is about 2.5 times the maximum length of the input vectors, produces the security equivalent to symmetric encryption. Finally, we note that the fraction of stop-words grows fast as the security increases.

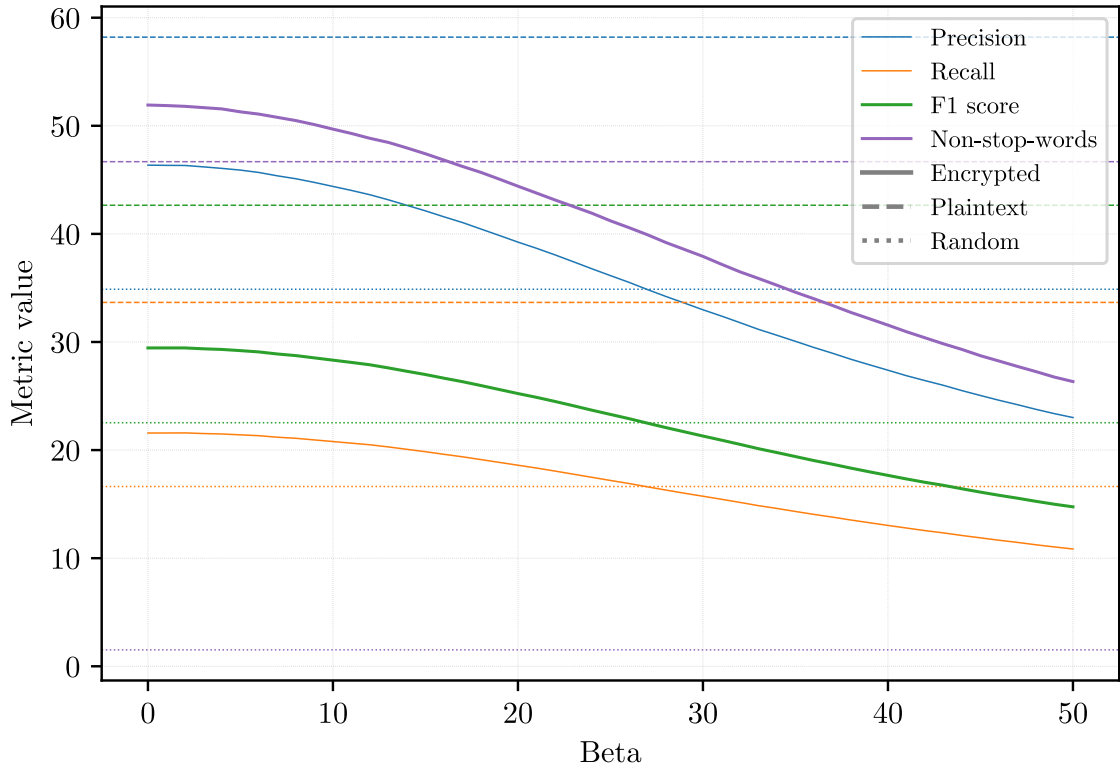


Figure 6-5: Inversion attack precision, recall,  $F_1$  score and percentage of non-stop-words among recovered words for plaintext attack (dashed), random attack (dotted) and different values of  $\beta$  (solid) for the TREC dataset. Horizontal bars depict the baselines.

### Private model

For the private model experiments, we have trained the model  $\Upsilon$  on the already encrypted embeddings for 30 epochs. That is, the model  $\Upsilon$  is trained on  $(\text{ENC}(\Phi(x)), \mathcal{W}(x))$  pairs from the training dataset and then we run predictions for the encrypted auxiliary embeddings, similar to public model. The training and validation datasets are encrypted with the same set of public and private parameters (i.e., same key  $k$ ,  $s$  and  $\beta$ ).

To produce the datasets, we have chosen  $\beta$  values of 0.0,  $\lceil \sqrt{\max N} \rceil$ ,  $\max N$  and  $2 \cdot \max N$  (for  $\max N \approx 11$  being the longest input length), as well as random

Dataset (encrypted with $\beta$ )	Precision	Recall	<b>F<sub>1</sub> score</b>	Non-stop-words
$\beta = 0$	41.59 %	23.36 %	<b>29.91 %</b>	3.67 %
$\beta = \lceil \sqrt{\max N} \rceil = 4$	41.91 %	23.75 %	<b>30.32 %</b>	4.96 %
$\beta \approx \max N = 11$	40.82 %	24.20 %	<b>30.39 %</b>	5.18 %
$\beta \approx 2 \cdot \max N = 22$	40.44 %	23.75 %	<b>29.92 %</b>	5.76 %
Random embeddings	35.91 %	26.49 %	<b>30.49 %</b>	0 %

Table 6.2: Black-box inversion attack performance for the private model experiments. The attack model  $\Upsilon$  is both trained and validated on the specified datasets.

embeddings, see Table 6.2. We immediately notice that the attack model accuracy metrics are similar among all five datasets, with the metrics for encrypted and random embeddings being almost equal. This result implies that for the case of private model — when the adversary has only black-box access to the system as a whole — the protection by the DCPE is absolute.

## 6.5 Search accuracy against security tradeoff

Equipped with the empirical data from *k-anon* experiments on search accuracy and attack efficiency, we can correlate these values with the security parameter, the approximation term  $\beta$ .

We note that the search accuracy degrades faster than attack efficiency, which implies that there is a tradeoff between functionality and security. Depending on the accuracy loss that the application can tolerate, the plot on Figure 6-6 can tell what level of protection against the inversion attack would be at that search accuracy. This plot also demonstrates the levels of accuracy and security for the  $\beta$  as a derivative of the dataset, in particular, the length of the longest vector embedding  $\max N$ .

We observe that at  $\beta = \sqrt{\max N}$ , the search accuracy corresponds to a plaintext dataset and the attack efficiency has dropped significantly compared to the plaintext

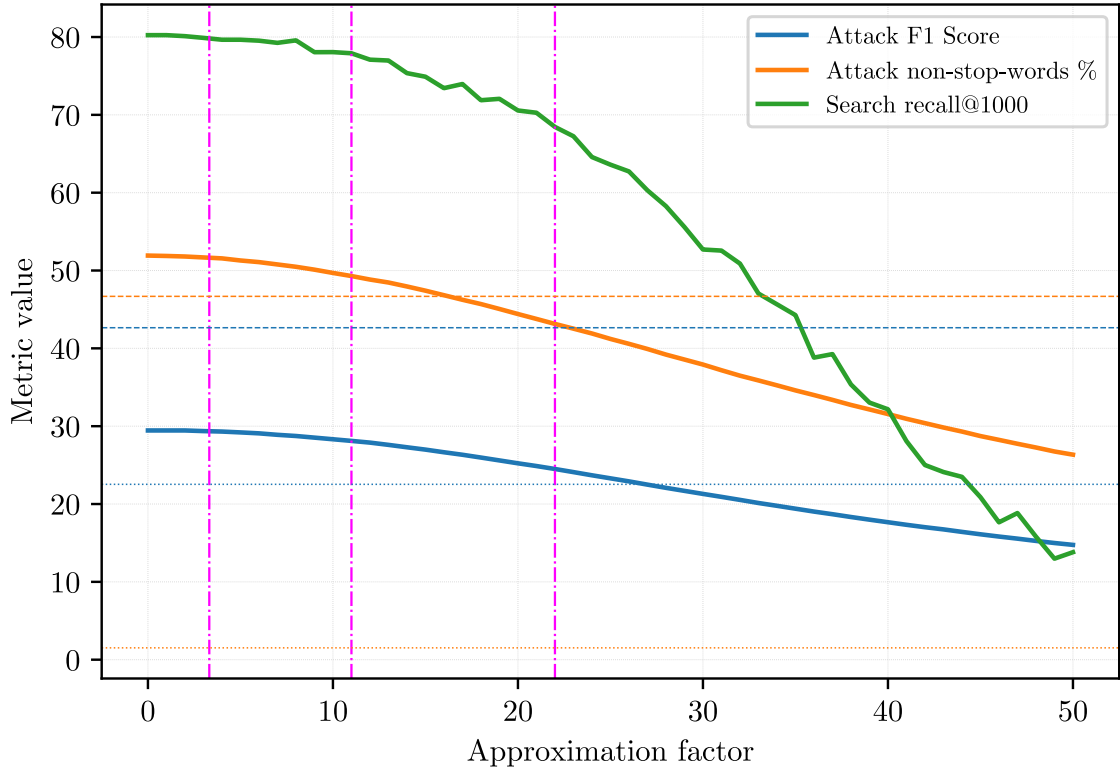


Figure 6-6: The correlation of search accuracy (the `recall@1000`) and the attack efficiency (the `F1` score and the `percent of non-stop-words`) with the approximation term  $\beta$ . The `vertical bars` show special values of  $\beta$ :  $\sqrt{\max N}$ ,  $\max N$  and  $2 \cdot \max N$  for  $\max N \approx 11$  being the length of the longest vector in the dataset.

version. We also see that at  $\beta = \max N$  both search accuracy and attack efficiency drop insubstantially compared to  $\beta = \sqrt{\max N}$ . Finally, we note that at  $\beta = 2 \cdot \max N$ , although both measures drop significantly, after that point the accuracy goes down much faster, which implies that increasing  $\beta$  beyond twice the longest vector size is pessimal. This value of  $\beta$  is also close to a point where the attack `F1` score intercepts the `F1` score of a random embedding attack, which further confirms that this value of  $\beta$  is optimal.

*Given the significant drop of attack efficiency for smaller  $\beta$  while retaining almost optimal search accuracy, we conclude that `k-anon` (applying a `Distance Comparison Preserving Encryption` to an embedding to protect `kNN` queries) is efficient. `k-anon`*

developed is also highly tuneable, with  $\beta$  corresponding to the application-specific accuracy and security requirements. Finally, the construction is cheap in terms of performance and functionality — the encryption of inputs is very fast and is done only once as a preprocessing step, and the existing algorithms work naturally over the encrypted data.

## 6.6 Conclusions

In this work, we developed and analyzed *k-anon*, a system that answers *kNN* queries securely in an outsourced setting. We adapted a *DCPE* scheme, ran experiments on search accuracy and inversion attack efficiency over encrypted inputs. We analyzed the correlation between the accuracy and security, and concluded that the approach provides meaningful and tunable security and attack resiliency guarantees for a comparatively small penalty in search accuracy.

### Future Work

An immediate deeper analysis of inversion attack over encrypted embeddings may include an evaluation of the words that the attack model returns beyond simple matching against the set of stop-words. It is reasonable to assume that with stronger encryption the *quality* of the predicted words may degrade, where the quality may be defined as the frequency of a word in the vocabulary, for example.

Another direction can be running other attacks against the embedding, beyond the inversion attack. These may be some of the Song and Raghunathan [SR20] attacks or adaptations of lots of general attacks against *ML* models, like membership attacks [SSSS17; YMMS21].

Finally, the *DCPE* adapted from [FGHO21] preserves Euclidean distance. It would be interesting to explore which results the inner-product distance comparison preserving encryption would produce.

## Chapter 7

# Conclusions and Future Work

In this thesis, we covered the concept of an outsourced database system and two types of adversaries — snapshot and persistent — that have different capabilities on an untrusted server. We focused on three query types — point, range and *k*-nearest-neighbor — and we have gone over the many works that propose systems that execute the relevant query types in a presence of an adversary. For the case of range queries in a snapshot adversary model, we provided an in-depth theoretical and practical analysis, an evaluation framework, a benchmark methodology, and its application to five OPE / ORE schemes and five secure range query protocols. For the case of point and range queries in a persistent adversary model, we offered an efficient and secure query mechanism,  $\mathcal{E}$ psolute, along with a novel definition of security, based on Differential Privacy. For the case of *k*-nearest-neighbor queries in a snapshot adversary model, we offered a similarity search protocol, *k-anon*, and the analysis of its search accuracy and susceptibility to certain inversion attacks.

The key take-away and future research directions that this thesis highlights are fourfold.

### Practicality and reproducibility

First and foremost, future research in the area of secure outsourced database systems should focus more prominently on practicality and reproducibility. After analyzing a plethora of works in the literature (see [BKR19; Bog+21]) we discovered that a large



fraction of constructions either do not have experiments, or their code is unavailable or otherwise not suitable for inspection, or the experimental results are not reproducible. We firmly believe in the reproducibility mission (such as SIGMOD<sup>1</sup> and pVLDB<sup>2</sup> efforts) and we encourage the works in the area to join the initiative.<sup>3</sup>

### Practicality of property-preserving encryption

Second, our works [BKR19; BKOZ22] demonstrate the practical value of property-preserving encryption as a component of a secure database system. While an argument can be made that a property-preserving encryption is inherently less-than-ideally secure from a purely cryptographic perspective, we counter that its value is much greater in a practical outsourced database system, which may not necessarily require perfect secrecy. A construction using such encryption scheme can be practical as long as the scheme’s performance is measured, its leakage is quantified and the effect of this leakage on the security of the entire system is properly analyzed.

### Practicality of using “heavy” primitives and protocols

Third, as our work,  $\mathcal{E}$ psolute, demonstrates, the primitives and protocols that are (rightly) considered heavyweight, such as ORAM and DP-sanitizers, can still be used efficiently in an outsourced system. In  $\mathcal{E}$ psolute, we show that a clever parallelization and optimization on both macro and micro levels can result in a very fast system overall.<sup>4</sup> We encourage practitioners to revisit using “heavy” primitives and protocols, such as ORAM, homomorphic encryption, garbled circuits, zero-knowledge proofs, in their systems.

---

<sup>1</sup><https://reproducibility.sigmod.org>

<sup>2</sup><https://vldb.org/pvldb/reproducibility/>

<sup>3</sup>We note that our work [BKR19] has received “Most Reproducible Paper” award.

<sup>4</sup>We have independently explored running  $\mathcal{E}$ psolute in a Trusted Execution Environment, and we have observed even higher performance.

### More query types

Finally, while we have covered three query types for a secure outsourced database system, we need more types to build a full-featured database that can compete with existing mainstream **RDBMS** like PostgreSQL. The directions include **JOIN**, **GROUP BY**, **AGGREGATE** queries and custom predicates.

## Appendix A

### Abstract of [BCET21]

In permissioned blockchain systems, participants are admitted to the network by receiving a credential from a certification authority. Each transaction processed by the network is required to be authorized by a valid participant who authenticates via her credential. Use case settings where privacy is a concern thus require proper privacy-preserving authentication and authorization mechanisms.

Anonymous credential schemes allow a user to authenticate while showing only those attributes necessary in a given setting. This makes them a great tool for authorizing transactions in permissioned blockchain systems based on the user’s attributes. In most setups, there is one distinct certification authority for each organization in the network. Consequently, the use of plain anonymous credential schemes still leaks the association of a user to the organization that issued her credentials. Camenisch, Drijvers and Dubovitskaya [CDD17] therefore suggest the use of a delegatable anonymous credential scheme to also hide that remaining piece of information.

In this paper, we propose the revocation and auditability — two functionalities that are necessary for real-world adoption — and integrate them into the scheme. We present a complete protocol, its security definition and the proof, and provide its open-source implementation. Our distributed-setting performance measurements show that the integration of the scheme with Hyperledger Fabric [And+18], while incurring an overhead in comparison to the less privacy-preserving solutions, is practical for settings with stringent privacy requirements.

## Appendix B

### Abstract of [NBK19]

**Motivation:** The complexity of protein-protein interactions (PPIs) is further compounded by the fact that an average protein consists of two or more domains, structurally and evolutionary independent subunits. Experimental studies have demonstrated that an interaction between a pair of proteins is not carried out by all domains constituting each protein, but rather by a select subset. However, finding which domains from each protein mediate the corresponding PPI is a challenging task.

**Results:** Here, we present Domain Interaction Statistical POTential (DISPOT), a simple knowledge-based statistical potential that estimates the propensity of an interaction between a pair of protein domains, given their SCOP family annotations. The statistical potential is derived based on the analysis of more than 352 000 structurally resolved protein-protein interactions obtained from DOMMINO, a comprehensive database on structurally resolved macromolecular interactions.

**Availability and implementation:** DISPOT is implemented in Python 2.7 and packaged as an open-source tool. DISPOT is implemented in two modes, *basic* and *auto-extraction*. The source code for both modes is available on [GitHub](#) and standalone docker images on [DockerHub](#). The web-server is freely available at [dispot.korkinlab.org](http://dispot.korkinlab.org).

**Contact:** [korkin@korkinlab.org](mailto:korkin@korkinlab.org) or [onarykov@wpi.edu](mailto:onarykov@wpi.edu)

**Supplementary information:** [Supplementary data](#) are available at *Bioinformatics* online.

# Bibliography

- [NET18] .NET Foundation. *Benchmark.NET*. <https://github.com/dotnet/BenchmarkDotNet>. 2018.
- [Aba+16] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. “TensorFlow: A System for Large-Scale Machine Learning”. In: *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*. OSDI’16. USENIX Association, 2016, pp. 265–283. DOI: [10.5281/zenodo.4724125](https://doi.org/10.5281/zenodo.4724125).
- [AKSX04] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. “Order Preserving Encryption for Numeric Data”. In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’04. Association for Computing Machinery, 2004, pp. 563–574. DOI: [10.1145/1007568.1007632](https://doi.org/10.1145/1007568.1007632).
- [And+18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolić, Sharon Weed Cocco, and Jason Yellick. “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains”. In: *Proceedings of the Thirteenth EuroSys Conference*. EuroSys ’18. 2018, pp. 1–15. DOI: [10.1145/3190508.3190538](https://doi.org/10.1145/3190508.3190538).
- [Ara+13] Arvind Arasu, Spyros Blanas, Ken Eguro, Manas Joglekar, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, Prasang Upadhyaya, and Ramarathnam Venkatesan. “Secure Database-as-a-Service with Cipherbase”. In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’13. Association for Computing Machinery, 2013, pp. 1033–1036. DOI: [10.1145/2463676.2467797](https://doi.org/10.1145/2463676.2467797).

- [Arg03] Lars Arge. “The Buffer Tree: A Technique for Designing Batched External Data Structures”. In: *Algorithmica* 37.1 (Sept. 2003), pp. 1–24. DOI: [10.1007/s00453-003-1021-x](https://doi.org/10.1007/s00453-003-1021-x).
- [Ary+98] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. “An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions”. In: *Journal of the ACM* 45.6 (Nov. 1998), pp. 891–923. DOI: [10.1145/293347.293348](https://doi.org/10.1145/293347.293348).
- [Baj+16] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. *MS MARCO: A Human Generated MACHine Reading COmprehension Dataset*. 2016. DOI: [10.48550/ARXIV.1611.09268](https://doi.org/10.48550/ARXIV.1611.09268).
- [BS13] Sumeet Bajaj and Radu Sion. “TrustedDB: A trusted hardware-based database with privacy and data confidentiality”. In: *IEEE Transactions on Knowledge and Data Engineering* 26.3 (2013), pp. 752–765. DOI: [10.1109/TKDE.2013.38](https://doi.org/10.1109/TKDE.2013.38).
- [BK15] Elaine Barker and John Kelsey. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. 2015. DOI: [10.6028/NIST.SP.800-90Ar1](https://doi.org/10.6028/NIST.SP.800-90Ar1).
- [BK16] Elaine Barker and John Kelsey. *Recommendation for Random Bit Generator (RBG) Constructions*. 2016.
- [Bat+17] Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel Kho, and Jennie Rogers. “SMCQL: Secure Querying for Federated Databases”. In: *Proceedings of the VLDB Endowment* 10.6 (Feb. 2017), pp. 673–684. DOI: [10.14778/3055330.3055334](https://doi.org/10.14778/3055330.3055334).
- [Bat+18] Johes Bater, Xi He, William Ehrich, Ashwin Machanavajjhala, and Jennie 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](https://doi.org/10.14778/3291264.3291274).
- [BM70] Rudolf Bayer and Edward M. McCreight. “Organization and Maintenance of Large Ordered Indices”. In: *Proceedings of the 1970 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*. SIGFIDET ’70. Association for Computing Machinery, 1970, pp. 107–141. DOI: [10.1145/1734663.1734671](https://doi.org/10.1145/1734663.1734671).
- [BBKN14] Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi 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](https://doi.org/10.1007/s10994-013-5404-1).

- [BNS13] 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](https://doi.org/10.1007/978-3-642-40328-6_26).
- [BNZ19] Amos Beimel, Kobbi Nissim, and Mohammad 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](https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2019.65).
- [BGRS99] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. “When Is “Nearest Neighbor” Meaningful?” In: *The International Conference on Database Theory – ICDT ’99*. Springer Berlin Heidelberg, 1999, pp. 217–235. DOI: [10.1007/3-540-49257-7\\_15](https://doi.org/10.1007/3-540-49257-7_15).
- [Bin+18] Vincent Bindschaedler, Paul Grubbs, David Cash, Thomas Ristenpart, and Vitaly Shmatikov. “The Tao of Inference in Privacy-Protected Databases”. In: *PVLDB* 11.11 (2018), pp. 1715–1728. DOI: [10.14778/3236187.3236217](https://doi.org/10.14778/3236187.3236217).
- [BLR13] 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](https://doi.org/10.1145/1374376.1374464).
- [BPP17] Tobias Boelter, Rishabh Poddar, and Raluca Ada Popa. “A Secure One-Roundtrip Index for Range Queries”. In: UCB/EECS-2017-7 (Apr. 2017). URL: [digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2017-7.pdf](https://digitalassets.lib.berkeley.edu/techreports/ucb/text/EECS-2017-7.pdf).
- [Bog17] **Dmytro Bogatov**. “Analysis of a Dynamic Voluntary Contribution Mechanism Public Good Game”. In: *IPE Journal* 26 (2017).
- [BCET21] **Dmytro Bogatov**, Angelo De Caro, Kaoutar Elkhiyaoui, and Björn Tackmann. “Anonymous Transactions with Revocation and Auditing in Hyperledger Fabric”. In: *International Conference on Cryptology and Network Security*. Springer, 2021. DOI: [10.1007/978-3-030-92548-2\\_23](https://doi.org/10.1007/978-3-030-92548-2_23).
- [BH16] **Dmytro Bogatov** and Jillian Rose Hennessy. “Data MATTERS: Customizing Economic Indices to Measure State Competitiveness”. In: *WPI Library* (2016). URL: [digital.wpi.edu/concern/student\\_works/b2773x39g](https://digital.wpi.edu/concern/student_works/b2773x39g).

- [Bog+21] **Dmytro Bogatov**, Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O’Neill. “ $\epsilon$ -solute: Efficiently Querying Databases While Providing Differential Privacy”. In: *Proceedings of the 2021 ACM SIG-SAC Conference on Computer and Communications Security – CCS ’2021*. 2021. DOI: [10.1145/3460120.3484786](https://doi.org/10.1145/3460120.3484786).
- [BKOZ22] **Dmytro Bogatov**, George Kollios, Adam O’Neill, and Hamed Zamani. “*k*-anon: Secure Similarity Search in Outsourced Databases”. Apr. 2022.
- [BKR19] **Dmytro Bogatov**, George Kollios, and Leonid 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](https://doi.org/10.14778/3324301.3324309).
- [Bog21] Daria Bogatova. “Neurovascular Interface in Aging Zebrafish”. <https://bogatova.org/assets/docs/neurovascular-interface-in-aging-zebrafish-daria-bogatova-bachelor-thesis.pdf>. Bachelor’s Thesis. Goethe University Frankfurt am Main, May 2021.
- [BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam 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](https://doi.org/10.1007/978-3-642-01001-9_13).
- [BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O’Neill. “Order-Preserving Encryption Revisited: Improved Security Analysis and Alternative Solutions”. In: *Advances in Cryptology – CRYPTO 2011*. Springer Berlin Heidelberg, 2011, pp. 578–595. DOI: [10.1007/978-3-642-22792-9\\_33](https://doi.org/10.1007/978-3-642-22792-9_33).
- [Bon+15] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. “Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2015, pp. 563–594. DOI: [10.1007/978-3-662-46803-6\\_19](https://doi.org/10.1007/978-3-662-46803-6_19).
- [BNSV15] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil 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](https://doi.org/10.1109/FOCS.2015.45).
- [BZ16] Mark Bun and Mark Zhandry. “Order-Revealing Encryption and the Hardness of Private Learning”. In: *Theory of Cryptography*. Springer Berlin Heidelberg, 2016, pp. 176–206. DOI: [10.1007/978-3-662-49096-9\\_8](https://doi.org/10.1007/978-3-662-49096-9_8).



- [Bur+05] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. “Learning to Rank Using Gradient Descent”. In: *Proceedings of the 22nd International Conference on Machine Learning*. ICML '05. Association for Computing Machinery, 2005, pp. 89–96. DOI: [10.1145/1102351.1102363](https://doi.org/10.1145/1102351.1102363).
- [CDD17] Jan Camenisch, Manu Drijvers, and Maria Dubovitskaya. “Practical UC-Secure Delegatable Credentials with Attributes and Their Application to Blockchain”. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Association for Computing Machinery, 2017, pp. 683–699. DOI: [10.1145/3133956.3134025](https://doi.org/10.1145/3133956.3134025).
- [CGPR15] David Cash, Paul Grubbs, Jason Perry, and Thomas 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](https://doi.org/10.1145/2810103.2813700).
- [Cas+14] David Cash, Joseph Jaeger, Stanislaw Jarecki, Charanjit S. Jutla, Hugo Krawczyk, Marcel-Cătălin Rosu, and Michael Steiner. “Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation”. In: *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. 2014. DOI: [10.14722/ndss.2014.23264](https://doi.org/10.14722/ndss.2014.23264).
- [Cas+13] David Cash, Stanislaw Jarecki, Charanjit Jutla, Hugo Krawczyk, Marcel-Cătălin Roşu, and Michael Steiner. “Highly-Scalable Searchable Symmetric Encryption with Support for Boolean Queries”. In: *Advances in Cryptology – CRYPTO 2013*. Springer Berlin Heidelberg, 2013, pp. 353–373. DOI: [10.1007/978-3-642-40041-4\\_20](https://doi.org/10.1007/978-3-642-40041-4_20).
- [Cas+18] David Cash, Feng-Hao Liu, Adam O’Neill, Mark Zhandry, and Cong Zhang. “Parameter-hiding order revealing encryption”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2018, pp. 181–210. DOI: [10.1007/978-3-030-03326-2\\_7](https://doi.org/10.1007/978-3-030-03326-2_7).
- [CCMS19] T-H. Hubert Chan, Kai-Min Chung, Bruce M. Maggs, and Elaine Shi. “Foundations of Differentially Oblivious Algorithms”. In: *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '19. Society for Industrial and Applied Mathematics, 2019.
- [CXL16] Zhao Chang, Dong Xie, and Feifei Li. “Oblivious RAM: A dissection and experimental evaluation”. In: *Proceedings of the VLDB Endowment* 9.12 (2016), pp. 1113–1124. DOI: [10.14778/2994509.2994528](https://doi.org/10.14778/2994509.2994528).

- [CLT16] Binyi Chen, Huijia Lin, and Stefano Tessaro. “Oblivious Parallel RAM: Improved Efficiency and Generic Constructions”. In: *Theory of Cryptography*. Ed. by Eyal Kushilevitz and Tal Malkin. Springer Berlin Heidelberg, 2016, pp. 205–234. DOI: [10.1007/978-3-662-49099-0\\_8](https://doi.org/10.1007/978-3-662-49099-0_8).
- [CLWW16] Nathan Chenette, Kevin Lewi, Stephen A. Weis, and David J. Wu. “Practical Order-Revealing Encryption with Limited Leakage”. In: *Revised Selected Papers of the 23rd International Conference on Fast Software Encryption - Volume 9783*. FSE 2016. Springer-Verlag, 2016. DOI: [10.1007/978-3-662-52993-5\\_24](https://doi.org/10.1007/978-3-662-52993-5_24).
- [Cra09] Nick 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](https://doi.org/10.1007/978-0-387-39940-9_488).
- [Cui+20] Ningning Cui, Xiaochun Yang, Bin Wang, Jianxin Li, and Guoren Wang. “SVkNN: Efficient Secure and Verifiable k-Nearest Neighbor Query on the Cloud Platform\*”. In: *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 2020, pp. 253–264. DOI: [10.1109/ICDE48307.2020.00029](https://doi.org/10.1109/ICDE48307.2020.00029).
- [CD21] Pádraig Cunningham and Sarah Jane Delany. “K-Nearest Neighbour Classifiers - A Tutorial”. In: *ACM Computing Surveys* 54.6 (July 2021). DOI: [10.1145/3459665](https://doi.org/10.1145/3459665).
- [Dam64] Fred J. Damerau. “A Technique for Computer Detection and Correction of Spelling Errors”. In: *Communications of the ACM* 7.3 (Mar. 1964), pp. 171–176. DOI: [10.1145/363958.363994](https://doi.org/10.1145/363958.363994).
- [Dav19] Chair Mike Cowlishaw David G. Hough. “IEEE Standard for Floating-Point Arithmetic”. In: *IEEE Std 754-2019 (Revision of IEEE 754-2008)* (2019), pp. 1–84. DOI: [10.1109/IEEESTD.2019.8766229](https://doi.org/10.1109/IEEESTD.2019.8766229).
- [Dav16] Tom Woller David Kaplan Jeremy Powell. “AMD memory encryption”. In: *White Paper* (2016). URL: [developer.amd.com/wordpress/media/2013/12/AMD\\_Memory\\_Encryption\\_Whitepaper\\_v7-Public.pdf](https://developer.amd.com/wordpress/media/2013/12/AMD_Memory_Encryption_Whitepaper_v7-Public.pdf).
- [DPPS20] Ioannis Demertzis, Dimitrios Papadopoulos, Charalampos Papamanthou, and Saurabh Shintre. “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.
- [Dem+16] Ioannis Demertzis, Stavros Papadopoulos, Odysseas Papapetrou, Antonios Deligiannakis, and Minos 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](https://doi.org/10.1145/2882903.2882911).

- [DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: [10.18653/v1/N19-1423](https://doi.org/10.18653/v1/N19-1423).
- [DDC16] F. Betül Durak, Thomas M. DuBuisson, and David Cash. “What Else is Revealed by Order-Revealing Encryption?” In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16*. Association for Computing Machinery, 2016, pp. 1155–1166. DOI: [10.1145/2976749.2978379](https://doi.org/10.1145/2976749.2978379).
- [Dwo+06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni 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](https://doi.org/10.1007/11761679_29).
- [DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. “Calibrating noise to sensitivity in private data analysis”. In: *Theory of cryptography conference*. Springer, 2006, pp. 265–284. DOI: [10.1007/11681878\\_14](https://doi.org/10.1007/11681878_14).
- [DNPR10] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. “Differential privacy under continual observation”. In: *Proceedings of the forty-second ACM symposium on Theory of computing*. 2010. DOI: [10.1145/1806689.1806787](https://doi.org/10.1145/1806689.1806787).
- [Dwo01] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. 2001. DOI: [10.6028/NIST.SP.800-38A](https://doi.org/10.6028/NIST.SP.800-38A).
- [Dwo07] Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. 2007. DOI: [10.6028/NIST.SP.800-38D](https://doi.org/10.6028/NIST.SP.800-38D).
- [Dwo+01] Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and James Dray. *Advanced Encryption Standard (AES)*. en. 2001. DOI: [10.6028/NIST.FIPS.197](https://doi.org/10.6028/NIST.FIPS.197).
- [ESJ14] Yousef Elmehdwi, Bharath K. Samanthula, and Wei Jiang. “Secure k-nearest neighbor query over encrypted data in outsourced environments”. In: *2014 IEEE 30th International Conference on Data Engineering*. 2014, pp. 664–675. DOI: [10.1109/ICDE.2014.6816690](https://doi.org/10.1109/ICDE.2014.6816690).

- [EWSG04] Yuval Elovici, Ronen Waisenberg, Erez Shmueli, and Ehud Gudes. “A Structure Preserving Database Encryption Scheme”. In: *Secure Data Management*. Springer Berlin Heidelberg, 2004, pp. 28–40. DOI: [10.1007/978-3-540-30073-1\\_3](https://doi.org/10.1007/978-3-540-30073-1_3).
- [ELL18] Jieun Eom, Dong Hoon Lee, and Kwangsu Lee. “Multi-Client Order-Revealing Encryption”. In: *IEEE Access* 6 (2018), pp. 45458–45472. DOI: [10.1109/ACCESS.2018.2864991](https://doi.org/10.1109/ACCESS.2018.2864991).
- [EZ19] Saba Eskandarian and Matei Zaharia. “ObliDB: Oblivious query processing for secure databases”. In: *Proceedings of the VLDB Endowment* 13.2 (2019), pp. 169–183. DOI: [10.14778/3364324.3364331](https://doi.org/10.14778/3364324.3364331).
- [FGHO21] Georg Fuchsbauer, Riddhi Ghosal, Nathan Hauke, and Adam O’Neill. *Approximate Distance-Comparison-Preserving Symmetric Encryption*. Cryptology ePrint Archive, Report 2021/1666. <https://ia.cr/2021/1666>. 2021.
- [Fuh+17] Benny Fuhry, Raad Bahmani, Ferdinand Brasser, Florian Hahn, Florian Kerschbaum, and Ahmad-Reza Sadeghi. “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](https://doi.org/10.1007/978-3-319-61176-1_22).
- [Gen10] Craig Gentry. “Computing arbitrary functions of encrypted data”. In: *Communications of the ACM* 53.3 (2010), pp. 97–105. DOI: [10.1145/1666420.1666444](https://doi.org/10.1145/1666420.1666444).
- [Ghi+08] Gabriel Ghinita, Panos Kalnis, Ali Khoshgozaran, Cyrus Shahabi, and Kian-Lee Tan. “Private Queries in Location Based Services: Anonymizers Are Not Necessary”. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’08. Association for Computing Machinery, 2008, pp. 121–132. DOI: [10.1145/1376616.1376631](https://doi.org/10.1145/1376616.1376631).
- [Gol87] 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](https://doi.org/10.1145/28395.28416).
- [GO96] 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](https://doi.org/10.1145/233551.233553).
- [GLMP18] Paul Grubbs, Marie-Sarah Lacharité, Brice Minaud, and Kenneth 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](https://doi.org/10.1145/3243734.3243864).

- [GRS17] 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](https://doi.org/10.1145/3102980.3103007).
- [Gru+17] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas 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](https://doi.org/10.1109/SP.2017.44).
- [GJW19] 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](https://doi.org/10.1145/3319535.3363210).
- [Haa+17] Helene Haagh, Yue Ji, Chenxing Li, Claudio Orlandi, and Yifan Song. “Revealing Encryption for Partial Ordering”. In: *Cryptography and Coding*. Springer International Publishing, 2017, pp. 3–22. DOI: [10.1007/978-3-319-71045-7\\_1](https://doi.org/10.1007/978-3-319-71045-7_1).
- [HILM02] Hakan Hacigümüş, Bala Iyer, Chen Li, and Sharad Mehrotra. “Executing SQL over Encrypted Data in the Database-Service-Provider Model”. In: *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’02. New York, NY, USA, 2002, pp. 216–227. DOI: [10.1145/564691.564717](https://doi.org/10.1145/564691.564717).
- [HRMS10] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. “Boosting the Accuracy of Differentially Private Histograms through Consistency”. In: *Proceedings of the VLDB Endowment* 3.1–2 (Sept. 2010), pp. 1021–1032. DOI: [10.14778/1920841.1920970](https://doi.org/10.14778/1920841.1920970).
- [HR10] Viet Tung Hoang and Phillip Rogaway. “On Generalized Feistel Networks”. In: *Proceedings of the 30th Annual Conference on Advances in Cryptology*. CRYPTO’10. Springer-Verlag, 2010, pp. 613–630.
- [Hof+21] Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. “Intra-Document Cascading: Learning to Select Passages for Neural Document Ranking”. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 1349–1358. DOI: [10.1145/3404835.3462889](https://doi.org/10.1145/3404835.3462889).
- [HMCK12] Bijit Hore, Sharad Mehrotra, Mustafa Canim, and Murat Kantarcioglu. “Secure multidimensional range queries over outsourced data”. In: *VLDB Journal* 21.3 (2012), pp. 333–358. DOI: [10.1007/s00778-011-0245-7](https://doi.org/10.1007/s00778-011-0245-7).
- [Hou04] Russ Housley. *Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)*. RFC 3686. Jan. 2004. URL: <https://tools.ietf.org/html/rfc3686>.

- [Hsu+14] Justin Hsu, Marco Gaboardi, Andreas Haeberlen, Sanjeev Khanna, Arjun Narayan, Benjamin C Pierce, and Aaron Roth. “Differential privacy: An economic method for choosing epsilon”. In: *2014 IEEE 27th Computer Security Foundations Symposium*. IEEE, 2014, pp. 398–410. DOI: [10.1109/CSF.2014.35](https://doi.org/10.1109/CSF.2014.35).
- [HXRC11] Haibo Hu, Jianliang Xu, Chushi Ren, and Byron Choi. “Processing private queries over untrusted data cloud through privacy homomorphism”. In: *2011 IEEE 27th International Conference on Data Engineering*. 2011, pp. 601–612. DOI: [10.1109/ICDE.2011.5767862](https://doi.org/10.1109/ICDE.2011.5767862).
- [HH19] Walt Hubis and Eric Hibbard. “IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices”. In: *IEEE Std 1619-2018 (Revision of IEEE Std 1619-2007)* (2019), pp. 1–41. DOI: [10.1109/IEEESTD.2019.8637988](https://doi.org/10.1109/IEEESTD.2019.8637988).
- [IM98] Piotr Indyk and Rajeev Motwani. “Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality”. In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*. STOC ’98. Association for Computing Machinery, 1998, pp. 604–613. DOI: [10.1145/276698.276876](https://doi.org/10.1145/276698.276876).
- [IKLO16] Yuval Ishai, Eyal Kushilevitz, Steve Lu, and Rafail Ostrovsky. “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](https://doi.org/10.1007/978-3-319-29485-8_6).
- [IKK12] 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.
- [IKK14] 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](https://doi.org/10.1145/2557547.2557561).
- [JS11] Tibor Jager and Juraj Somorovsky. “How to Break XML Encryption”. In: *Proceedings of the 18th ACM Conference on Computer and Communications Security*. CCS ’11. Association for Computing Machinery, 2011, pp. 413–422. DOI: [10.1145/2046707.2046756](https://doi.org/10.1145/2046707.2046756).
- [Jan95] Jan Jannink. “Implementing Deletion in B+-Trees”. In: *ACM SIGMOD Record* 24.1 (Mar. 1995), pp. 33–38. DOI: [10.1145/202660.202666](https://doi.org/10.1145/202660.202666).

- [JK02] Kalervo Järvelin and Jaana Kekäläinen. “Cumulated Gain-Based Evaluation of IR Techniques”. In: *ACM Transactions on Information Systems* 20.4 (Oct. 2002), pp. 422–446. ISSN: 1046-8188. DOI: [10.1145/582415.582418](https://doi.org/10.1145/582415.582418).
- [JDJ21] Jeff Johnson, Matthijs Douze, and Hervé Jégou. “Billion-Scale Similarity Search with GPUs”. In: *IEEE Transactions on Big Data* 7.3 (2021), pp. 535–547. DOI: [10.1109/TBDATA.2019.2921572](https://doi.org/10.1109/TBDATA.2019.2921572).
- [JPS21] Mireya Jurado, Catuscia Palamidessi, and Geoffrey Smith. “A Formal Information-Theoretic Leakage Analysis of Order-Revealing Encryption”. In: *2021 IEEE 34th Computer Security Foundations Symposium (CSF)*. 2021, pp. 1–16. DOI: [10.1109/CSF51468.2021.00046](https://doi.org/10.1109/CSF51468.2021.00046).
- [KS88] Voratas Kachitvichyanukul and Bruce W. Schmeiser. “Algorithm 668: H2PEC: Sampling from the Hypergeometric Distribution”. In: *ACM Transactions on Mathematical Software* 14.4 (Dec. 1988), pp. 397–398. DOI: [10.1145/50063.214387](https://doi.org/10.1145/50063.214387).
- [KAK10] Hasan Kadhém, Toshiyuki Amagasa, and Hiroyuki Kitagawa. “MV-OPES: Multivalued-Order Preserving Encryption Scheme: Novel Scheme for Encrypting Integer Value to Many Different Values”. In: *IEICE Transactions on Information and Systems* E93.D.9 (2010), pp. 2520–2533. DOI: [10.1587/transinf.E93.D.2520](https://doi.org/10.1587/transinf.E93.D.2520).
- [KAK13] Hasan Kadhém, Toshiyuki Amagasa, and Hiroyuki Kitagawa. “Optimization Techniques for Range Queries in the Multivalued-partial Order Preserving Encryption Scheme”. In: *Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Springer Berlin Heidelberg, 2013, pp. 338–353. DOI: [10.1007/978-3-642-29764-9\\_23](https://doi.org/10.1007/978-3-642-29764-9_23).
- [Kap+20] Haim Kaplan, Katrina Ligett, Yishay Mansour, Moni Naor, and Uri Stemmer. “Privately Learning Thresholds: Closing Exponential Gap”. In: *Proceedings of Thirty Third Conference on Learning Theory*. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2263–2285.
- [KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Second. Chapman & Hall/CRC, 2014. ISBN: 9781466570269.
- [KKNO16] Georgios Kellaris, George Kollios, Kobbi Nissim, and Adam O’neill. “Generic attacks on secure outsourced databases”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 1329–1340. DOI: [10.1145/2976749.2978386](https://doi.org/10.1145/2976749.2978386).
- [Ker15] Florian Kerschbaum. “Frequency-Hiding Order-Preserving Encryption”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS ’15. Association for Computing Machinery, 2015, pp. 656–667. DOI: [10.1145/2810103.2813629](https://doi.org/10.1145/2810103.2813629).

- [KS14] Florian Kerschbaum and Axel Schroepfer. “Optimal Average-Complexity Ideal-Security Order-Preserving Encryption”. In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’14. Association for Computing Machinery, 2014, pp. 275–286. DOI: [10.1145/2660267.2660277](https://doi.org/10.1145/2660267.2660277).
- [KT19] Florian Kerschbaum and Anselme Tueno. “An Efficiently Searchable Encrypted Data Structure for Range Queries”. In: *Computer Security – ESORICS 2019*. Springer International Publishing, 2019, pp. 344–364. DOI: [10.1007/978-3-030-29962-0\\_17](https://doi.org/10.1007/978-3-030-29962-0_17).
- [KSS13] Ali Khoshgozaran, Houtan Shirani-Mehr, and Cyrus Shahabi. “Blind Evaluation of Location Based Queries Using Space Transformation to Preserve Location Privacy”. In: *Geoinformatica 17.4* (Oct. 2013), pp. 599–634. DOI: [10.1007/s10707-012-0172-9](https://doi.org/10.1007/s10707-012-0172-9).
- [KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. “Delegatable Pseudorandom Functions and Applications”. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS ’13. Association for Computing Machinery, 2013, pp. 669–684. DOI: [10.1145/2508859.2516668](https://doi.org/10.1145/2508859.2516668).
- [KB14] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. DOI: [10.48550/ARXIV.1412.6980](https://doi.org/10.48550/ARXIV.1412.6980).
- [Knu16] Donald Ervin Knuth. *Seminumerical algorithms*. 3rd ed. Vol. 2. Addison-Wesley, 2016, pp. 145–146.
- [Koc+19] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. “Spectre Attacks: Exploiting Speculative Execution”. In: *2019 IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 1–19. DOI: [10.1109/SP.2019.00002](https://doi.org/10.1109/SP.2019.00002).
- [KPT20] 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](https://doi.org/10.1109/SP40000.2020.00029).
- [LMP18] 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](https://doi.org/10.1109/SP.2018.00002).
- [LN18] Kasper Green Larsen and Jesper Buus Nielsen. “Yes, There is an Oblivious RAM Lower Bound!” In: *Advances in Cryptology – CRYPTO 2018*. Springer International Publishing, 2018, pp. 523–542. DOI: [10.1007/978-3-319-96881-0\\_18](https://doi.org/10.1007/978-3-319-96881-0_18).



- [LSY20] Kasper Green Larsen, Mark Simkin, and Kevin Yeo. “Lower Bounds for Multi-server Oblivious RAMs”. In: *Theory of Cryptography*. Springer International Publishing, 2020, pp. 486–503. DOI: [10.1007/978-3-030-64375-1\\_17](https://doi.org/10.1007/978-3-030-64375-1_17).
- [LLLT19] Xinyu Lei, Alex X. Liu, Rui Li, and Guan-Hua Tu. “SecEQP: A Secure and Efficient Scheme for SkNN Query Problem Over Encrypted Geodata on Cloud”. In: *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 2019, pp. 662–673. DOI: [10.1109/ICDE.2019.00065](https://doi.org/10.1109/ICDE.2019.00065).
- [Lev66] Vladimir Iosifovich Levenshtein. “Binary codes capable of correcting deletions, insertions and reversals.” In: *Soviet Physics Doklady* 10.8 (Feb. 1966), pp. 707–710.
- [LW16] Kevin Lewi and David 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](https://doi.org/10.1145/2976749.2978376).
- [LSP15] Frank Li, Richard Shin, and Vern Paxson. “Exploring Privacy Preservation in Outsourced K-Nearest Neighbors with Multiple Data Owners”. In: *Proceedings of the 2015 ACM Workshop on Cloud Computing Security Workshop*. CCSW ’15. Association for Computing Machinery, 2015, pp. 53–64. DOI: [10.1145/2808425.2808430](https://doi.org/10.1145/2808425.2808430).
- [LWZ19] Yuan Li, Hongbing Wang, and Yunlei Zhao. “Delegatable Order-Revealing Encryption”. In: *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. Asia CCS ’19. Association for Computing Machinery, 2019, pp. 134–147. DOI: [10.1145/3321705.3329829](https://doi.org/10.1145/3321705.3329829).
- [LWHZ21] Yuan Li, Xing-Chen Wang, Lin Huang, and Yun-Lei Zhao. “Order-Revealing Encryption: File-Injection Attack and Forward Security”. In: *Journal of Computer Science and Technology* 36.4 (Aug. 2021), pp. 877–895. DOI: [10.1007/s11390-020-0060-y](https://doi.org/10.1007/s11390-020-0060-y).
- [LW12] Dongxi Liu and Shenlu Wang. “Programmable Order-Preserving Secure Index for Encrypted Database Query”. In: *2012 IEEE Fifth International Conference on Cloud Computing*. 2012, pp. 502–509. DOI: [10.1109/CLOUD.2012.65](https://doi.org/10.1109/CLOUD.2012.65).
- [LW13] Dongxi Liu and Shenlu Wang. “Nonlinear order preserving index for encrypted database query in service cloud environments”. In: *Concurrency and Computation: Practice and Experience* (2013), pp. 1967–1984. DOI: [10.1002/cpe.2992](https://doi.org/10.1002/cpe.2992).

- [LCZ17] Zhe Liu, Kim-Kwang Raymond Choo, and Minghao Zhao. “Practical-oriented protocols for privacy-preserving outsourced big data analysis: Challenges and future research directions”. In: *Computers & Security* 69 (2017), pp. 97–113. DOI: [10.1016/j.cose.2016.12.006](https://doi.org/10.1016/j.cose.2016.12.006).
- [Lv+21] Chunyang Lv, Jianfeng Wang, Shi-Feng Sun, Yunling Wang, Saiyu Qi, and Xiaofeng Chen. “Efficient Multi-client Order-Revealing Encryption and Its Applications”. In: *Computer Security – ESORICS 2021*. Springer International Publishing, 2021, pp. 44–63. DOI: [10.1007/978-3-030-88428-4\\_3](https://doi.org/10.1007/978-3-030-88428-4_3).
- [Lyn18] Ben Lynn. *Pairings-based Crypto (PBC)*. 2018. URL: <https://crypto.stanford.edu/pbc/> (visited on 08/15/2018).
- [MRS18] Matteo Maffei, Manuel Reinert, and Dominique Schröder. “On the Security of Frequency-Hiding Order-Preserving Encryption”. In: *Cryptology and Network Security*. Springer International Publishing, 2018, pp. 51–70. DOI: [10.1007/978-3-030-02641-7\\_3](https://doi.org/10.1007/978-3-030-02641-7_3).
- [McS09] Frank 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. DOI: [10.1145/1559845.1559850](https://doi.org/10.1145/1559845.1559850).
- [MPRV09] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. “Computational differential privacy”. In: *Annual International Cryptology Conference*. Springer. 2009, pp. 126–142. DOI: [10.1007/978-3-642-03356-8\\_8](https://doi.org/10.1007/978-3-642-03356-8_8).
- [Mis+18] Pratyush Mishra, Rishabh Poddar, Jerry Chen, Alessandro Chiesa, and Raluca Ada Popa. “Oblix: An efficient oblivious search index”. In: *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2018, pp. 279–296. DOI: [10.1109/SP.2018.00045](https://doi.org/10.1109/SP.2018.00045).
- [MCA07] Mohamed F. Mokbel, Chi-Yin Chow, and Walid G. Aref. “The New Casper: A Privacy-Aware Location-Based Database Server”. In: *2007 IEEE 23rd International Conference on Data Engineering*. 2007. DOI: [10.1109/ICDE.2007.369052](https://doi.org/10.1109/ICDE.2007.369052).
- [MRS09] Ben Morris, Phillip Rogaway, and Till Stegers. “How to Encipher Messages on a Small Domain”. In: *Advances in Cryptology - CRYPTO 2009*. Springer Berlin Heidelberg, 2009, pp. 286–302. DOI: [10.1007/978-3-642-03356-8\\_17](https://doi.org/10.1007/978-3-642-03356-8_17).
- [ML14] Marius Muja and David G. Lowe. “Scalable Nearest Neighbor Algorithms for High Dimensional Data”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.11 (2014), pp. 2227–2240. DOI: [10.1109/TPAMI.2014.2321376](https://doi.org/10.1109/TPAMI.2014.2321376).

- [Mur+20] Kit Murdock, David Oswald, Flavio D. Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. “Plundervolt: Software-based Fault Injection Attacks against Intel SGX”. In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 1466–1482. DOI: [10.1109/SP40000.2020.00057](https://doi.org/10.1109/SP40000.2020.00057).
- [NBK19] Oleksandr Narykov, **Dmytro Bogatov**, and Dmitry Korkin. “DISPOT: a simple knowledge-based protein domain interaction statistical potential”. In: *Bioinformatics* 35.24 (July 2019), pp. 5374–5378. DOI: [10.1093/bioinformatics/btz587](https://doi.org/10.1093/bioinformatics/btz587).
- [NKW15] 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](https://doi.org/10.1145/2810103.2813651).
- [NBXO15] Batyrilan Nurbekov, **Dmytro Bogatov**, Jiacong S Xu, and Richard Joseph O’Brien. “Investment Trading And Risk Management: Scientifically Developing and Analyzing Trading Systems”. In: *WPI Library* (2015). URL: [digital.wpi.edu/concern/parent/p2676w12m/file\\_sets/gb19f6396](http://digital.wpi.edu/concern/parent/p2676w12m/file_sets/gb19f6396).
- [OSC04] Gultekin Ozsoyoglu, David A. Singer, and Sun S. Chung. “Anti-Tamper Databases”. In: *Data and Applications Security XVII: Status and Prospects*. Ed. by Sabrina De Capitani di Vimercati, Indrakshi Ray, and Indrajit Ray. Boston, MA: Springer US, Jan. 2004, pp. 133–146. DOI: [10.1007/1-4020-8070-0\\_10](https://doi.org/10.1007/1-4020-8070-0_10).
- [Pap+16] Antonis Papadimitriou, Ranjita Bhagwan, Nishanth Chandran, Ramachandran Ramjee, Andreas Haeberlen, Harmeet Singh, Abhishek Modi, and Saikrishna Badrinarayanan. “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.
- [PBP10] Stavros Papadopoulos, Spiridon Bakiras, and Dimitris Papadias. “Nearest Neighbor Search with Strong Location Privacy”. In: *Proceedings of the VLDB Endowment* 3.1–2 (Sept. 2010), pp. 619–629. DOI: [10.14778/1920841.1920920](https://doi.org/10.14778/1920841.1920920).
- [PBP19] 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](https://doi.org/10.14778/3342263.3342641).

- [PLZ13] Raluca Ada Popa, Frank H. Li, and Nickolai Zeldovich. “An Ideal Security Protocol for Order-Preserving Encoding”. In: *2013 IEEE Symposium on Security and Privacy*. 2013, pp. 463–477. DOI: [10.1109/SP.2013.38](https://doi.org/10.1109/SP.2013.38).
- [PRZB11] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. “CryptDB: Protecting Confidentiality with Encrypted Query Processing”. In: *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. SOSP '11. Association for Computing Machinery, 2011, pp. 85–100. DOI: [10.1145/2043556.2043566](https://doi.org/10.1145/2043556.2043566).
- [PVC18] 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](https://doi.org/10.1109/SP.2018.00025).
- [QYL13] Wahbeh Qardaji, Weining Yang, and Ninghui 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](https://doi.org/10.14778/2556549.2556576).
- [QA08] Yinian Qi and Mikhail J. Atallah. “Efficient Privacy-Preserving  $k$ -Nearest-Neighbor Search”. In: *2008 The 28th International Conference on Distributed Computing Systems*. 2008, pp. 311–319. DOI: [10.1109/ICDCS.2008.79](https://doi.org/10.1109/ICDCS.2008.79).
- [RACY16] Daniel S. Roche, Daniel Apon, Seung Geol Choi, and Arkady Yerukhovich. “POPE: Partial Order Preserving Encoding”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. CCS '16. Association for Computing Machinery, 2016, pp. 1131–1142. DOI: [10.1145/2976749.2978345](https://doi.org/10.1145/2976749.2978345).
- [RKR21] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. “A Primer in BERTology: What We Know About How BERT Works”. In: *Transactions of the Association for Computational Linguistics* 8 (Jan. 2021), pp. 842–866. DOI: [10.1162/tac1\\_a\\_00349](https://doi.org/10.1162/tac1_a_00349).
- [Roy+20] Amrita Roy Chowdhury, Chenghong Wang, Xi He, Ashwin Machanavajjhala, and Somesh Jha. “Crypt $\epsilon$ : 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](https://doi.org/10.1145/3318464.3380596).
- [Sah+18] Cetin Sahin, Tristan Allard, Reza Akbarinia, Amr El Abbadi, and Esther Pacitti. “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](https://doi.org/10.1109/ICDE.2018.00082).

- [Sah+16] Cetin Sahin, Victor Zakhary, Amr El Abbadi, Huijia Lin, and Stefano Tessaro. “TaoStore: Overcoming Asynchronicity in Oblivious Data Storage”. In: *2016 IEEE Symposium on Security and Privacy (SP)*. 2016, pp. 198–217. DOI: [10.1109/SP.2016.20](https://doi.org/10.1109/SP.2016.20).
- [SJB14] Bharath Kumar Samanthula, Wei Jiang, and Elisa Bertino. “Privacy-preserving complex query evaluation over semantically secure encrypted data”. In: *European Symposium on Research in Computer Security*. Springer. 2014, pp. 400–418. DOI: [10.1007/978-3-319-11203-9\\_23](https://doi.org/10.1007/978-3-319-11203-9_23).
- [SGF18] Sajin Sasy, Sergey Gorbunov, and Christopher Fletcher. “ZeroTrace : Oblivious Memory Primitives from Intel SGX”. In: *Network and Distributed System Security (NDSS) Symposium* (Feb. 2018). DOI: [10.14722/ndss.2018.23243](https://doi.org/10.14722/ndss.2018.23243).
- [SKG20] Stephan van Schaik, Andrew Kwong, and Daniel Genkin. “SGAxe: How SGX Fails in Practice”. In: 2020. URL: [sgaxe.com/files/SGAxe.pdf](https://sgaxe.com/files/SGAxe.pdf).
- [SK96] Bruce Schneier and John Kelsey. “Unbalanced Feistel networks and block cipher design”. In: *Fast Software Encryption*. Springer Berlin Heidelberg, 1996, pp. 121–144. DOI: [10.1007/3-540-60865-6\\_49](https://doi.org/10.1007/3-540-60865-6_49).
- [SWG19] Michael Schwarz, Samuel Weiser, and Daniel Gruss. “Practical Enclave Malware with Intel SGX”. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer International Publishing, 2019, pp. 177–196. DOI: [10.1007/978-3-030-22038-9\\_9](https://doi.org/10.1007/978-3-030-22038-9_9).
- [Sch+17] Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. “Malware Guard Extension: Using SGX to Conceal Cache Attacks”. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer International Publishing, 2017, pp. 3–24. DOI: [10.1007/978-3-319-60876-1\\_1](https://doi.org/10.1007/978-3-319-60876-1_1).
- [SYSC21] Ning Shen, Jyh-Haw Yeh, Hung-Min Sun, and Chien-Ming Chen. “A Practical and Secure Stateless Order Preserving Encryption for Outsourced Databases”. In: *2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC)*. 2021, pp. 133–142. DOI: [10.1109/PRDC53464.2021.00025](https://doi.org/10.1109/PRDC53464.2021.00025).
- [SCSL11] Elaine Shi, T-H Hubert Chan, Emil Stefanov, and Mingfei Li. “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](https://doi.org/10.1007/978-3-642-25385-0_11).
- [SSSS17] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. “Membership Inference Attacks Against Machine Learning Models”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017, pp. 3–18. DOI: [10.1109/SP.2017.41](https://doi.org/10.1109/SP.2017.41).

- [Ska+19] Dimitrios Skarlatos, Mengjia Yan, Bhargava Gopireddy, Read Sprabery, Josep Torrellas, and Christopher W. Fletcher. “MicroScope: Enabling Microarchitectural Replay Attacks”. In: *Proceedings of the 46th International Symposium on Computer Architecture*. ISCA '19. Association for Computing Machinery, 2019, pp. 318–331. DOI: [10.1145/3307650.3322228](https://doi.org/10.1145/3307650.3322228).
- [SR20] Congzheng Song and Ananth Raghunathan. “Information Leakage in Embedding Models”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. Association for Computing Machinery, 2020, pp. 377–390. DOI: [10.1145/3372297.3417270](https://doi.org/10.1145/3372297.3417270).
- [ST15] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. 2015. DOI: [10.6028/NIST.FIPS.180-4](https://doi.org/10.6028/NIST.FIPS.180-4).
- [Ste+13] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. “Path ORAM: An Extremely Simple Oblivious RAM Protocol”. In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13. Association for Computing Machinery, 2013, pp. 299–310. DOI: [10.1145/2508859.2516660](https://doi.org/10.1145/2508859.2516660).
- [SSS12] Emil Stefanov, Elaine Shi, and Dawn Xiaodong Song. “Towards Practical Oblivious RAM”. In: *Network and Distributed System Security Symposium (NDSS)*. 2012.
- [SV11] Siddharth Suri and Sergei Vassilvitskii. “Counting triangles and the curse of the last reducer”. In: *Proceedings of the 20th international conference on World wide web*. 2011, pp. 607–614. DOI: [10.1145/1963405.1963491](https://doi.org/10.1145/1963405.1963491).
- [TYM14] Isamu Teranishi, Moti Yung, and Tal Malkin. “Order-Preserving Encryption Secure Beyond One-Wayness”. In: *Advances in Cryptology – ASIACRYPT 2014*. Springer Berlin Heidelberg, 2014, pp. 42–61. DOI: [10.1007/978-3-662-45608-8\\_3](https://doi.org/10.1007/978-3-662-45608-8_3).
- [Tra17] Transparent California. *California public pay and pension 2017 dataset*. <https://transparentcalifornia.com/salaries/2017/state-of-california/>. 2017.
- [Tra19] Transparent California. *California public pay and pension 2019 dataset*. <https://transparentcalifornia.com/salaries/2019/state-of-california/>. 2019.
- [Tur+18] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry McKay, Mary Baish, and Michael Boyle. *Recommendation for the Entropy Sources Used for Random Bit Generation*. 2018. DOI: [10.6028/NIST.SP.800-90B](https://doi.org/10.6028/NIST.SP.800-90B).

- [US 18] U.S. Census Bureau. *American Community Survey Public Use Microdata Sample*. <https://www.census.gov/programs-surveys/acs/microdata.html>. 2018.
- [Vai11] Vinod Vaikuntanathan. “Computing blindfolded: New developments in fully homomorphic encryption”. In: *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*. IEEE. 2011, pp. 5–16. DOI: [10.1109/FOCS.2011.98](https://doi.org/10.1109/FOCS.2011.98).
- [Van+18] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom, and Raoul Strackx. “Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution”. In: *Proceedings of the 27th USENIX Security Symposium*. USENIX Association, Aug. 2018. ISBN: 9781931971461.
- [Van+20] Jo Van Bulck, Daniel Moghimi, Michael Schwarz, Moritz Lippi, Marina Minkin, Daniel Genkin, Yuval Yarom, Berk Sunar, Daniel Gruss, and Frank Piessens. “LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection”. In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 54–72. DOI: [10.1109/SP40000.2020.00089](https://doi.org/10.1109/SP40000.2020.00089).
- [Vau02] Serge Vaudenay. “Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS ...”. In: *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology*. EUROCRYPT '02. Springer-Verlag, 2002. DOI: [10.1007/3-540-46035-7\\_35](https://doi.org/10.1007/3-540-46035-7_35).
- [VGG19] Dhinakaran Vinayagamurthy, Alexey Gribov, and Sergey Gorbunov. “StealthDB: a scalable encrypted database with full SQL query support”. In: *Proceedings on Privacy Enhancing Technologies 2019 (2019)*, pp. 370–388. DOI: [10.2478/popets-2019-0052](https://doi.org/10.2478/popets-2019-0052).
- [Wal77] Alastair J. Walker. “An Efficient Method for Generating Discrete Random Variables with General Distributions”. In: *ACM Transactions on Mathematical Software* 3.3 (Sept. 1977), pp. 253–256. DOI: [10.1145/355744.355749](https://doi.org/10.1145/355744.355749).
- [WHL20] Boyang Wang, Yantian Hou, and Ming Li. “QuickN: Practical and Secure Nearest Neighbor Search on Encrypted Large-Scale Data”. In: *IEEE Transactions on Cloud Computing* (2020). DOI: [10.1109/TCC.2020.3009961](https://doi.org/10.1109/TCC.2020.3009961).
- [WMK16] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. *EMP-toolkit: Efficient MultiParty computation toolkit*. <https://github.com/emp-toolkit>. 2016.

- [WZ18] Xingchen Wang and Yunlei Zhao. “Order-Revealing Encryption: File-Injection Attack and Forward Security”. In: *Computer Security*. Springer International Publishing, 2018, pp. 101–121. DOI: [10.1007/978-3-319-98989-1\\_6](https://doi.org/10.1007/978-3-319-98989-1_6).
- [Wel+18] Sean Welleck, Zixin Yao, Yu Gai, Jialin Mao, Zheng Zhang, and Kyunghyun Cho. “Loss Functions for Multiset Prediction”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Curran Associates Inc., 2018, pp. 5788–5797.
- [WL83] M. Anthony Wong and Tom Lane. “A  $k$ th Nearest Neighbour Clustering Procedure”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 45.3 (1983), pp. 362–368. ISSN: 00359246.
- [WCKM09] Wai Kit Wong, David Wai-lok Cheung, Ben Kao, and Nikos Mamoulis. “Secure KNN Computation on Encrypted Databases”. In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’09. Association for Computing Machinery, 2009, pp. 139–152. DOI: [10.1145/1559845.1559862](https://doi.org/10.1145/1559845.1559862).
- [Woz+13] Sander Wozniak, Michael Rossberg, Sascha Grau, Ali Alshawish, and Guenter Schaefer. “Beyond the Ideal Object: Towards Disclosure-Resilient Order-Preserving Encryption Schemes”. In: *Proceedings of the 2013 ACM Workshop on Cloud Computing Security Workshop*. CCSW ’13. Association for Computing Machinery, 2013, pp. 89–100. DOI: [10.1145/2517488.2517496](https://doi.org/10.1145/2517488.2517496).
- [XY12] Liangliang Xiao and I-Ling Yen. *A Note for the Ideal Order-Preserving Encryption Object and Generalized Order-Preserving Encryption*. Cryptology ePrint Archive, Report 2012/350. <https://ia.cr/2012/350>. 2012.
- [XYH12] Liangliang Xiao, I-Ling Yen, and Dung T. Huynh. *Extending Order Preserving Encryption for Multi-User Systems*. Cryptology ePrint Archive, Report 2012/192. <https://ia.cr/2012/192>. 2012.
- [XWG10] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. “Differential privacy via wavelet transforms”. In: *IEEE Transactions on knowledge and data engineering* 23.8 (2010), pp. 1200–1214. DOI: [10.1109/TKDE.2010.247](https://doi.org/10.1109/TKDE.2010.247).
- [Xie+16] Dong Xie, Guanru Li, Bin Yao, Xuan Wei, Xiaokui Xiao, Yunjun Gao, and Minyi Guo. “Practical private shortest path computation based on Oblivious Storage”. In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. 2016, pp. 361–372. DOI: [10.1109/ICDE.2016.7498254](https://doi.org/10.1109/ICDE.2016.7498254).



- [XPHF19] Min Xu, Antonis Papadimitriou, Andreas Haeberlen, and Ariel Joseph Feldman. “Hermetic: Privacy-preserving distributed analytics without (most) side channels”. In: 2019. URL: [haeberlen.cis.upenn.edu/papers/hermetic-tr.pdf](https://haeberlen.cis.upenn.edu/papers/hermetic-tr.pdf).
- [YK21] JiHye Yang and Kee Sung Kim. “Practical Frequency-Hiding Order-Preserving Encryption with Improved Update”. In: *Security and Communication Networks* 2021 (Dec. 2021). DOI: [10.1155/2021/1160305](https://doi.org/10.1155/2021/1160305).
- [YLX13] Bin Yao, Feifei Li, and Xiaokui Xiao. “Secure nearest neighbor revisited”. In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. 2013, pp. 733–744. DOI: [10.1109/ICDE.2013.6544870](https://doi.org/10.1109/ICDE.2013.6544870).
- [YMMS21] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, and Reza Shokri. *Enhanced Membership Inference Attacks against Machine Learning Models*. 2021. DOI: [10.48550/ARXIV.2111.09679](https://doi.org/10.48550/ARXIV.2111.09679).
- [YGFJ18] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh 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](https://doi.org/10.1109/CSF.2018.00027).
- [YPBV14] Xun Yi, Russell Paulet, Elisa Bertino, and Vijay Varadharajan. “Practical k nearest neighbor queries with location privacy”. In: *2014 IEEE 30th International Conference on Data Engineering*. 2014, pp. 640–651. DOI: [10.1109/ICDE.2014.6816688](https://doi.org/10.1109/ICDE.2014.6816688).
- [YPBV16] Xun Yi, Russell Paulet, Elisa Bertino, and Vijay Varadharajan. “Practical Approximate k Nearest Neighbor Queries with Location and Query Privacy”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.6 (2016), pp. 1546–1559. DOI: [10.1109/TKDE.2016.2520473](https://doi.org/10.1109/TKDE.2016.2520473).
- [Yu+21] Xixun Yu, Yidan Hu, Rui Zhang, Zheng Yan, and Yanchao Zhang. “Secure Outsourced Top-k Selection Queries against Untrusted Cloud Service Providers”. In: *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. 2021, pp. 1–10. DOI: [10.1109/IWQOS52092.2021.9521321](https://doi.org/10.1109/IWQOS52092.2021.9521321).
- [Zha+22] Yu Zhan, Danfeng Shen, Pu Duan, Benyu Zhang, Zhiyong Hong, and Baocang Wang. “MDOPE: Efficient multi-dimensional data order preserving encryption scheme”. In: *Information Sciences* 595 (2022). DOI: [10.1016/j.ins.2022.03.001](https://doi.org/10.1016/j.ins.2022.03.001).
- [ZRL21] Songnian Zhang, Suprio Ray, and Rongxing Lu. “SOREL: Efficient and Secure ORE-based Range Query over Outsourced Data”. In: *IEEE Transactions on Big Data* (2021), pp. 1–1. DOI: [10.1109/TBDATA.2021.3089986](https://doi.org/10.1109/TBDATA.2021.3089986).

- [Zhe+17] Wenting Zheng, Ankur Dave, Jethro G. Beekman, Raluca Ada Popa, Joseph E. Gonzalez, and Ion Stoica. “Opaque: An Oblivious and Encrypted Distributed Analytics Platform”. In: *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*. NSDI’17. USENIX Association, 2017, pp. 283–298.
- [Zhu+15] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. “Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Dec. 2015, pp. 19–27. DOI: [10.1109/ICCV.2015.11](https://doi.org/10.1109/ICCV.2015.11).
- [ZXT13] Youwen Zhu, Rui Xu, and Tsuyoshi Takagi. “Secure K-NN Computation on Encrypted Cloud Data without Sharing Key with Query Users”. In: *Proceedings of the 2013 International Workshop on Security in Cloud Computing*. Cloud Computing ’13. Association for Computing Machinery, 2013, pp. 55–60. DOI: [10.1145/2484402.2484415](https://doi.org/10.1145/2484402.2484415).

## Curriculum Vitæ

Dmytro **BOGATOV**

Software Developer at Amazon |


Ph.D. in Database Security from Boston University

dbogatov.org  github.com/dbogatov  +1 857 777 8350

 linkedin.com/in/dmytrobogatov  dmytro@dbogatov.org

 Google Scholar  Boston, MA  Proud Ukrainian 



I am originally from Ukraine. I have completed Bachelor's, Master's and Ph.D. in Computer Science. Working on Doctorate, I have defended the thesis "Secure and Efficient Query Processing in Outsourced Databases" [Bog22], advised by George Kollios, Professor at Boston University. Aside from work, I am serving as a Director and Secretary for **Mriya Inc.**  non-profit, managing shipments of medical supplies and equipment to Ukrainian military, territorial defense and civilians affected by the war.

### SKILLS

---

<b>Programming</b>	C#, TypeScript / JavaScript, Python, Swift, Java, C/C++, HTML, SQL, $\LaTeX$ , Go, Bash/ZSH, x86 ASM
<b>Frameworks</b>	.NET Core, ASP MVC, SGX, xUnit, Angular, boost::
<b>Common skills</b>	git, Gerrit, docker, bash, console, VS Code, unit testing
<b>DevOps</b>	Ubuntu / CentOS, Kubernetes, CI / CD, NGINX, GitLab admin
<b>Graduate classes</b>	Advanced Algorithms, Cryptography, Advanced Database Systems, Advanced Operating Systems, Distributed Systems, Compiler Design, Applied Cryptography, Optimization Methods

### EDUCATION

---

2013	Kyiv	High school (Lyceum 157) diploma. High distinction.	GPA 4.00.
2017	WPI	Bachelor of Science in Computer Science. High distinction.	GPA 3.99.
2019	BU	Master of Science in Computer Science with a specialization in Cyber Security.	GPA 4.00.
2022	BU	Doctor of Philosophy in Computer Science (Database Security).	

### PUBLICATIONS

[Bog22] **D. Bogatov.** "Secure and Efficient Query Processing in Outsourced Databases". PhD thesis. May 2022. doi: 10.48550/arXiv.2206.10753.

- [BKOZ22] **D. Bogatov**, G. Kollios, A. O’Neill, and H. Zamani. “*k-anon*: Secure Similarity Search in Outsourced Databases”. Apr. 2022.
- [BCET21] **D. Bogatov**, A. D. Caro, K. Elkhyaoui, and B. Tackmann. “Anonymous Transactions with Revocation and Auditing in Hyperledger Fabric”. In: *Cryptology and Network Security*. Springer International Publishing, 2021, pp. 435–459. doi: 10.1007/978-3-030-92548-2\_23.
- [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.
- [BKR19] **D. Bogatov**, G. Kollios, and L. Reyzin. “A Comparative Evaluation of Order-Revealing Encryption Schemes and Secure Range-Query Protocols”. In: *Proc. VLDB Endow.* 12.8 (2019), pp. 933–947. doi: 10.14778/3324301.3324309.
- [NBK19] O. Narykov, **D. Bogatov**, and D. Korokin. “DISPOT: A simple knowledge-based protein domain interaction statistical potential”. In: *Bioinformatics* (July 2019). doi: 10.1093/bioinformatics/btz587.
- [Bog17] **D. Bogatov**. “Analysis of a Dynamic Voluntary Contribution Mechanism Public Good Game”. In: *IPE Journal* 26 (2017).
- [BH16] **D. Bogatov** and J. R. Hennessy. “Data MATTERS: Customizing Economic Indices to Measure State Competitiveness”. In: *WPI Library* (2016).
- [NBXO15] B. Nurbekov, **D. Bogatov**, J. S. Xu, and R. J. O’Brien. “Investment Trading And Risk Management: Scientifically Developing and Analyzing Trading Systems”. In: *WPI Library* (2015).

## CONFERENCES (★ PRESENTED A POSTER, PRESENTATION OR A LIGHTNING TALK)

- |   |  |
|---|--|
| ★ Encrypted Search Workshop at Brown    | ➤ Second Workshop on Encryption for Secure Search, Bertinoro, Italy              |
| ➤ EUROCRYPT 2019, Darmstadt, Germany    | ★ 23 <sup>rd</sup> Annual Issues in Political Economy Conference, Washington, DC |
| ★ Northeastern Database Day 2019 at MIT | ★ 2016 IEEE MIT Undergraduate Research Technology Conference, Cambridge, MA      |
| ★ VLDB 2019, Los Angeles, CA            |  |
| ★ CANS 2021, Vienna, Austria (virtual)  |  |
| ★ ACM CCS 2021, Seoul, Korea (virtual)  |  |

## PAPER REVIEWS

2019	IEEE ICDE	2021	SIGMOD Repro	2022	SIGMOD Repro
2020	SIGMOD/PODS	2022	SIGMOD/PODS	2022	IEEE TDSC
2021	SIGMOD/PODS	2022	IEEE ICDE	2023	SIGMOD/PODS

## PROFESSIONAL EXPERIENCE (RECENT / RELEVANT)

---

Present	<b>Software Development Engineer II (Redshift team), <span style="color: #8B4513;">AMAZON WEB SERVICES</span>, Boston, MA</b>
May 2022	<ul style="list-style-type: none"> <li>&gt; Query processing optimization</li> <li>&gt; Security and Privacy at Redshift</li> </ul> <div style="display: flex; gap: 5px;"> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">research</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">databases</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">big data</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">performance</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">distributed systems</span> </div>
Sep 2021	<b>Software Developer Intern (Redshift team), <span style="color: #8B4513;">AMAZON WEB SERVICES</span>, Palo Alto, CA (virtual)</b>
Jun 2021	<ul style="list-style-type: none"> <li>&gt; Query processing optimization</li> <li>&gt; <span style="color: #8B4513;">Row Level Security (RLS)</span> </li> </ul> <div style="display: flex; gap: 5px;"> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">research</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">databases</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">big data</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">performance</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">distributed systems</span> </div>
Nov 2020	<b>Software Developer Intern (Redshift team), <span style="color: #8B4513;">AMAZON WEB SERVICES</span>, Palo Alto, CA (virtual)</b>
Aug 2020	<ul style="list-style-type: none"> <li>&gt; Query processing optimization</li> <li>&gt; <span style="color: #8B4513;">Row Level Security (RLS)</span> </li> </ul> <div style="display: flex; gap: 5px;"> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">research</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">databases</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">big data</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">performance</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">distributed systems</span> </div>
Dec 2019	<b>Applied Scientist Intern (Crypto Algorithms team), <span style="color: #8B4513;">AMAZON WEB SERVICES</span>, Seattle, WA</b>
Oct 2019	<ul style="list-style-type: none"> <li>&gt; Computations over encrypted data, SSE schemes</li> </ul> <div style="display: flex; gap: 5px;"> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">research</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">encrypted search</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">databases</span> </div>
Sep 2019	<b>Cloud &amp; Computing Infrastructure Intern (Crypto), <span style="color: #8B4513;">IBM RESEARCH</span> — ZÜRICH, Rüschlikon, ZH, Switzerland</b>
Jul 2019	<ul style="list-style-type: none"> <li>&gt; Anonymous Blockchain Transactions with Revocation and Auditing</li> <li>&gt; Developed production-grade cryptographic <span style="color: #8B4513;">Go library</span> </li> </ul> <div style="display: flex; gap: 5px;"> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">research</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">blockchain</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">hyperledger fabric</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">delegatable credentials</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">Go</span> </div>
Present	<b>All my past experiences, <span style="color: #8B4513;">ALL OVER THE WORLD</span>, USA, Ukraine, Switzerland</b>
Sep 2014	<ul style="list-style-type: none"> <li>&gt; Ph.D. / Research Assistant @ <b>Boston University</b> (<i>Boston, MA, USA</i>)</li> <li>&gt; Web / DevOps developer @ <b>RedwoodEDA</b> (<i>Shrewsbury, MA, USA</i>)</li> <li>&gt; Co-founder / web developer @ <b>Shevastream</b> (<i>Kyiv, Ukraine</i>)</li> <li>&gt; Web developer @ <b>Worcester Polytechnic Institute</b> (<i>Worcester, MA, USA</i>)</li> <li>&gt; Web developer @ <b>TradeStation Securities</b> (<i>Plantation, FL, USA</i>)</li> <li>&gt; Senior Teaching Assistant @ <b>CS Department at WPI</b> (<i>Worcester, MA, USA</i>)</li> </ul> <div style="display: flex; gap: 5px;"> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">teaching</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">web development</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">startup</span> <span style="border: 1px solid #ccc; border-radius: 3px; padding: 2px 5px;">devops</span> </div>

## PROJECTS (RECENT / RELEVANT)

### EPSOLUTE: DIFFERENTIALLY PRIVATE SECURE RANGE QUERIES

SEP 2018 — OCT 2021

[github.com/epsolute](https://github.com/epsolute)  [publication](#) 

Developed ORAM-based differential privacy range and point query engine

C++ ORAM differential privacy range queries

### ANONYMOUS TRANSACTIONS IN BLOCKCHAINS

JUN 2019 — DEC 2021

[github.com/IBM/dac-lib](https://github.com/IBM/dac-lib)  [publication](#) 

Developed production-grade Go cryptographic library

Go blockchains research cryptography privacy

### EVALUATION OF SECURE RANGE QUERY PROTOCOLS

MAY 2017 — FEB 2018

[ore.dbogotov.org](https://ore.dbogotov.org)  [publication](#) 

Analyzed, implemented and evaluated 5 ORE schemes and 5 protocols

C# .NET Core research ORE range query protocols

### STATUS SITE

FEB 2017 — SEP 2017

[status.dbogotov.org](https://status.dbogotov.org)  [source code](#) 

A multi-component system for collecting and analyzing the infrastructure health

C# .NET Core ASP MVC TypeScript Kubernetes micro-services

## LANGUAGES

English ● ● ● ● ●  
 Ukrainian ● ● ● ● ●  
 Russian ● ● ● ● ●

## SELECT AWARDS

★ pVLDB 2021 Reproducibility Award at BU  
 > Chair's Fellowship at BU  
 > "Outstanding Junior" at WPI

## REFEREES

[George Kollios](#) 

Professor, BU

@ gkollios@bu.edu

 [Scholar Profile](#) 
[Leonid Reyzin](#) 

Professor, BU

@ reyzin@bu.edu

 [Scholar Profile](#) 
[Manos Athanassoulis](#) 

Assistant Professor, BU

@ mathan@bu.edu

 [Scholar Profile](#) 