Abstract of "Information Leakage in Encrypted Systems Through an Algorithmic Lens"
by Evgenios Kornaropoulos, Ph.D., Brown University, May 2019.

As the volume and complexity of generated data grow, users would like to maintain the ability to issue expressive queries on their data without sacrificing privacy. Encrypted databases are one of the most promising approaches towards this direction. However, this efficiency comes with the price of leaking information about the plaintext data. In this thesis, we use an algorithmic approach to develop rigorous attacks on encrypted databases and secure protocols. In the first part of this thesis, we consider secure protocols that allow two parties to jointly compute the similarity between their respective high-dimensional data without revealing the data to each other. Typical secure similarity approximation protocols first execute a sketching algorithm offline where the necessary randomness for the computation is "leaked" by design to all the participants. We show how a participant can generate an adversarial input so as to heavily mis-approximate the similarity and harm the correctness of the computation. In the second part of the thesis, we study a scenario where a client outsources to a server an encrypted database indexed by a one-dimensional attribute and issues a sequence of k-nearest neighbor queries. An adversary (e.g., the server or man-in-the-middle) observes which encrypted records are retrieved in response to each query and attempts to reconstruct the plaintext values of the indexed attribute. This so-called access pattern is "leaked" by design so as to achieve efficient performance. We prove that for ordered k-NN responses, the adversary can approximate plaintext attribute values with considerable accuracy. For unordered k-NN responses, we characterize the set of all valid reconstructions and present an attack that reconstructs the plaintext attribute values with tight approximation guarantees. State-of-the-art attacks on access pattern leakage operate under the assumption that the queries are generated uniformly at random. The last part of this thesis shows how to overcome this unrealistic assumption by using tools from learning theory, statistics, and optimization. Namely, we present the first leakage-based attacks for both range and k-NN queries that make no assumptions about the data or the query distribution. We introduce a new set of distribution-agnostic techniques which, as we demonstrate, achieve accurate reconstruction under a broad class of query distributions.

Information Leakage in Encrypted Systems Through an Algorithmic Lens

by

Evgenios Kornaropoulos

B.Sc., Computer Science, University of Crete, Greece, 2009

M.Sc., Computer Science, University of Crete, Greece, 2012

Sc.M., Computer Science, Brown University, 2014

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2019

This dissertation by Evgenios Kornaropoulos is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

Roberto Tamassia, Director

Recommended to the Graduate Council

Date _____ _____

Vasileios Kemerlis, Reader
(Brown University)

Date _____ _____

Thomas Ristenpart, Reader
(Cornell University)

Approved by the Graduate Council

Date _____ _____

Andrew G. Campbell
Dean of the Graduate School

Acknowledgements

I am proud to call Roberto Tamassia my academic father. Roberto gave me the space, the time, the resources, and most importantly the guidance to sharpen my technical skills as well as develop my research taste. His mentorship, his interdisciplinary approach, and his kindness were a source of inspiration during my time at Brown. I feel very lucky that Roberto's ethics and technical rigorousness shaped me professionally during these formative years.

I am grateful to my thesis committee members Vasileios Kemerlis and Thomas Ristenpart. I feel fortunate that my time at Brown overlapped with Vasilis' beginning as a professor in our department. Vasilis was both a good friend that would diffuse stress by sharing his experience as well as a tireless colleague that leads by example and wants to help everybody around him excel. I thoroughly enjoyed my discussions with Tom during the last year. His influential research agenda inspired my way of approaching research. Tom's work strikes the perfect balance between practicality, rigorousness, and "coolness" that I poise to have in my own research.

I am very thankful for my friendship with Alexandra Papoutsaki, Babis Papamanthou, and Foteini Baldimtsi. Their professional and personal achievements and support inspired me to do my best, be happy, and be proud of my work. Babis is my academic big brother and our frequent research meetings resulted in the Chapter 3 and the Chapter 4 of this dissertation. He put the bar high and was always available when I wanted to share my enthusiasm about a newly discovered insight or my frustration about a dead-end.

I would also like to thank the professors that I interacted with and learned from while at Brown: Anna Lysyanskaya, Eli Upfal, Franco Preparata, John Savage, Sorin Istrail, Seny Kamara, Shriram Krishnamurthi, Ugur Cetintemel. A big thanks goes to the staff that accepted my usually late requests with a big smile, thanks Lauren, Gene, and Dawn. I am grateful to my first mentors Ioannis G. Tollis and Panagiotis Tsakalides from University of Crete. I did my first steps in research under

Yannis' supervision during my graduate studies in Greece and I am thankful for the time and energy he put into our work.

My great co-authors Petros Efstathopoulos, Michael Mitzenmacher, Michael Goodrich, and Matteo Riondato, shared their ideas and thoughts that led to a series of interesting papers. I had a great time interning at Symantec Research Labs with Petros where I worked on a project that resulted in the Chapter 2 of this dissertation. A thank you goes to James Kelley and Peter Shah for the great internship experience at NetApp Advanced Technology Group.

A heartfelt thank you to all the friends I made during my time at Brown you all helped me in more ways than I can count: Apoorva, Cam, Chris, DK, Emanuel, Esha, Jasper, Maria, Megumi, Nedi, Nickolai, Odysseas, Paris, Socrates, Tarik, Yorgos. My friends from FRCF Alecia, Adam, Hampus, Anastasia, Jordan and the rest of the group that helped me decompress my everyday stress by breaking a sweat as well as my friend Paul from PVDonuts that negated all my effort at the gym with their donuts. A big thank you goes to my Brattleboro family, Paul and Nan, who visited us and hosted us often. My secret weapon is the support and the happiness that Kirtley brings to my life. I feel very special to have her by my side and I can't wait to see what life brings us.

My family from Greece is my anchor. I found time and time again that their company recharges my batteries like nothing else in this world. Thank you Sotia, Freek, Achilleas, Eugenio, Elena, Poppy, and the rest of my cousins, uncles, and aunts that if I were to list you all I would probably need 2-3 pages more! I can't thank enough my father Manos Kornaropoulos who raised me and allowed me to mature and become the person I am today. This dissertation is dedicated to him and in memory of my mother Asteropi.

I would also like to thank the U.S. National Science Foundation, the Kanellakis Fellowship from Brown University, the Gerondelis scholarship, as well as the NetApp Faculty award that sponsored my research during my years in Providence. I am humbled and thankful that the Graduate School of Brown University chose this dissertation as the recipient of the Joukowsky Family Foundation Outstanding Dissertation Award in the Physical Sciences. It is a true honor to receive such a recognition and I am sharing this prize with my colleagues, my mentors, and my support team who were with me in every step of this journey.

Dedicated to my father Manos

and

in memory of my mother Asteropi

Contents

List of Tables **ix**

List of Figures **x**

1 Introduction **1**

1.1 Analysis of Secure Sketching Protocols 2

1.2 Analysis of k-NN Queries in Encrypted Databases 5

1.3 Analysis of k-NN & Range Queries in Encrypted Databases Under Minimal Assumptions 6

1.4 Contributions of This Thesis 11

2 Adversarial Inputs for Secure Similarity Approximation Protocols **15**

2.1 Preliminaries 16

2.1.1 Secure Sketching 16

2.1.2 Similarity Approximation 17

2.1.3 Semi-Homomorphic Cryptosystems 19

2.2 Threat Model 19

2.3 Perturbation Attack 21

2.3.1 Attacking Minhash Sketches 23

2.3.2 Attacking Cosine Sketches 30

2.4 Server-Aided Approximation 33

2.4.1 Building Blocks 36

2.4.2 Protocols for the Server-Aided Model 44

2.5 Scalability Evaluation 51

2.6 Related Work 53

3 Leakage of k-NN Queries in Encrypted Databases 55

3.1 Preliminaries 56

3.2 Exact Reconstruction 60

3.2.1 Reconstructing the Order of Records 61

3.2.2 Overview of the Attack For Ordered Responses 63

3.2.3 Ordering Voronoi Segments and Computing Bisectors 64

3.2.4 From Exact Bisectors to Full Reconstruction 66

3.2.5 Exact Reconstruction Impossibility for Unordered Responses 69

3.3 Approximate Reconstruction 69

3.3.1 Ordered Responses: Estimating Voronoi Segment Lengths 70

3.3.2 Unordered Responses: Defining the Reconstruction Space 73

3.3.3 Overview of the Unordered Response Attack 78

3.3.4 Unordered Responses: Reconstruction for $k \geq 2$ 81

3.4 Evaluation of Approximate Reconstruction 98

3.4.1 Evaluation of Unordered Response Attack 99

3.4.2 Evaluation of Ordered Response Attack 102

4 Attacks on Encrypted Databases Beyond the Uniform Query Distribution 103

4.1 Preliminaries 103

4.2 How to Exploit Search-Pattern Leakage 105

4.2.1 Conditional Probability Distributions over the Leakage 105

4.2.2 Estimate Support Size of Each Distribution 107

4.3 Revisiting Data Reconstruction Attacks 116

4.3.1 Reconstruction from Range Queries 116

4.3.2 Reconstruction from k-NN Queries 124

4.4 Related Work 133

⋆ Parts of this dissertation appeared in proceedings of conferences. In particular, Chapter 2 is an extended version of [103], Chapter 3 is an extended version of [104], and Chapter 4 is in submission.

List of Tables

1.1 Assumptions of state-of-the-art value reconstruction attacks and our new attacks . . 8

2.1 Evaluation of the perturbation attack on minhash sketches over synthetic data. The term κ denotes the size of the sketch, s is the size of the set under attack, f_{Success} is the frequency of success of the probabilistic Algorithm 1. The data points shown are the average over 5,000 instantiations. Time is measured in seconds. 22

2.2 Evaluation of the perturbation attack on cosine sketches over synthetic data. The data points show the average value over 10 instantiations. 32

2.3 An overview of the protocols. For brevity we assume that the public keys of the server $PK_P^{(S)}, PK_{GM}^{(S)}$ and the client $PK_P^{(C)}, PK_{GM}^{(C)}, PK_{DGK}^{(C)}$ are publicly available and thus not passed as an input to the protocols. 35

2.4 Communication Overhead of Sketching. Average over 5 runs. 53

3.1 Evaluation of AttackUnordered on the SpitzLoc dataset 100

3.2 Evaluation of AttackOrdered on the SpitzLoc dataset 102

List of Figures

1.1 Illustration of the perturbation attack on an e-discovery data. The adversary can add the 5 red-colored words in the original email with id 549 and the approximate distance of our instantiation will be 1 even though the exact distance is 0.004. 4

1.2 On the left there is a black-and-white picture of the Trojan horse. In the middle there are $n = 1840$ sampled two-dimensional values from the original picture projected to a Hilbert curve of order 7 so as to reduce the k-NN queries to one dimension. On the right side we demonstrate the reconstruction of the plaintext values *solely based on the query leakage* under the studied assumptions. The depicted setup has relative error 0.01% both in 1D and 2D, and $k = 9$. 7

1.3 Visual comparison between plaintext values of real-world private geolocation dataset Spitz (in red) and values reconstructed by our attack Agnostic-Reconstruction-KNN on k-NN queries under a *Gaussian distribution* and $k = 10$ (in black). Our attack achieves an approxiamate reconstruction (1) under a non-uniform query distribution and (2) with *half the queries* and larger k values compared to previous work [104]. . 8

1.4 (a) Heatmap of the Span distribution of range queries on values from 0 to 100, where the probability of query $[a, b]$ is proportional to $(N-b+a)^{25}$. (b) Reference probabilities of plaintext values under the uniform query distribution (blue histogram) and Span query distribution (red histogram). Reconstructing the values of an encrypted database, shown with solid bars, from their empirical reference probabilities, is easy under the uniform query distribution but hard under the Span query distribution. 10

2.1 An illustration of the objective function of the maximization problem of Algorithm 2 where $n = 2$ and $\vec{v} = (20, 10)$. The X-,Y-axis denote the x_1 and x_2 dimension of vector to be added, $\vec{x}$. 31

2.2 Illustration of the perturbation attack on a gene-expression of an adenoma patient. The proposed attack on vector $\vec{v}$ outputs the perturbed $\vec{v^+}$ with approximate cosine distance 1 even though the exact distance is 0.0036. 33

2.3 Two-party protocols between a client and the server that are used as building blocks for sketching. 36

2.4 Protocol EncComparison is a slight modification of the comparison protocol found in [24, 153]. Protocol EncComparison2 is the same up to step **(9)** where it terminates by outputting $|t|$ to the Server. 37

2.5 The sketching protocols between the server and the client for the server-aided model. 44

2.6 The reconstruction of SketchingCosine between the server and the clients. The reconstruction for SketchingOdd is the same for steps **(1)**-**(6)**; steps **(7)**, **(8)** follow the reconstruction of Equation (2.3). 48

2.7 Subfigure (a): Time performance for varied set size of the SketchingOdd protocol. Time averaged for a single minhash over five runs. Subfigure (b): Time performance for varied number of vector dimensions of the SketchingCosine protocol. Time averaged for a single random projection over five runs. 52

2.8 Throughput of the server Reconstruct protocol for varied sketch sizes. Average values over 5 runs. 53

3.1 The partition of $[\alpha, \beta]$ in Voronoi segments of ordered and unordered responses for $k = 2$ (left) and $k = 3$ (right). The curly brackets on the bottom indicate the *unordered responses* that correspond to each Voronoi segments. The vertically written k-tuples indicate the *ordered responses*. The term $b_{i,j}$ denotes the bisector between v_i and v_j which is also the Voronoi endpoint that separates the corresponding neighboring Voronoi segments. 56

3.2 An overview of the attack based on *ordered responses* where $k = 2$. 63

3.3 Values v_0 and v_2 must be equidistant, specifically ξ_0 afar, from $b_{0,2}$. Using the derived equations we can express the locations of v_2, v_4, v_6, v_8 because the locations of bisectors $b_{0,2}, b_{2,4}, b_{4,6}, b_{6,8}$ stay fixed. Similarly, by using offset ξ_1 we can express the locations of v_3, v_5, v_7, v_9. 75

3.4 The possible values of ξ_0, ξ_1 in X- and Y-axis respectively for a given $[v]$. The feasible region is colored with blue. The set of redundant ordering constraints are depicted with gray dotted-lines, and the non-redundant ordering constraints with black bold-lines. The boundary constraints are depicted with red dotted-lines. 77
3.5 An overview of the attack based on *unordered responses* for $k = 2$. 80
3.6 The addition of a single new value to the left of the group G does not change the location of the values and their bisectors. The only thing that changes is the labeling, i.e., sub-index, of the values and bisectors and is indicated in red. 96
3.7 Three date ranges for the month October of the publicly available mobile records with the geolocation of the German Green party politician Malte Spitz. on the first row we demonstrate the original dataset and in the bottom the accuracy of the reconstruction that we achieve for unordered responses for $k = 2$. 100
3.8 Distribution of length terms per ordering constraint for different values of k. The error compounds as the number of length terms per constraint grows. 101

4.1 To observe response $r = \{id_1, id_2, id_3\}$ the start of the query range must be in-between v_0 and v_1 and the end must be in-between v_3 and v_4. Thus, the total number of queries that return r is $(v_1 - v_0) \cdot (v_4 - v_3)$. 106
4.2 Voronoi diagram of a database with 6 values $v_0, \ldots, v_5$ and 2-NN queries. Short vertical black lines indicate distinct queries and tall vertical green lines indicate bisectors $b_{i,i+2}$ for values v_i and v_{i+2}. 107
4.3 An illustration of three query distributions with the same histogram. The result of a support size estimation is the *same* in all three cases. 108
4.4 Comparison of estimators JACKKNIFE-SELFTUNE and VALIANT-VALIANT with respect to their relative error in support size estimation. 110
4.5 Evaluation of the estimators is conducted under various query distributions parameterized as a Beta probability mass function. 113
4.6 Comparison of estimators under uniform query distribution. 114
4.7 Illustrative example of a database along with all the possible conditional probability distributions and their corresponding support size. 116

4.8 Comparison between GeneralizedKKNO and our attack, Agnostic-Reconstruction-Range, under the uniform query distribution. 121
4.9 Performance of our attack, Agnostic-Reconstruction-Range, under parameterizations of query distributiona Short-Ranges and Value-Centered. 122
4.10 Real-world dataset Spitz of a privacy-sensitive geolocation trace: (a) data for Otober 1-31, 2009; (b) mapping of the points to a Hilbert curve which reduces the 2D data to 1D; (c) query distribution under attack, which consists of a permutation of a discretized Beta(α, β) distribution; (d) another query distribution under attack, which is a Gaussian centered at the city of Hannover, Germany. 131
4.11 Absolute relative error of Agnostic-Reconstruction-KNN for varying query distributions on the Spitz dataset. 132
4.12 Performance of Agnostic-Reconstruction-KNN for varying query distributions on synthetic data. 133

Chapter 1

Introduction

Thesis statement: The information leaked by systems that handle expressive queries on encrypted data can be an avenue for sophisticated attacks. We posit that by utilizing algorithmic techniques we can analyze the security blindspots of these privacy-preserving techniques and quantitatively asses their limitations under minimal assumptions.

In this thesis, we use an algorithmic approach to develop and analyze rigorous attacks on encrypted databases and secure protocols. A better understanding of the landscape of attacks will not only *warn* the community about the unknown vulnerabilities of proposed schemes but also *inform* the cryptographers about what to avoid and how to mitigate the discovered shortcomings.

In the first part of this thesis, we study secure protocols that allow two parties to jointly compute the similarity between their respective high-dimensional data without revealing the data to each other. In the second part of the thesis, we study a scenario where a client outsources to a server an encrypted database indexed by a one-dimensional attribute and issues a sequence of k-nearest neighbor queries. The last part of this thesis shows how to overcome unrealistic assumption such as uniform query distribution by using tools from learning theory, statistics, and optimization. Namely, we present the first leakage-based attacks for both range and k-NN queries that make no assumptions about the data or the query distribution. We introduce a new set of distribution-agnostic techniques which, as we demonstrate, achieve accurate reconstruction under a broad class of query distributions.

1.1 Analysis of Secure Sketching Protocols

Quantifying similarity between high-dimensional data points is a cornerstone problem in the area of data mining. The history of the problem goes back to 1901 with the influential work of Jaccard [94] and has a wide range of applications in today's software systems and services especially in the areas of healthcare [143], law [81], finance [80], recommendation engines [84], personalization systems [50], social networks [158], databases [159], earth science [126], link prediction [114], forensics [136]. The wide adoption of this concept in diverse fields highlights the importance of similarity computation—the spectrum of application is so broad that instead of listing them we refer the reader to books [109, 116, 142] describing some of the applications of similarity computation. Some of the applications above consider similarity computation between high-dimensional data in the presence of *strict privacy requirements*. As motivating examples, we consider two such areas: *electronic discovery* and *healthcare*.

Electronic discovery (or *e-discovery*) typically focuses on the discovery and identification of information among privacy-sensitive electronic files as part of a lawsuit or formal investigation. It has been reported that the area of legal forensic discovery is a $9.9 billion market [51]. The community of technologists and legal experts in the area has formed the Electronic Discovery Reference Model (EDRM) [5], which is a framework that describes standards for the recovery and discovery of digital information during the legal process—e.g., criminal evidence discovery. According to the EDRM paradigm, during the phase of "Preparation" the discovery model filters documents so as to shortlist the ones most interesting/relevant among a voluminous collection of data. Section 1.2 of EDRM's directive [6] explicitly lists *"Similarity Hashing"* as a recommended action to shortlist privacy-sensitive documents. Thus, similarity approximation is a vital component of this multi-billion dollar business.

The area of *patient similarity* has attracted attention from both industry [102] as well as the medical community [27, 79, 143]. The emerging area of personalized medicine, where patient similarity plays a central role, aims at treatments tailored to individual characteristics of each patient. To achieve this goal, one needs to organize similar patients into subgroups that have the same response to a given treatment. This approach has dramatically changed the area of pharmacogenetics [162]. From a computational perspective, entire research teams (e.g., [102]) are focusing on the problem of *patient similarity*, applying advanced algorithmic techniques so as to discover groups of patients with similar health record profiles, while aiming to provide high secrecy for the sensitive healthcare records. Health information exchange protocols are already in place [8], allowing patient similarity

computation across hospitals of different US states. These are two of the many important examples highlighting not only the central role of similarity detection in important business areas, but also the need for performing such detection using secure and robust methods, due to the sensitivity of the analyzed data.

Secure Sketching. As computing exact similarity metrics on very large datasets is prohibitively expensive, state-of-the-art methods seek to *approximate* the similarity function that needs to be computed, by working with a succinct representation of the data that is called a *sketch*. Sketching is the mainstream approach for efficiently approximating a plethora of functions and applications [22, 25, 26, 40, 44, 58, 71, 88, 89, 91, 113, 119, 122, 141, 146]. The seminal work by Feigenbaum *et al.* [64] set the foundation for secure multiparty computation of approximation functions. Furthermore, the community has made several important steps towards private computation on genomic data in a time-efficient and scalable manner [14, 20, 43, 49, 129]. Wang *et al.* [155] demonstrate the potential of secure approximations, by running a privacy-preserving similarity query for a human genome on 1 million records distributed across the U.S., in a couple of minutes. All of the above works only consider an honest-but-curious adversary. In this thesis we extend the threat model and demonstrate how easy it is to craft adversarial inputs for sketching algorithms within this new model.

On Crafting Adversarial Inputs. The sketching protocol as presented by Feigenbaum *et al.* [64] has two phases: 1) the sketching function is applied locally by each party, and 2) the reconstruction function is performed via secure multiparty computation. Our offline attack is mounted on the first phase by a data owner who exploits the fact that i) the *randomness* of the sketching algorithm is known to all the participants, and ii) the sketching algorithm is performed *locally*. Such an adversary can steer any similarity approximation between the perturbed data and any other data point to an *incorrect output*, regardless of the secure computation protocols of the second phase. Our first attack uses simple probabilistic arguments, and is mounted on the *minhash sketching*, which is deployed to measure the Jaccard similarity between two sets. Our second attack formulates a high-dimensional constrained optimization problem, and is mounted on the *cosine sketching*, which is deployed to measure the cosine similarity between two vectors.

Threat Model. In this threat model the *only action* the attacker is allowed to take is to change the *input data* to the sketching algorithm. This is because any other alteration that concerns, a) the steps of the locally computed sketching algorithm, b) the sketch computed, and c) the secure protocols, can be easily detected by applying verifiable computation mechanisms [69, 137]. We focus

our attention to the threat related to the input data of the sketching algorithm, and leave as an open problem the task of deploying efficiently verifiable computation for the remaining steps (i.e., potential attacks on items a, b, and c above). The *input to the sketching algorithm* is the very first step of the pipeline and is provided directly by the user, therefore the protocols have no means of verifying if it is a legitimate input or an adversarially perturbed input. This new threat model formally capture the attack surface of malicious perturbations of the input data with the end-goal of *violating the correctness of the similarity approximation.*

Motives for Mis-approximating Similarity. The motivation for such attacks can be clearly demonstrated by considering the previously discussed examples of e-discovery and patient similarity (among others). In the case of e-discovery the plaintiff party is interested in correctly approximating similarity between privacy-sensitive documents so as to discover important evidence. On the contrary, the defending party might prefer to masquerade evidence by causing mis-approximation. Applying a perturbation attack on document similarity approximation algorithms will conceal important documents from the shortlisted set that will be thoroughly investigated on a criminal evidence discovery case, such an outcome violates the directive of EDRM [6]. An illustration of the power of our proposed attacks is illustrated in Figure 1.1. In the case of patient similarity, a perturbation attack will cause a pair of patients that have similar medical profiles to be assigned to different subgroups. For instance, all future patients that are assigned to a subgroup with perturbed data will receive personalized treatment that is not effective or, even worse, lethal. Such an attacker not only causes a disrupted service on the patient similarity component of a personalized treatment engine, but also introduces liability for the participating parties.

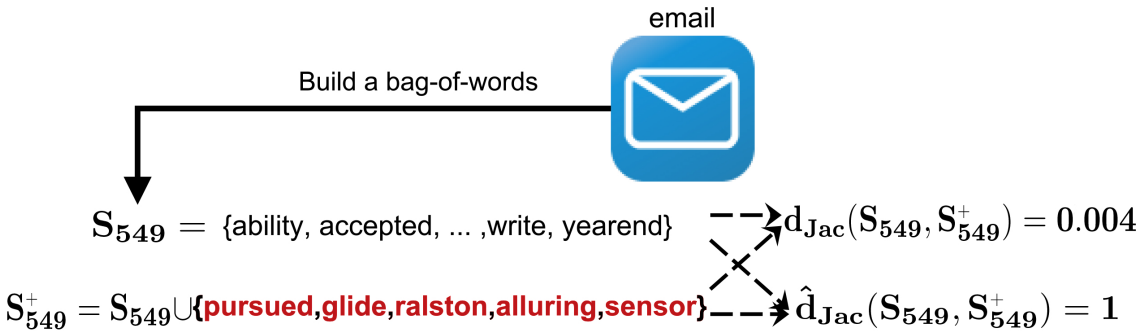


Figure 1.1: Illustration of the perturbation attack on an e-discovery data. The adversary can add the 5 red-colored words in the original email with id 549 and the approximate distance of our instantiation will be 1 even though the exact distance is 0.004.

Proposed Mitigation. To mitigate perturbation attacks we follow the standard server-aided paradigm [98] and formulate a *server-aided secure approximation architecture* that requires the participation of three parties, as opposed to two of the previous schemes. A new honest-but-curious entity—the server—stores the common randomness which is treated as private information. A user, who no longer knows the common randomness, is therefore forced to run a protocol with the server to build an encrypted sketch, as opposed to the local computation of the previous model's first phase. During the sketching protocol the user doesn't learn any information about the common randomness and the server doesn't learn any information about the user's data. The sketch-generation takes place *only once for each data point*, and the sketch can be reused for any future pairwise approximation. Most importantly, under this new server-aided framework the users *do not have direct access* to the common random input and thus they can not mount an offline perturbation attack. In this thesis, we devise and implement new secure protocols in order to generate minhash and cosine sketches in our proposed architecture. Given a pair of sketches[1] our implementation achieves throughput of 30-600 approximations per second for data points with hundreds of dimensions.

1.2 Analysis of k-NN Queries in Encrypted Databases

Systems for *Searchable Encryption* (SE) [23, 36, 45, 53, 67, 96, 147, 149] allow a client to outsource an encrypted database to a server who can subsequently answer certain types of queries by operating *solely on the encrypted data*. In order to meet real-world efficiency demands, SE constructions allow, by definition, some well-defined *leakage* of information.

In the case of encrypted single-keyword search [23, 36, 45, 96, 149], this leakage reveals which file identifiers match the encrypted queried keyword—also known as *access pattern leakage*. The impact of this type of leakage had not been clear for a long time and it was only until recently that the community started to study its implications. In particular, the works of Islam *et al.* [93], Cash *et al.* [37], and recently Zhang *et al.* [163], demonstrate how an attacker can utilize access patterns to launch *query-recovery* attacks under various assumptions.

However, in the case of richer queries (e.g., range [63, 90, 138] and SQL [135, 139]), more severe *data-recovery* attacks are possible due to the expressiveness of the query. In particular, the work by Kellaris, Kollios, Nissim, and O'Neill [99] attacks SE-type systems that support range queries

[1] The parameterization, and consequently the efficiency, of the sketching instantiation depends on the approximation guarantees.

(e.g., [63, 85, 110]) by observing record identifiers whose plaintext values belong to the queried range. Similarly, a recent work by Lacharité, Minaud, and Paterson [106] further explores range query leakage to achieve exact and approximate reconstruction for the case of dense datasets with *orders of magnitude fewer queries* (when compared to [99]). Finally, order-preserving encryption based systems (e.g., CryptDB [139]) supporting even more expressive queries (such as SQL) have been shown to be vulnerable to data-recovery attacks [59, 82, 128] even without observing *any queries*, just by the setup leakage.

In this thesis, we explore the implications of another generic query leakage profile, that of k-nearest neighbor (k-NN) queries, which return the k nearest points of a database to a given query point with respect to a distance metric. A *spatial database* is engineered to model, store, and query data defined in a geometric space. There is a plethora of systems and products (e.g., Geomesa [7], PostGIS for PostgreSQL [3], and IBM's Cloudant NoSQL DB Geospatial [1]) that provide scalable solutions for handling spatial data. *Proximity queries* such as k-NN, appear in all of the above systems.

Support for *k-NN queries on encrypted databases* has drawn a lot of attention in the database community for more than a decade [61, 70, 101, 112, 124, 133, 154, 156, 160]. Several of the above designs, e.g. [101, 154], reveal as query leakage the k encrypted records returned to the client as response to a k-NN query. In this thesis, we analyze what a passive adversary can achieve by *only observing the set of encrypted records returned by a sequence of k-NN queries.* Our leakage-abuse attacks achieve significant accuracy of data recovery for one-dimensional k-NN queries. Also, as higher-dimensional data can be efficiently queried by mapping it to one-dimensional values (e.g., via Hilbert curves) [108, 130, 145, 161], our approach is applicable to a wider family of constructions. Our findings suggest a reevaluation of what is considered secure in the area of k-NN queries for encrypted databases.

1.3 Analysis of k-NN & Range Queries in Encrypted Databases Under Minimal Assumptions

In *searchable encryption* [45, 96, 147], a client encrypts a privacy-sensitive data collection and outsources an encrypted database to a server that can efficiently answer search queries without ever decrypting the database. Known constructions handle rich and expressive queries [53, 63, 95] under the definitional framework of *structured encryption* (STE) [41].

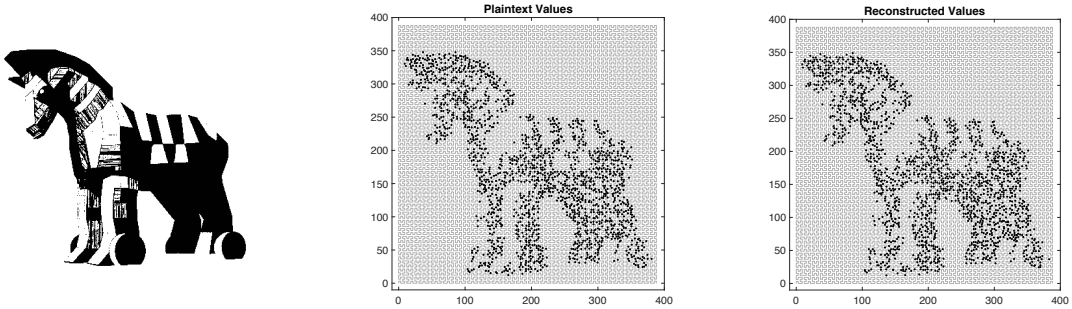


Figure 1.2: On the left there is a black-and-white picture of the Trojan horse. In the middle there are $n = 1840$ sampled two-dimensional values from the original picture projected to a Hilbert curve of order 7 so as to reduce the k-NN queries to one dimension. On the right side we demonstrate the reconstruction of the plaintext values *solely based on the query leakage* under the studied assumptions. The depicted setup has relative error 0.01% both in 1D and 2D, and $k = 9$.

To strike a balance between efficiency and privacy, structured encryption schemes reveal, by design, certain information about the query and its corresponding response—this is the so-called *leakage*. Despite cryptographic proofs guaranteeing that *nothing more is leaked* but what the designer allowed, the implications of the legitimately leaked information have not been fully grasped yet. The first generation of leakage-based attacks [37, 93, 163] focused on *query reconstruction* under various assumptions. The next generation of attacks [83, 99, 104, 106] supported *plaintext value reconstruction* by a server answering expressive queries on a one-dimensional database. In chapter 4, we present the first efficient reconstruction methods from range and k-NN queries that do not rely on any assumption about the query distribution or the underlying data.

We overview the limitations of four state-of-the-art attacks supported by a detailed theoretical analysis and experimental evaluation [83, 99, 104, 106] and outline our new approach.

Uniform Query Distribution Assumption. The first value reconstruction attack for range queries was proposed by Kellaris-Kollios-Nissim-O'Neil (KKNO) [99]. It assumes that queries are issued *uniformly at random*. Lacharité-Minaud-Paterson (LMP) [106] studied the same problem for the special case of *dense* databases—this is a simpler problem since reconstructing order is equivalent to reconstructing values. The work by Grubbs-Lacharité-Minaud-Paterson (GLMP) [83] gives three main reconstruction attacks for range queries under different assumptions: attacks Generalized-KKNO and ApproxValue assume an underlying *uniform query distribution*, extend the underlying ideas of KKNO, and present a new analysis on the query complexity; attack AOR-to-ADR does not assume uniform queries but assumes that the attacker knows *both the query distribution and an*

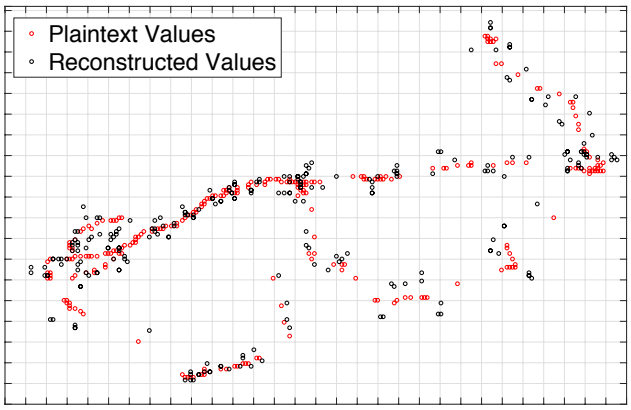


Figure 1.3: Visual comparison between plaintext values of real-world private geolocation dataset Spitz (in red) and values reconstructed by our attack AGNOSTIC-RECONSTRUCTION-KNN on k-NN queries under a *Gaussian distribution* and $k = 10$ (in black). Our attack achieves an approxiamate reconstruction (1) under a non-uniform query distribution and (2) with *half the queries* and larger k values compared to previous work [104].

approximation of the data distribution. Kornaropoulos-Papamanthou-Tamassia (KPT) [104] propose reconstruction attacks for k-nearest neighbor queries under the *uniform query distribution.* The above attacks, summarized in Table 1.1, set the foundations for understanding the implications of leakage but only succeed under *strong assumptions* that potentially do not hold in the real world, e.g., uniform query distribution. Thus, the following question still remains open:

"Is it possible to devise attacks that reconstruct an approximation of the plaintext values without any assumptions about the query distribution or the data distribution?"

Our results answers this question in the affirmative and presents reconstruction techniques that are *query and data distribution agnostic.* The key to achieve such a generalization lies in the *search pattern leakage* which is revealed in all known STE schemes but has been overlooked so far. See Figure 1.3 for an illustration of a reconstruction achieved by our attacks.

Table 1.1: Assumptions of state-of-the-art value reconstruction attacks and our new attacks

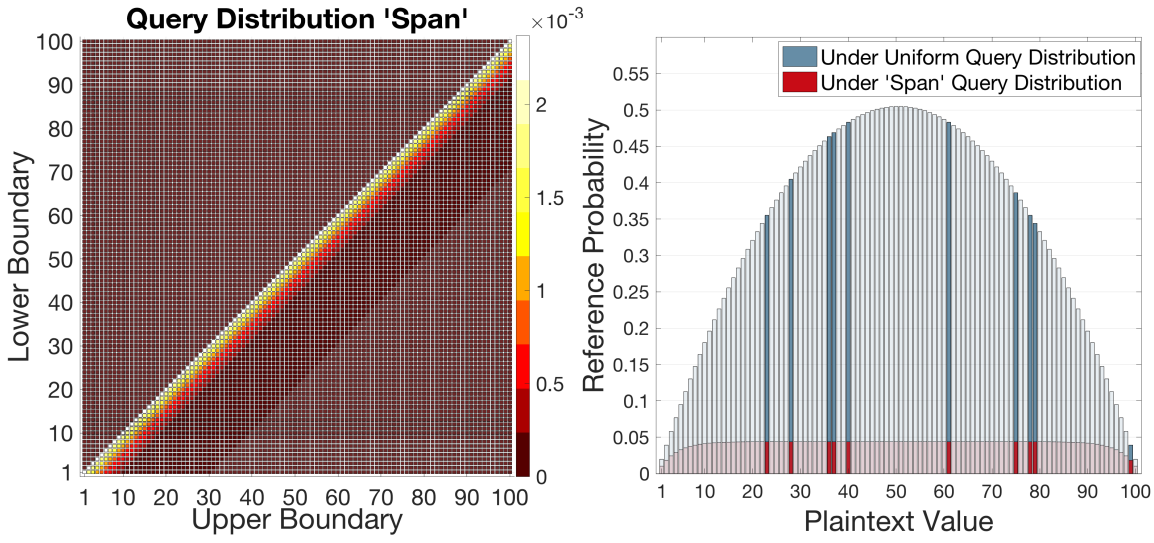
Value Reconstruction Attack Algorithms	**Query Type**	**Assumptions**				
		Query Distribution	Data Values in a Fixed Region	Known Data Distribution	Known Query Distribution	Dense Database
Chapter 3 - KPT [104]	k-NN	Uniform	-	-	-	-
KKNO [99]	Range	Uniform	-	-	-	-
GLMP [83] GENERALIZEDKKNO	Range	Uniform	-	-	-	-
GLMP [83] APPROXVALUE	Range	Uniform	●	-	-	-
GLMP [83] AOR to ADR	Range	**Unknown**	-	●	●	-
LMP [106]	Range	**Unknown**	-	-	-	●
Chapter 4	k-NN & Range	**Unknown**	-	-	-	-

Fundamental Limitations of Current Range Attacks. A natural approach for answering the above question would be to extend existing algorithmic techniques to work for arbitrary query distributions. To explore this possibility, we first give a high-level intuition of the range reconstruction attacks KKNO, GeneralizedKKNO, and ApproxValue. Through the *access pattern leakage*, which appears in the vast majority of STE schemes, the attacker can see *which* and *how many* queries return a given encrypted record. Assume the attacker knows the space of possible plaintext values, e.g., values from 0 to 100 representing attribute age. If range queries are *generated uniformly*, the attacker expects values in the middle (e.g., age $= 50$) to be returned more often than values towards the ends (e.g., age $= 1$). Formally, the *reference probability* of a value v captures the likelihood that value v will be returned in a response to a query. It is defined as $\sum_{r \in \mathcal{R}_v} \Pr[r]$, where $\mathcal{R}_v$ is the set of ranges containing v and $\Pr[r]$ is the probability of querying range r. Reference probabilities can be easily pre-computed by an attacker who knows the query distribution.

The reference probability of plaintext values for two query distributions is shown with histograms in in Figure 1.4(b). After observing enough queries, the attacker computes the frequency of each encrypted value in the responses and then tries to find a close match of with a pre-computed reference probability, taking the corresponding plaintext value as a reconstruction. This technique works well for the uniform query case as the reference probabilities (blue histogram) vary significantly over the universe of plaintext values. However, *there are fundamental limitations when trying to exend this approach.*

For instance consider the Span range query distribution, inspired by a realistic behavior from a client, depicted as a heatmap in Figure 1.4(a), where the lower boundary of the range is on the Y-axis, the upper boundary is on the X-axis and the color of each square denotes the probability of issuing this query. One can visually confirm that queries around the diagonal, i.e., queries with short span, have brighter color, hence are more likely to be issued. The reference probability for the Span query distribution is shown with the red histogram in Figure 1.4(b). Note that the reference probabilities of 60% of potential plaintext values differ by less than 10^{-8}. Thus, the Span query distribution causes all state-of-the-art attacks to fail as the adversary cannot distinguish most encrypted values from their observed frequency.

More generally, one can define query distributions where the reference probabilities are *identical* so no matter how many queries are observed, the adversary cannot distinguish between potential



(a) (b)

Figure 1.4: (a) Heatmap of the Span distribution of range queries on values from 0 to 100, where the probability of query $[a, b]$ is proportional to $(N - b + a)^{25}$. (b) Reference probabilities of plaintext values under the uniform query distribution (blue histogram) and Span query distribution (red histogram). Reconstructing the values of an encrypted database, shown with solid bars, from their empirical reference probabilities, is easy under the uniform query distribution but hard under the Span query distribution.

plaintext reconstructions in the information-theoretic sense. Interestingly, the fact that frequency-based attacks fail in "smooth" distributions is used as a form of mitigation by Lacharité-Paterson [105], who study introducing multiplicities in the records and "spreading" the frequency of among the copies. From the above example we see that for range queries we need a *radically different reconstruction approach* to generalize.

How Many Queries Return a Response? Taking a step back to rethink reconstruction attacks, there is a piece of information that has not been fully exploited to overcome the uniform query assumption. This is the *number of queries that a return a given response*, r, among the possible range queries. Let x_r be the number of lower query boundaries that can potentially return response r, and let y_r be the number of upper boundaries that can potentially return the same response, r. The total number of queries that return response r is essentially $N_r = x_r \cdot y_r$, but more importantly, both x_r and y_r are *distances between consecutive encrypted values*. Therefore if we knew N_r for all r we could set up a system of equations containing x_r and y_r and retrieve the distances between all values in the database, effectively computing the values themselves.

However, the exact values of N_r are not available. Our main approach lies in *estimating* N_r *using*

search pattern leakage (which is part of all known constructions) and then setting up *carefully-crafted optimization problems* to retrieve an estimation of the underlying distances/values.

Harnessing Search-Pattern Leakage. The *search pattern leakage* reveals to the adversary if two encrypted queries, called *search tokens*, are generated from the same query. Interestingly none of the aforementioned state-of-the-art attacks [83, 99, 104, 106] utilize the search-pattern leakage, considering it harmless. We argue that this leakage can be instead exploited. Suppose that 10^3 observed search tokens (not-necessarily distinct) return response r. If these 10^3 tokens are the same, we can make a probabilistic argument that there aren't that many queries that return r. On the contrary, if all 10^3 are distinct, then there are clearly at least 10^3 queries, and likely more, that return r. More formally, the problem of estimating the number of *unseen outcomes from the frequency of observed outcomes* is called *support size estimation* and it has a rich history [28, 68, 152]. We use non-parametric support size estimation techniques that make no assumptions about the underlying distribution to re-think reconstruction algorithms for encrypted databases. Our techniques *reconstruct very accurately even in the case of "smooth" frequencies of retrieval* due to the fact that our attacks are based on *the number of possible queries that return a response*, a quantity that can be efficiently estimated under flat frequencies, as we demonstrate in our experiments.

1.4 Contributions of This Thesis

- **Chapter 2:** We identify and formalize the notion of *perturbation attacks* against secure multiparty approximation. We propose two attacks, the first is on the *minhash sketching* that is used to approximate the similarity between two sets, and the second is on the *cosine sketching* that is used to approximate the similarity between two vectors. We apply the proposed attacks on both real and synthetic data. Following the paradigm of server-aided design, we propose a *server-aided* approach that mitigates offline perturbation attacks. In our setup, a server has exclusive access to the common randomness, and is assisting the clients in the sketch computation. Thus, a user does not learn any information about the common random input. Additionally, the server doesn't learn any information about the user's data[2].

- **Chapter 3:** We study what a *passive* and *persistent* adversary can achieve by observing the query leakage that only reveals which k encrypted records are retrieved for a private k-NN query

[2]Other than the result of the approximation of unknown inputs.

on a database with n one-dimensional values. We study the case of **unordered responses** where the adversary observes the set of k retrieved records as well as the case of **ordered responses** where the adversary observes the k-tuple of retrieved records ordered in ascending order with respect to the private query point. We assume that the private query points are generated uniformly at random. Our **exact reconstruction results** are:

Ordered Responses. We show that an adversary with auxiliary information can achieve exact reconstruction in time $O(k\,n\log n)$. This auxiliary information is rather unrealistic, e.g., the lengths of the Voronoi diagram which is a conceptual partition of the space based on DB, but our goal is to study the *feasibility* of exact reconstruction. *Unordered Responses.* We prove that even a computationally unbounded adversary can not achieve exact reconstruction for this generic k-NN leakage. Our impossibility proof shows that there exist an infinite number of DB reconstructions that the observed query leakage can potentially come from, thus it is infeasible for an adversary to deterministically output client's DB.

Even though from the adversarial point of view the above results do not look promising (i.e., unrealistic auxiliary information and impossibility), there is still hope. For our main results we show the following **approximate reconstruction results**:

Ordered Responses. We show an attack where the adversary has *no access to auxiliary information* but still approximately reconstructs with failure probability δ the plaintext values with relative error ϵ_R in time $O(k\,n\log n + \frac{1}{\epsilon_R^2}(k^2\,n + \log\frac{1}{\delta}))$. In the heart of this technique is an estimator that *approximates* the previously-handed auxiliary information. The recovered values are at most $\pm\epsilon_R$ afar from the client's DB values with probability at least $1-\delta$, where ϵ_R, δ are tunable.

Unordered Responses. In the main result of our analysis we study the geometric structure of infinite reconstructions, what we call *feasible region*. Armed with insights about the geometry of this feasible region, we present a novel approximation approach that outputs a reconstructed DB with a upper-bounded worst-case reconstruction accuracy. Interestingly, the bound is a function of a characteristic quantity of the feasible region, what we call *diameter of the feasible region*, and in the evaluation section we examine the interplay between the diameter and the accuracy of the reconstruction.

- **Chapter 4:** We first describe how the adversary can achieve knowledge transfer from statistics

and learning theory to reconstructing encrypted databases. By partitioning the multiset of observed token-response pairs (t, r), the adversary can study each partition separately and draw inferences about the number of possible tokens that return r. We benchmark the state-of-the-art non-parametric support size estimation techniques under various (unknown to the adversary) query distributions. Our experiments indicate that certain estimators are better under different query distributions so we propose a new modular approach to pick the best estimation for the sample in hand. We further derive analytical expressions for known high-order non-parametric estimators, which is of independent interest.

Armed with techniques for estimating the number of queries that return a response, we develop a new machinery to approximately reconstruct an encrypted database. On a high-level, each estimation gives us information about two distances between encrypted values. But these estimations are made independently and with a different sample sizes. We propose an efficient new algorithm, Agnostic-Reconstruction-Range, that is based on an unconstrained convex optimization problem so as to piece together the above independent estimations and output *estimated distances between consecutive values of the database*. Our modeling gives higher weight to estimations made after observing a larger number of queries. We test our attack under a variety of query distributions and database densities, and show it achieves reconstructions with good accuracy. Also, Agnostic-Reconstruction-Range outperforms GeneralizedKKNO for the majority of tested setups under the uniform query distribution, which is noteworthy because our algorithm is unaware of how the queries are issued and GeneralizedKKNO is fine-tuned for the uniform case.

For the problem of reconstruction from k-NN queries, we plug our support size estimators into the KPT algorithm to derive an estimation of the *length of the Voronoi segments* without relying on the uniform query distribution. Even though in theory this direct application is valid, due to the fact that for arbitrary query distributions the estimations are less accurate than in the uniform case, our initial experiments demonstrated that more often than not the resulting collection of estimated lengths is *not a Voronoi diagram* and thus KPT returns no reconstruction. To remedy this problem, we propose a new and efficient approach via forming a constrained convex optimization problem that discovers the *minimum distortion* of the estimated lengths so as to force the lengths to *become* a valid Voronoi diagram. The formulation

of KPT appears as a set of constraints in this new algorithm. Due to the minimum distortion insight, our proposed Agnostic-Reconstruction-KNN always outputs a reconstruction as opposed to the all-or-nothing approach of KPT. Furthermore, since we don't explicitly build the set of all possible solutions, our approach scales to larger k compared with KPT. An illustration of a reconstruction for a real-world dataset of privacy-sensitive geolocation is shown in Figure 1.3 . This reconstruction is achieved with *half the queries* compared to KPT, under a *Gaussian query distribution*, and with one-dimensional relative error of 0.08%.

Chapter 2

Adversarial Inputs for Secure Similarity Approximation Protocols

In this chapter we introduce and study the effectiveness of adversarial inputs for secure similarity approximation protocols. We propose concrete perturbation attacks for the well-studied minhash and cosine sketching techniques, and measure the performance and scalability of the attacks on both real and synthetic data, while tuning various parameters. Subsequently, we formally define a server-aided model that mitigates the aforementioned attacks. We also propose new sketching protocols for this architecture, building upon state-of-the-art sketching techniques. Our design and implementation aims at speeding up the reconstruction protocols, as they constitute the part of the overall computation that is executed most frequently—thus having the most severe impact on overall performance. We evaluate the implementation of the proposed protocols and demonstrate that this architecture achieves the desired scalability for the reconstruction process, with reasonable performance.

2.1 Preliminaries

k-Independent Hashing. Space and time-efficient hash functions provide rigorous guarantees about the distribution of their values, such a family is the family of *k-independent* hash functions. Let U be the domain of the inputs to the hash function and let $x \in U$ be a specific input. Let $p > |U|$ be a prime and $a_0, a_1, \ldots, a_{k-1} \in \mathbb{Z}_p$ be uniformly chosen values over the prime field $\mathbb{Z}_p$. A commonly used construction of a k-independent family is based on polynomials of degree $k-1$:

$$h(x) = (\alpha_{k-1}x^{k-1} + \ldots + \alpha_1 x + \alpha_0) \mod p.$$

2.1.1 Secure Sketching

Exact similarity computation between two data points takes at least linear time with respect to the size of the data, since we need to parse the data item for *every comparison* regardless of the similarity function. A way to overcome this overhead is to settle with an *approximation* of similarity.

Definition 1. *(Def. 10.1 in [121]) A randomized algorithm gives an* (ϵ, δ)-approximation *for the value* ν *if the output* ν' *of the algorithm satisfies,* $\Pr(|\nu' - \nu| \leq \epsilon\nu) \geq 1 - \delta$.

We are interested in *sketching techniques* that are well-studied and widely applied in the area of data-mining and information retrieval [25, 26, 40, 44, 58, 89, 113, 122, 141, 146]. A benefit of sketching is that the succinct summary of the data, i.e., *the sketch*, is built once and can be reused in future pairwise approximations. Thus the super-linear overhead occurs only during the construction of the sketch which significantly speeds up the total time performance over a series of similarity approximations. The notion of a sketching protocol is defined as:

Definition 2. *(Def. 8 in [64]) A* sketching protocol *for a 2-argument function* $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ *is:*

- *A* sketching function*,* $\mathcal{S} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ *mapping one input and a random string to a* sketch *consisting of a (typically) short string.*
- *A (deterministic)* reconstruction function $\mathcal{G} : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$*, mapping a pair of sketches to an approximate output.*

On inputs $\alpha, \beta \in \{0,1\}^n$*, the protocol proceeds as follows. First, Alice and Bob locally compute a sketch* $\sigma_A = \mathcal{S}(\alpha, r_{cmn})$ *and* $\sigma_B = \mathcal{S}(\beta, r_{cmn})$ *respectively, where* r_{cmn} *is a common random input.*

Then, the parties exchange sketches, and both output locally $\hat{f} = \mathcal{G}(\sigma_A, \sigma_B)$. *We denote by* $\hat{f}(\alpha, \beta)$ *the randomized function defined as the output of the protocol on inputs* α, β. *A sketching protocol as above is said to* (ϵ, δ)*-approximate* f*, if* $\hat{f}$ (ϵ, δ)*-approximates* f.

We note that in this chapter we are interested in normalized similarity therefore the output of the sketching protocol takes values in $[0, 1]$. Following the terminology of Goldreich for multiparty computation (Section 7.2 [73]) we capture the above process with the following *functionality*:

$$\mathcal{F}_{\text{Approx}}\big((\alpha, r_{cmn}), (\beta, r_{cmn})\big) \rightarrow (\hat{f}(\alpha, \beta), \hat{f}(\alpha, \beta)), \tag{2.1}$$

where the first (resp. second) pair is the input of client C_A (resp. client C_B) and the output to both parties is the (ϵ, δ)-approximation $\hat{f}(\alpha, \beta)$. We note here that if the clients execute the sketching computation with *different randomness* then the output of the reconstruction is meaningless[1], thus the randomness must be the same. We emphasize that α, β are user-provided inputs and their legitimacy relies on the honesty and intention of the user.

A metric space is a set X accompanied with a *distance function* $d : X \times X \rightarrow \mathbb{R}$, or simply *distance*, that measures the distance between points $x, y \in X$. We are interested in the approximation of distance functions from which we can derive the similarity. Given the similarity we can compute the corresponding distance, and vice versa, thus the two terms are used interchangeably in the rest of the chapter.

2.1.2 Similarity Approximation

Approximating Jaccard Similarity. The *Jaccard similarity coefficient* (or Jaccard index) measures the similarity between two sets. Formally, given sets S_1, S_2 the Jaccard similarity coefficient and the Jaccard distance d_{Jac} are defined as:

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}, d_{Jac}(S_1, S_2) = 1 - J(S_1, S_2).$$

Minwise hashing [25, 26], or *minhashing*, is a technique for approximating the Jaccard index that has been successfully applied to numerous problems (e.g., [26, 113, 122, 141, 151]). Even though the analysis of the approximation is based on random permutations [25], in practice we use minhash

[1]This is equivalent to using different hash functions for the approximation of Jaccard similarity, or using different random vectors for the approximation of the cosine similarity.

functions that are defined as $h_i^{min}(S) = min_{x \in S}(h_i(x))$, where h_i is a k-independent hash function. Using κ distinct minhash functions one can build a *minhash sketch*, also called minhash signature, $\sigma(S)$ for input set S. Given two minhash sketches we approximate the Jaccard distance $\hat{d}_{Jacc}$ as follows:

$$\begin{aligned} \hat{d}_{Jacc}(S_1, S_2) &= \frac{1}{\kappa} d_H(\sigma(S_1), \sigma(S_2)), \\ \sigma(S) &= (h_1^{min}(S), \ldots, h_\kappa^{min}(S)), \end{aligned} \tag{2.2}$$

where d_H denotes the hamming distance between the two input arguments. The common random input r_{cmn} from Definition 2 is used to initialize the minhash functions.

Mitzenmacher *et al.* [122] introduced an approximation technique using odd sketches. An *odd sketch* of set S, denoted as $odd(S)$, consists of 1) a bit array T of size u and 2) a hash function $h_{odd} : U \to [0, u-1]$. In order to approximate the Jaccard similarity via odd sketches one uses the values of the minhash sketch $\sigma(S) = (x_1,\ldots,x_\kappa)$ as the input set for the odd sketch. Whenever an item $x_i = h_i^{min}(S)$, where $i \in [1, \kappa]$, is hashed to the odd sketch T using function h_{odd}, the bit in position $h_{odd}(x_i)$ of T is flipped. We approximate the Jaccard index as follows [122]:

$$\hat{J}_{odd}(S_1, S_2) = 1 + \frac{u}{4\kappa} \ln\Big(1 - \frac{2|odd(\sigma(S_1))\Delta odd(\sigma(S_2))|}{u}\Big), \tag{2.3}$$

where $|odd(\sigma(S_1))\Delta odd(\sigma(S_2))|$ denotes the number of 1s in the sketch resulted after the exclusive-or operation over the odd sketches, κ denotes the number of independent minhash values, and u denotes the size of the odd sketch. Jaccard distance is approximated using eq. (2.3), as $\hat{d}_{Jacc}(S_1, S_2) = 1 - \hat{J}_{odd}(S_1, S_2)$. The common random input r_{cmn} is used to initialize h_{odd} and $h_1^{min}, \ldots, h_\kappa^{min}$. Thus all the parties of the sketching protocol (see Definition 2) generate the same hash functions.

Approximating Cosine Similarity. The work of Charikar [40] introduced the notion of *cosine sketching* commonly used [58] to estimate the similarity between two vectors. Formally, let $\vec{v_1}, \vec{v_2} \in \mathbb{R}^n$ the *cosine similarity* as

$$C(\vec{v_1}, \vec{v_2}) = \frac{\vec{v_1} \cdot \vec{v_2}}{\|\vec{v_1}\|_2 \|\vec{v_2}\|_2}, d_{cos}(\vec{v_1}, \vec{v_2}) = (1 - C(\vec{v_1}, \vec{v_2}))/2, \tag{2.4}$$

where $\|\cdot\|_2$ is the Euclidean norm of the vector. The resulting similarity $C(\vec{v_1}, \vec{v_2})$ ranges from -1 to 1 which is interpreted as completely opposite and as exactly the same, respectively. The cosine

sketching technique is based on sign random projections. Let $\vec{v} \in \mathbb{R}^n$ be a unit vector[2], then the cosine sketch is a κ-dimensional bit vector $\sigma(\vec{v}) = (\sigma_1, \ldots, \sigma_\kappa)$. The components σ_i for $i \in [1, \kappa]$ and the symmetric cosine sketch distance [115] are defined as:

$$\sigma_i = \begin{cases} 1, & \text{if } \vec{w_i}^T \cdot \vec{v} \geq 0 \\ 0, & \text{if } \vec{w_i}^T \cdot \vec{v} < 0, \end{cases}, \hat{d}_{cos}(\vec{v_1}, \vec{v_2}) = \frac{d_H(\sigma(\vec{v_1}), \sigma(\vec{v_2}))}{\kappa}, \tag{2.5}$$

where $\vec{w_i} \in \mathbb{R}^n$ is sampled uniformly at random from the set of n-dimensional unit vectors. The common random input r_{cmn} is used to initialize the vectors $\vec{w_i}$, for $i \in [1, \kappa]$.

2.1.3 Semi-Homomorphic Cryptosystems

We use the described notation to highlight that messages are encrypted under different cryptosystems.

Paillier Cryptosystem. The Paillier cryptosystem [132] is semantically secure. The term $[m]$ denotes the encryption of message m under the key pair $K_P = (PK_P, SK_P)$; from the additive homomorphism we have that $[m_1] \cdot [m_2] = [m_1 + m_2]$.

Goldwasser-Micali Cryptosystem. The Goldwasser - Micali (GM) cryptosystem [77] is semantically secure. The term $|m|$ denotes the encryption of the bit m under the key pair $K_{GM} = (PK_{GM}, SK_{GM})$; from the homomorphism we have that $|m_1| \cdot |m_2| = |m_1 \oplus m_2|$, where $\oplus$ is the XOR operation.

Damgård-Geisler-Krøigaard Cryptosystem. The Damgård-Geisler-Krøigaard (DGK) cryptosystem [47, 48] is semantically secure. The DGK cryptosystem is considered to be much more efficient [21, 62, 107] than Paillier due to its small plaintext space. The term $\langle m \rangle$ denotes the encryption of message $m \in \mathbb{Z}_u$ under the key pair $K_{DGK} = (PK_{DGK}, SK_{DGK})$. DGK is additively homomorphic; moreover, it embeds reductions modulo u to its homomorphic operations, therefore $\langle m_1 \rangle \cdot \langle m_2 \rangle = \langle (m_1 + m_2) \mod u \rangle$.

2.2 Threat Model

In this chapter we consider a new threat model where the adversary can maliciously perturb *only* her input to the sketching algorithm which is executed offline and locally, a behavior that is challenging to

[2]In case the input vector is not unit we convert it by normalizing.

detect. We form this new threat model so as to formally capture and study an *algorithmic blindspot* that permits the proposed family of attacks. At a high level, this new adversary does not interfere with the computation of the sketching, the reconstruction, and the communication, i.e., adversary follows the prescribed protocols *after* the perturbation of the input data. Thus, our threat model is *not* the honest-but-curious. Our adversary is *not* trying to learn the input of the other party, the goal is to make his/her own data look different from what it really is with respect to the approximation. Consider the following class of protocols that compute the functionality $\mathcal{F}_{\text{Approx}}$ from the previous Section.

Class of Protocols for $\mathcal{F}_{\text{Approx}}$

- **Step 1:** Generate and distribute the common random input r_{cmn} to all the parties.
- **Step 2:** Each party inputs her data and r_{cmn} so as to locally compute the sketching function $\mathcal{S}$.
- **Step 3:** Parties run an MPC protocol that outputs the result of the reconstruction function $\mathcal{G}$.

Attack Surface of $\mathcal{F}_{\text{Approx}}$. We assume that the adversary participates in the above protocol. We distinguish two possible *offline* attacks on this class of protocols, the attacker can: 1) deviate from the correct execution of the locally computed sketching, and/or 2) execute the sketching correctly, but corrupt its output—and therefore the input to the reconstruction function $\mathcal{G}$. Both attacks can be detected using verifiable computation [69, 137], i.e., provide proof of correctness for the computation and the output of $\mathcal{S}$. Addressing such mitigations is outside the scope of our work and is left as future work. We focus on the remaining attack surface: since cryptographic techniques exist to detect the above attacks, the last resort for the adversary is to perturb the input to the sketching function.

Perturbing the Input to $\mathcal{S}$. To capture the remaining attack surface, in the new threat model we extend the above class of protocols by allowing the adversary to locally execute a function right before Step 2. Specifically, the adversary executes a randomized function Perturb that takes as an input the data point α and the common random input r_{cmn} outputted by Step 1. Function Perturb runs locally, without any interaction, and outputs a value α^{+} that will serve as the new input to the sketching function $\mathcal{S}$. We emphasize that after the execution of Perturb the adversary *behaves in a semi-honest fashion*, i.e., she honestly follows the sketching function and honestly executes the

MPC protocol. Thus, in our threat model the only malicious activity of the adversary is the local execution of Perturb.

2.3 Perturbation Attack

In this Section we define *perturbation attacks* on the class of protocols defined in Section 2.2. A successful attack on secure sketching protocols for a *distance function* yields a perturbed input such that although the pair (original input, perturbed input) is close with respect to the corresponding distance function, the approximation instantiation appears vastly distant. Thus, if one compares the sketch of *any* data point that is close to the original input, to the sketch of the perturbed input the distance is heavily mis-approximated.

To the best of our knowledge our work is the first that *concretely demonstrates* the pitfalls of using common random input r_{cmn} for secure sketching protocols. In this analysis we focus on distance functions, analogous definitions can be formed for other functions. Note that Definition 2 deals with two inputs α and β from distinct users, whereas the following definition deals with the input of a single user and it perturbed version, i.e., α and α^+.

Definition 3. *Let $\mathcal{F}_{Approx}$ be the functionality described in Equation (2.1) for a sketching approach of a distance function d. Let $\mathsf{Perturb}(\cdot)$ be the function that adversary $\mathcal{A}$ can apply according to the threat model of Section 2.2. Let $\alpha \in X$ be a point of the metric space (X, d) with distance function d. Let r_{cmn} be the common random input to the sketching function $\mathcal{S}$. Then we say that $\mathsf{Perturb}(\cdot)$ is a* successful (ν, ν')-perturbation attack *for sketching function $\mathcal{S}$ if for any α and r_{cmn}, $\mathsf{Perturb}(\alpha, r_{cmn})$ outputs a point α^+ such that:*

1. *The true distance between α, α^+ is ν, $d(\alpha, \alpha^+) = \nu$,*
2. *The approximate distance between α, α^+ is ν' according to $(\mathcal{S}, \mathcal{G})$ with input r_{cmn}, $\hat{d}_{(\mathcal{S},\mathcal{G})}(\alpha, \alpha^+) = \nu'$,*
3. *The inequality $|\nu' - \nu| > \epsilon\nu$ holds.*

where ϵ is the parameter of the (ϵ, δ) approximation guarantees of $\hat{d}_{(\mathcal{S},\mathcal{G})}$.

One might suggest that it is trivial to mount a successful perturbation attack by generating random data and call it α^+. This naive approach would successfully increase the approximate distance

ν' (condition 2), but it would *heavily distort* the original input and as a result the true distance ν would increase as well, i.e., doesn't satisfy the inequality of condition 3. Intuitively, for the case where $\nu' > (1+\epsilon)\nu$, condition 3 guarantees that the perturbed data "appears" more distant from the original than it truly is even when we consider the valid approximation error ϵ. For the case where $\nu' < (1-\epsilon)\nu$, condition 3 guarantees that the perturbed data "appears" more similar from the original than it truly is. In our analysis we focus on the case $\nu' > (1+\epsilon)\nu$, thus the adversary wants to hide the high similarity by minimally perturbing the input. Due to the triangle inequality, if α appears distant to α^+ w.r.t. the approximation, then any data point β that is close to α will also appear distant to α^+ w.r.t. the approximation. We leave as an open problem the case where the adversary perturbs the data so as to make highly dissimilar items look similar.

	$d_{Jac} \geq 0.9$									$d_{Jac} = 1$								
	$s = 500$			$s = 1,000$			$s = 10,000$			$s = 500$			$s = 1,000$			$s = 10,000$		
κ	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time	d_{Jac}	f_{Success}	Time
10	0.01	1.00	0.01	0.008	1.00	0.03	0.0008	1.00	0.37	0.01	0.98	0.10	0.009	0.99	0.22	0.0009	0.99	2.52
50	0.08	1.00	0.07	0.043	1.00	0.15	0.004	1.00	1.47	0.09	0.95	1.32	0.047	0.96	3.04	0.005	0.98	38.9
100	0.15	1.00	0.14	0.082	1.00	0.30	0.008	1.00	3.00	0.16	0.89	4.07	0.090	0.92	9.23	0.009	0.96	120.1
200	0.27	1.00	0.34	0.159	1.00	0.59	0.018	1.00	5.25	0.28	0.82	12.60	0.166	0.86	27.6	0.019	0.94	380.1

Table 2.1: Evaluation of the perturbation attack on minhash sketches over synthetic data. The term κ denotes the size of the sketch, s is the size of the set under attack, f_{Success} is the frequency of success of the probabilistic Algorithm 1. The data points shown are the average over 5,000 instantiations. Time is measured in seconds.

On using Commitment Schemes. It appears that the perturbation attack can be avoided if we deploy commitment schemes [72] for the data *before* they receive r_{cmn}. Thus, any perturbation will be caught due to the binding property of the construction. This mitigation indeed works only if *all data* from *all the users* is available during the initialization of the system and no sketch is created thereafter. In all practical scenarios, however, the system is more dynamic—users generate additional data and join/leave at arbitrary times. If a user creates new data *after* the commitment phase then this new input can be perturbed since the randomness value is already known and the new data is not committed. One might argue that we can redistribute new randomness to all the clients periodically. This will defend against these attacks but it implies that every party must re-compute the sketches from scratch whenever new randomness is issued, which would go against the very reason we used sketching techniques in the first place—to avoid processing the high-dimensional data points multiple times.

On the Level of Distortion. Many of the occasions where secure similarity approximation protocols are applied typically employ multiple layers of forensic investigation mechanisms or sanity

checks. A legal document comprised of random words or a genomic expression with random data are easy to spot. Therefore, in order to *minimize the likelihood of getting detected* (e.g., by another mechanism in place or during an audit), the attacker is incentivized to minimize the amount of changes to the input data—making the changes less incriminating and harder to detect. Extensive transformations (e.g., substituting large amounts of data with random noise) are likely recorded in system logs, and can be incriminating as they demonstrate malicious intent. Another illustration of this intent comes from the case of adversarial inputs on facial recognition - wearing a mask that covers the entire face clearly shows intent of avoiding facial recognition whereas an attacker that is wearing a set of "adversarially" decorated 3D-printed glasses [144] can fool such a system into matching the attacker to any maliciously-chosen individual.

Objectives of Perturbation Attacks. Note that d_{Jac} and d_{cos} as defined in Section 2.1.2 take values from the range $[0, 1]$. Ideally, a successful (ν, ν')-perturbation attack 1) maximizes the approximate distance $\hat{d}$ so as α and α^+ appear as distant as possible, e.g., $\hat{d}_{Jac}(\alpha, \alpha^+) \approx 1$, while 2) minimizes the true distance between α and α^+, e.g., $d_{Jac}(\alpha, \alpha^+) \approx 0$. We present two such attacks that utilize different tools, namely a randomized algorithm and constrained optimization formulation, and provide different guarantees. We slightly abuse notation and indicate by $\hat{d}_{Jac}$ and $\hat{d}_{cos}$ the approximate distance that is returned by a sketching protocol $(\mathcal{S}, \mathcal{G})$.

2.3.1 Attacking Minhash Sketches

Minhash sketches are used for approximating the Jaccard distance between sets. We propose a perturbation attack on minhash sketches guaranteed to perform the minimum number of changes to the original input set, thus minimizing $d(\alpha, \alpha^+)$. The perturbation that we apply is in the form of *adding new elements* to the set.

Intuition. The adversary takes as input a set S and the common random input r_{cmn}. The goal is to augment S with the smallest number of new elements in order to create S^+, such that $\hat{d}_{Jac}(S, S^+) = 1$. Recall that the approximate Jaccard distance between two sets is maximized when their κ-dimensional sketches $\sigma()$ differ in all dimensions, i.e., quantity $\hat{d}_{Jac}$ in equation (2.2) is equal to 1. Thus, the adversary is looking for at most[3] κ new elements such that every dimension of sketch $\sigma(S^+)$ is different from $\sigma(S)$. We denote by t' the number of samples drawn from the metric space.

Theorem 1. *Let S be the set of s values from the range $[0, m]$ that is given as an input to Algorithm*

[3]There is a case where the same new element of S^+ can contribute to more than one locations of the sketch $\sigma(S^+)$.

Algorithm 1: Attack Perturb on Minhash Sketches

Input: Original set S, common randomness r_{cmn}, sketch size κ, attempts t' to augment the original set
Output: S^+ s.t. $\hat{d}_{Jac}(S, S^+) = 1$, $d_{Jac}(S, S^+) = \frac{\kappa}{s+\kappa}$

```
Use r_cmn to sample κ hash functions (h_1, ..., h_κ);
σ(S) ← ( min_{x∈S}(h_1(x)), ..., min_{x∈S}(h_κ(x)) );
S^+ ← S;
for i = 1 to t' do
    Sample an element z_i ∉ S uniformly at random;
    for j = 1 to κ do
        if h_j(z_i) < min_{x∈S}(h_j(x)) then
            S^+ ← S^+ ∪ {z_i};
        end
    end
end
```

1. Let κ be the number of dimensions of the minhash sketch according to (2.2). Then a quasilinear number t' of samples are enough for Algorithm 1 to mount a successful $(1, \frac{\kappa}{s+\kappa})$-perturbation attack for minhash sketching with probability at least

$$\Pr\left(\{\textit{Succesful Attack}\}|(t' \geq 2c(s+1)\ln^3(s))\right) \geq 1 - \frac{6\kappa c^{1/2}}{s^c},$$

for any constant $c > 0$ assuming the codomain of the hash function is $\Omega(s\log^4(s))$.

Proof. If we augment set S with an element z_i picked uniformly at random from the set $[m]$ of available elements, we get $\Pr\left(\min\{\pi^{(j)}(S \cup \{z\})\} = \pi^{(j)}(z)\right) = 1/(|S|+1) = 1/(s+1)$ where $j \in [k]$ and $i \in [t]$. Therefore we sample t elements z_i in total and we test if the sample has smaller value than the current minimum, under the j-th permutation $\pi^{(j)}$. Let $Z_i^{(j)}$ be a random variable where $i \in [t]$ and $j \in [k]$. Random variable $Z_i^{(j)}$ takes value 1 if for the i-th sample $z_i \notin S$ holds $\pi^{(j)}(z_i) < \min\{\pi^{(j)}(S)\}$ and zero otherwise. The probability that $Z_i^{(j)}$ takes value 1 is:

$$\Pr(Z_i^{(j)} = 1) = \Pr(\min\{\pi^{(j)}(S \cup \{z_i\})\} = \pi^{(j)}(z_i)\big) = \frac{1}{s+1}$$

Let t be the number of distinct samples that we draw, then by $Z^{(j)}$ we denote the random variable such that $Z^{(j)} = \sum_{i=1}^{t} Z_i^{(j)}$ for the chosen t. Notice that since $\pi^{(j)}$ is a permutation if t samples from $[m]$ are distinct, then their corresponding outputs with respect to the permutation function are distinct; this is a part that we revisit in the proof. The probability that $Z^{(j)} = 0$ means that none of the sampled elements is smaller than the current minimum of S according to $\pi^{(j)}$. Given this probability we can compute the probability that at least one of the sampled elements elements is

smaller than $\min\{\pi^{(j)}(S)\}$.

We use the following expression of the Chernoff bound [121]:

$$Pr(Y \leq (1-\delta)\mu) \leq e^{-\mu\delta^2/2}. \tag{2.6}$$

Let $t = 2c(s+1)\ln^2(s)$ be the number of distinct samples that we draw and define δ as $\delta = \frac{1}{\sqrt{\ln(s)}}$. Then the mean of $Z^{(j)}$ is $E[Z^{(j)}] = 2c(s+1)\ln^2(s)\frac{1}{(s+1)} = 2c\ln^2(s)$ and from (2.6) we have:

$$\begin{aligned}
\Pr\left(Z^{(j)} \leq (1-\delta)\mu\right) &= \Pr\left(Z^{(j)} \leq (1-\frac{1}{\sqrt{\ln(s)}})2c\ln^2(s)\right) \\
&= \Pr\left(Z^{(j)} \leq \frac{\sqrt{\ln(s)}-1}{\sqrt{\ln(s)}}2c\sqrt{\ln(s)}\sqrt{\ln^3(s)}\right) \\
&= \Pr(Z^{(j)} \leq 2c(\sqrt{\ln(s)}-1)\ln^{3/2}(s)) \\
&\leq e^{-(2c\ln^2(s))(\frac{1}{\sqrt{ln(s)}})^2\frac{1}{2}} = e^{-c\ln(s)} = \frac{1}{s^c}.
\end{aligned}$$

Also notice that for $s > 3$ we have $2c(\sqrt{\ln(s)}-1)\ln^{3/2}(s) > 0$, therefore $\Pr(Z^{(j)} = 0) \leq \Pr(Z^{(j)} \leq 2c(\sqrt{\ln(s)}-1)\ln^{3/2}(s))$. Thus, we have $\Pr(Z^{(j)} \geq 1) = 1 - \Pr(Z^{(j)} = 0) \geq 1 - \frac{1}{s^c}$. The above event is based on the premise that we have $t = 2c(s+1)\ln^2(s)$ distinct elements z_i that are not in set S, which give t *distinct* outputs with respect to $\pi^{(j)}$. For efficiency reasons though, we use a hash function $h^{(j)}$ instead of a permutation $\pi^{(j)}$, thus the elements z_i for $i \in [t]$ do not necessarily give distinct outputs with respect to $h^{(j)}$.

Formally, let $h^{(j)} : [m] \to R$ be a hash function with domain $[m]$ and range R for which $|R| = n$. The attacker has to choose an input x_i from $[m]$ and as a second step compute $h^{(j)}(x_i) \in R$ in order to get an element from R.

This brings up the question, how many elements do we need to sample from $[m]$ in order to get $t = 2c(s+1)\ln^2(s)$ distinct elements from R that do not belong to set S? We answer the above question using a balls-and-bins argument on a given hash function $h^{(j)}$. The cardinality of R, that is $|R| = n$, represents the number of available bins. The number of samples that we draw, that is t', represents the number of balls that we throw. We define as $B_S^{(j)}$ the set of bins that contain a ball of the set S with respect to $h^{(j)}$. Let $t' \geq t$ be the number of balls that we throw, we compute the probability that t out of the total t' balls land in distinct bins and those bins do not belong to $B_S^{(j)}$.

Notice that the cardinality of $B_S^{(j)}$ can be at most s, therefore we compute the probability of

landing t balls in distinct bins among the available $n-s$ bins. Let $X_q^{(j)}$ be the random variable that takes value 1 if the q-th ball lands in a bin that is empty where $q \in [t']$; takes value 0 otherwise. Then the probability that $X_q^{(j)}$ takes value 1 is $\Pr(X_q^{(j)}=1) = \frac{n-s-(q-1)}{n} \geq \frac{n-s-(t'-1)}{n}$. For simplicity we approximate $\Pr(X_q^{(j)}=1)$ with the value $\frac{n-s-(t'-1)}{n}$ which is a tight lower bound on the probability that the q-th ball lands in an empty bin.

Let $X^{(j)}$ be the random variable that counts how many balls landed in an empty bin, then $X^{(j)} = \sum_{q=1}^{t'} X_q^{(j)}$. Let t' take the value $t' = t\ln(s)$, then we have $X^{(j)} = \sum_{q=1}^{t\ln(s)} X_q^{(j)}$ and its expectation is:

$$E[X^{(j)}] = E[\sum_{q=1}^{t\ln(s)} X_q^{(j)}] = \sum_{q=1}^{t\ln(s)} E[X_q^{(j)}]$$

$$= t\ln(s)\frac{n-s-t\ln(s)}{n} = t\ln(s) - \frac{st\ln(s)}{n} - \frac{t^2\ln^2(s)}{n}.$$

Random variables $X_q^{(j)}$ for $q \in [t']$ are clearly dependent, therefore we can not use the Chernoff bound of equation (2.6). We continue the analysis using the Poisson approximation technique [121] so we can work with independent random variables instead. Thus we define random variable $Y_q^{(j)}$ with parameter $\frac{n-s-(t'-1)}{n}$ and form their sum as $Y^{(j)} = \sum_{q=1}^{t'} Y_q^{(j)}$ that has expectation $\mu = t\ln(s) - \frac{st\ln(s)}{n} - \frac{t^2\ln^2(s)}{n}$. Given the sum of t' Poisson random variables $Y^{(j)}$, we are interested in bounding the probability that more than t out of the t' have value 1. In case $t < \mu$ we can use the following Chernoff bound for the sum of independent Poisson random variables $Y^{(j)}$:

$$\Pr(Y^{(j)} \leq t) \leq \frac{e^{-\mu}(e\mu)^t}{t^t}$$

If inequality $t' < \mu$ holds then, the following lower bound on n should also hold:

$$t < \mu \Rightarrow t < t\ln(s) - \frac{st\ln(s)}{n} - \frac{t^2\ln^2(s)}{n}$$

$$\Rightarrow n > \frac{s\ln(s) + t\ln^2(s)}{\ln(s)-1} > s + t\ln(s). \tag{2.7}$$

Thus assumption 1 is **(A1)** $n > s + t\ln(s)$. Using the above Chernoff bounds we have:

$$\Pr(Y^{(j)} \leq t) \leq \frac{e^{-\mu}(e\mu)^t}{t^t} = e^{-t\ln(s)+\frac{ts\ln(s)}{n}+\frac{t^2\ln^2(s)}{n}} e^t (\frac{\mu}{t})^t$$

$$= s^{-t} s^{\frac{ts}{n}} s^{\frac{t^2 \ln(s)}{n}} s^{2c(s+1)\ln(s)} (\frac{\mu}{t})^t$$

$$= s^{-t} s^{\frac{ts}{n}} s^{\frac{t^2 \ln(s)}{n}} s^{2c(s+1)\ln(s)} (t\ln(s) - \frac{st\ln(s)}{n} - \frac{t^2 \ln^2(s)}{n})^t t^{-t}$$

$$\leq s^{-t} s^{\frac{ts}{n}} s^{\frac{t^2 \ln(s)}{n}} s^{2c(s+1)\ln(s)} (\ln(s))^t$$

$$= (s^{\frac{s+t\ln(s)}{n} + \frac{2c(s+1)\ln(s)}{t} - 1} \ln(s))^t.$$

We further assume that $n > (s + t\ln(s))\ln(s)$ which is stronger than **(A1)**, so we have

$$\textbf{(A2)}\ n > (s + t\ln(s))\ln(s) \Rightarrow n > (s\ln(s) + 2c(s+1)\ln^4(s))$$

. We also use the fact that $t = 2c(s+1)\ln^2(s)$ in order to get the following expression:

$$\leq (s^{\frac{1}{\ln(n)} + \frac{1}{\ln(s)} - 1} \ln(s))^t = (s^{\frac{1}{\ln(n)} + \frac{1}{\ln(s)} - 1} s^{\log_s(\ln(s))})^t$$

$$= (s^{\frac{1}{\ln(n)} + \frac{1}{\ln(s)} - 1} s^{\frac{\ln(\ln(s))}{\ln(s)}})^t = (s^{\frac{2+\ln(\ln(s))}{\ln(n)} - 1})^t$$

$$= (s^{\frac{2+\ln(\ln(s))}{\ln(n)} 2c(s+1)\ln^2(s) - 2c(s+1)\ln^2(s)})$$

$$= (s^{(2+\ln(\ln(s)))2c(s+1)\ln(s) - 2c(s+1)\ln^2(s)})$$

$$= (s^{-(2+\ln(\ln(s)))2c(s+1)\ln(s) + 2c(s+1)\ln^2(s)})^{-1}$$

$$= \frac{1}{s^{2c(s+1)\ln(s)\left(\ln(s) - (2+\ln(\ln(s)))\right)}} \leq \frac{1}{s^{4c}}$$

, where we assumed that $(s+1)\ln(s)\big(\ln(s) - (2 + \ln(\ln(s)))\big) > 2$ which is true for $s > 24$.

Switching from the poisson approximation to the exact case (Corollary 5.9)

$$\Pr(Y^{(j)} \leq t) \leq \frac{1}{s^{4c}} \Rightarrow$$

$$\Pr(X^{(j)} \leq t) \leq e\sqrt{m}\frac{1}{s^{4c}} = \frac{e\sqrt{2c(s+1)\ln^3(s)}}{s^{4c}}$$

$$< \frac{e2c^{1/2}s^{1/2}\ln^{3/2}(s)}{s^{4c}} < \frac{6c^{1/2}}{s^{2c}}.$$

Thus we have that

$$\Pr(X^{(j)} = 0) \leq \Pr(X^{(j)} \leq t) \Rightarrow$$

$$\Pr(X^{(j)} \geq 1) = 1 - \Pr(X^{(j)} = 0) \geq 1 - \Pr(X^{(j)} \leq t) = 1 - \frac{6c^{1/2}}{s^{2c}}$$

, where we assume that $t \geq 0$.

We define as Q the random variable that takes value 1 if the output set S^+ of the algorithm builds a signature $\sigma(S^+)$ that is dissimilar to the signature $\sigma(S)$ of the input set and takes value 0 otherwise.

$$\Pr(Q = 1) \geq \Pr(Q = 1| \bigcap_{1 \leq j \leq \kappa} X^{(j)} \geq 1) \Pr(\bigcap_{1 \leq j \leq \kappa} X^{(j)} \geq 1)$$

$$= \Pr(Q = 1| \bigcap_{1 \leq j \leq \kappa} X^{(j)} \geq 1)(1 - \Pr(\bigcup_{1 \leq j \leq \kappa} X^{(j)} \geq 1))$$

, using the union bound we get:

$$\geq \Pr(Q = 1| \bigcap_{1 \leq j \leq \kappa} X^{(j)} \geq 1)(1 - \sum_{j=1}^{\kappa} \Pr(X^{(j)} \geq 1))$$

$$\geq \Pr(Q = 1| \bigcap_{1 \leq j \leq \kappa} X^{(j)} \geq 1)(1 - \kappa \cdot \frac{6c^{1/2}}{s^{2c}}) = 1 \cdot (1 - \frac{6\kappa c^{1/2}}{s^{2c}})$$

$$= 1 - \frac{6\kappa c^{1/2}}{s^{2c}}.$$

□

Attacking Synthetic Data. We demonstrate the frequency of success and the efficiency of the perturbation attack on synthetic data. We tested setups that range across all different variables of the problem: 1) dimension of the sketch $\kappa \in \{10, 50, 100, 200\}$, 2) size of the set under attack $s \in \{500, 1000, 10000\}$, 3) desired mis-approximation $\hat{d}_{Jac}() = 1$ or $\hat{d}_{Jac}() \geq 0.9$. Works such as [115] deploy a sketch of 64 bits to capture similarity of a collection of 8 billion webpages. Therefore, we think that sketches with size in the 10-200 range are indicative of what might be used in practice. The attack is implemented in `C++` where the elements of the original set are randomly generated numbers from a universe of size $2 \cdot 10^5$. We used 4-wise independent hash functions, and run 5,000 instantiations for each of the above setups. As observed in Table 2.1, when the desired approximation

is $\hat{d}_{Jac}() \geq 0.9$, the attack succeeds in *all* instantiations, and its execution time is less than 1 sec in most of the parameterizations. In this scenario it is enough for the adversary to discover smaller minhashes for 90% of the κ entries of the minhash sketch. Thus, if there are some small minhash values in the original sketch, the adversary can ignore those and "break" the rest of the sketch, whereas in the case of $\hat{d}_{Jac} = 1$ the adversary is forced to continue searching so as to "break" all κ minhashes. Overall, the frequency of success is extremely high, but there are a few cases for which the probabilistic guarantees of Theorem 1 are not met. One explanation is that the analysis was performed assuming that hash functions are truly random, whereas in the experiment we use 4-wise independent hashing. Table 2.1 clearly demonstrates that the probabilistic perturbation attack on minhash sketches succeeds in the vast majority of the instantiations and the total time ranges from less than a second to a couple of minutes even when dealing with sets that contain thousands of elements.

Attacking Real Data. To further verify the effectiveness of the attack we tested in real data using the bag of words dataset of Enron emails[4] which according to EDRM [5] has served for many years as *an industry-standard dataset* for e-discovery. We highlight that the findings of the attack on the synthetic data are expected to be similar to those on any real data, regardless of the context of the document, e.g., email, legal document. This is because the hash functions used are sampled uniformly at random and are independent of the input. In this real dataset every email is transformed into a multiset of words where the stop-words are removed. In this context Jaccard distance captures the similarity between any pair of emails. In our experiment we use the standard Rabin-Karp rolling hash function modulo $n = 105,943$. For simplicity we choose the size of the minhash sketch to be $\kappa = 5$ and the value of c to be 2 (see Theorem 1). Without loss of generality, for the purposes of this evaluation we focus on email with id-549 (denoted as set S_{549}), with size $s = 1181$ words, 492 of which are unique.

The average time to mount 100 instantiations of the attack was 2.2 seconds. Specifically, 83 out of 100 instantiations mounted successfully a $(0.004, 1)$-perturbation attack and terminated *in less than 1 second*. The remaining 17 instantiations took between 3 to 22 seconds due to the fact that at least one of the minhash values of the original sketch was already too small (< 10). Figure 1.1 illustrates one of the successful attacks where by adding the 5 words {pursued, glide, ralston, alluring, sensor} in the current email, i.e., create S^{+}_{549}, the approximate distance becomes 1, while the real distance is 0.004.

[4] https://archive.ics.uci.edu/ml/datasets/Bag+of+Words

Thus, any future comparison between S^{+}_{549} and a similar email will result in mis-approximation.

2.3.2 Attacking Cosine Sketches

Cosine sketching is used for approximating the cosine distance between vectors. We propose a perturbation attack on cosine sketching guaranteed to output $\hat{d}_{cos}(\cdot) = 1$, while the exact distance between the perturbed and the original vectors depends on the solution of the formulated constrained non-convex optimization problem. Our perturbation is in the form of *adding a new vector* $\vec{x}$ to the original vector $\vec{v}$.

Algorithm 2: Attack Perturb on Cosine Sketch

Input: $\vec{v} \in \mathbb{R}^n, r_{cmn}, \kappa$
Output: $\nu, \vec{v^+} \in \mathbb{R}^n$ s.t. $\hat{d}_{cos}(\vec{v}, \vec{v^+}) = 1$, $d_{cos}(\vec{v}, \vec{v^+}) = \nu$

1 Use r_{cmn} to sample vectors $(\vec{w_1}, \ldots, \vec{w_\kappa})$ from the unit $(n-1)$-sphere
2 Solve the following optimization problem

$$\vec{x} = \underset{\vec{x} \in \mathbb{R}^n}{\text{argmax}} \quad \frac{\vec{v} \cdot (\vec{v} + \vec{x})}{||\vec{v}||_2 ||\vec{v} + \vec{x}||_2}$$
$$\text{subject to} \quad \text{sgn}(\vec{w_i}^T \vec{v}) \cdot (\vec{w_i}^T (\vec{v} + \vec{x})) \leq 0, \; i = 1, \ldots, \kappa.$$

$\nu = d_{cos}(\vec{v}, \vec{v} + \vec{x})$
3 **return** $\nu, \vec{v^+} = \vec{v} + \vec{x}$

Intuition. The adversary takes as input the original vector $\vec{v} \in \mathbb{R}^n$ and r_{cmn}. The goal is to add a new vector $\vec{x}$ to the original $\vec{v}$ in order to create $\vec{v^+}$ such that $\hat{d}_{cos}(\vec{v}, \vec{v^+}) = 1$. Recall that the approximate cosine distance between two vectors is maximized when their κ-dimensional sketches $\sigma()$ differ in all dimensions. Thus the addition of vector $\vec{x}$ to $\vec{v}$ must *change the sign* of the κ inner products with respect to Equation (2.5) and consequently flip the bits of the sketch $\sigma(\vec{v^+})$. Overall, the adversary wants to maximize the approximate cosine distance, handled by the constraints of the optimization problem, and minimize the exact cosine distance, handled by the objective function of the optimization.

In Algorithm 2 the function $\text{sgn}(x)$ has output -1 in case $x < 0$ and output $+1$ in case $x \geq 0$. The unit $(n-1)$-sphere is defined as the set of points $\{u \in \mathbb{R}^n : ||u|| = 1\}$. Notice that minimizing the exact cosine distance is equivalent to maximizing the cosine similarity as it is described in Equation (2.4), so our problem is formed as a maximization of the cosine similarity $C(\vec{v}, \vec{v} + \vec{x})$. Algorithm 2 requires to solve a non-convex, non-linear, high-dimensional constrained optimization problem. Furthermore the objective function presents discontinuity at point $\vec{x} = -\vec{v}$, see Figure 2.1.

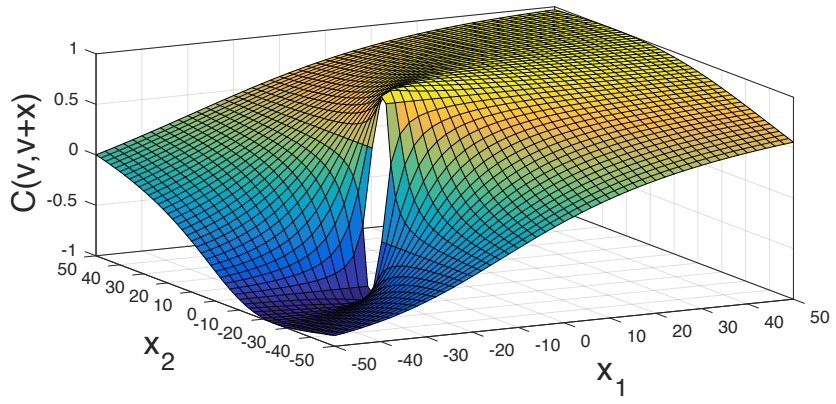


Figure 2.1: An illustration of the objective function of the maximization problem of Algorithm 2 where $n = 2$ and $\vec{v} = (20, 10)$. The X-,Y-axis denote the x_1 and x_2 dimension of vector to be added, $\vec{x}$.

Since closed form solutions are generally challenging for this setup, we approximate the solution of the above problem using iterative algorithms from standard optimization toolboxes. Figure 2.1 visualizes the objective function for a toy example where $v \in \mathbb{R}^2$. Due to the lack of formal guarantees about the quality of the approximation, we present the effectiveness of the attack in a form of a remark.

Remark 1. *Let $\vec{v} \in \mathbb{R}^n$ be the vector that is given as an input to Algorithm 2. Let also $\vec{w_i} \in \mathbb{R}^n$ be a vector sampled from the unit $(n-1)$-sphere using r_{cmn} according to Algorithm 2, where $i = [1, \kappa]$. Then, Algorithm 2 is a successful $(\nu, 1)$-perturbation attack for cosine sketching, where ν is the achieved similarity of the perturbed input that is returned by the algorithm.*

Attacking Synthetic Data. We evaluated the performance of the attack on synthetic data using the interior point algorithm of MATLAB [9] where the input vector $\vec{v}$ is an n-dimensional vector where the value of each element is chosen uniformly at random from $[0, 10^5]$. We tested setups that range across the different variables of the problem: 1) the number of dimensions of the vector under attack $n \in \{500, 1000, 5000\}$, and 2) the size of the sketch under attack $\kappa \in \{10, 50, 100, 200\}$. To generate vectors $\vec{w_i} \in \mathbb{R}^n$ we sampled vectors from the $(n-1)$-sphere of unit radius centered at the origin. We run the above setups with 10 different common randomness inputs r_{cmn} and present the mean. As one may observe in Table 2.2, the approximate distance is always $\hat{d}_{cos} = 1$ which implies that all the returned solutions were part of the feasible region of the optimization problem. The value of the exact cosine distance d_{cos} between the original and the perturbed data depends on the returned solution of the optimization problem. Note that different solution methods can potentially result in even lower d_{cos} values. Depending on the optimization toolbox and the number

of dimensions the time performance may vary, in our case all the executions terminated within a couple of minutes.

Table 2.2: Evaluation of the perturbation attack on cosine sketches over synthetic data. The data points show the average value over 10 instantiations.

	$n = 500$		$n = 1,000$		$n = 5,000$	
κ	d_{cos}	$\hat{d}_{cos}$	d_{cos}	$\hat{d}_{cos}$	d_{cos}	$\hat{d}_{cos}$
10	0.005	1	0.002	1	0.0005	1
50	0.02	1	0.01	1	0.002	1
100	0.05	1	0.02	1	0.005	1
200	0.11	1	0.06	1	0.01	1

Attacking Real Data. We demonstrate the perturbation attack of Algorithm 2 on a real dataset[5] of human gene-expression levels that can be found in the work of Notteramn *et al.* [131]. The authors perform a clustering analysis on the vectors of gene-expression levels so as to capture similarity patterns between healthy patients, patients with adenoma and patients with adenocarcinoma. It is rather common to perform similarity-based analysis on genomic data with the goal of understanding and diagnosing diseases at the molecular level. We highlight that the findings of the attack on the synthetic are expected to be similar to those on any real data. This is because the generative model of the input vector $\vec{v}$ does not affect the sign of the inner product with a random vector $\vec{w}$. We approximate the solution of the optimization problem using the interior point algorithm from MATLAB [9]. We use a cosine sketch of $\kappa = 100$ dimensions and we repeat the experiment for 10 different initializations of the vectors $(\vec{w_1}, \ldots, \vec{w_\kappa})$. The input vector is denoted as $\vec{v}$ and it has $n = 7,086$ dimensions each of which is a gene-expression measured with a DNA microarray. We report that all of the instantiations successfully satisfied the optimization constraints and thus resulted in $\hat{d}_{cos}(\vec{v}, \vec{v^+}) = 1$. The average ν value was 0.0033 with a maximum value of 0.0039. Therefore, on average we mounted a successful $(0.0033, 1)$-perturbation attack. One of the recorded instantiations is illustrated in Figure 2.2 where it shows that if the adversary perturbs $\vec{v}$ to form $\vec{v^+}$ then according to the cosine sketching initialization we have $\hat{d}_{cos}(\vec{v}, \vec{v^+}) = 1$, even though their exact cosine distance is $d_{cos}(\vec{v}, \vec{v^+}) = 0.0036$.

[5] http://genomics-pubs.princeton.edu/oncology/

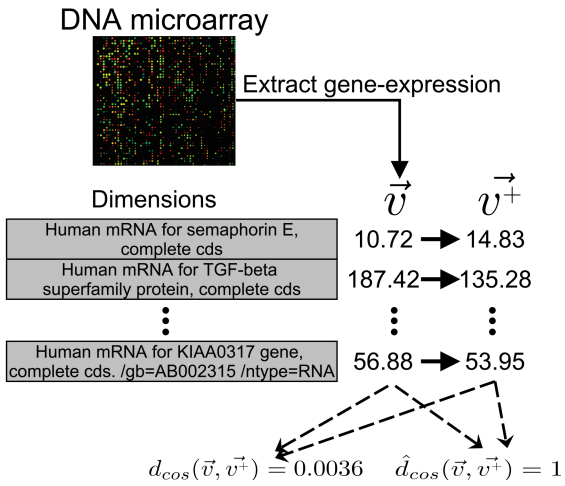


Figure 2.2: Illustration of the perturbation attack on a gene-expression of an adenoma patient. The proposed attack on vector $\vec{v}$ outputs the perturbed $\vec{v^+}$ with approximate cosine distance 1 even though the exact distance is 0.0036.

2.4 Server-Aided Approximation

In this Section we reframe the architecture of secure sketching protocols so that we can 1) still use the well-studied sketching techniques based on the common random input r_{cmn}, and 2) *eliminate* the possibility of an offline perturbation attack. In our proposed *server-aided* design we introduce a new semi-honest entity, i.e., the server S, that has exclusive access to the common random input r_{cmn} and assists in the sketching protocols. Compared to previous approaches, the main difference of our design is that a client *does not have direct access* to the common random input. The sketching function that was previously a local computation (as described in Section 2.2), is replaced by a two-party protocol denoted as Sketching between the server and the client. We capture the new *functionality* as follows:

Functionality $\mathcal{F}_{\text{S-approx}}$

- *Input:* Party C_A provides v_A, party C_B provides v_B, party S provides r_{cmn}.
- *Output:* All three parties receive $\hat{d}(v_A, v_B)$.

Notice that in case client C_A (similarly for client C_B) observes the values of σ_A, then it is possible for the C_A to infer r_{cmn}, which is an attack that defeats the purpose of the server-aided model. For example, in the case where r_{cmn} is used to sample k-independent hash functions then the set of values of σ_A consists of the evaluations of the above hash functions. An adversary that observes the

output of the polynomial-based hash function can easily infer the coefficients of the hash function by solving a system of equations [87]. In our design, protocol Sketching outputs the *encrypted* sketch σ_A to C_A so as to avoid the above type of attacks.

The Real Model. Let Π be a three-party protocol computing the functionality $\mathcal{F}_{\text{S-approx}}$. For ease of exposition we consider the execution of Π in the presence of an adversary $\mathcal{A}$ as being coordinated by a nonuniform environment $\mathcal{Z} = \{\mathcal{Z}_\lambda\}$, much like [32, 92]. In the beginning $\mathcal{Z}$ gives input $(1^\lambda, v_A)$ to C_A, input $(1^\lambda, v_B)$ to C_B, input $(1^\lambda, r_{cmn})$ to S, and gives z and X to $\mathcal{A}$, where z denotes an auxiliary input and $X \in \{C_A, C_B, S\}$ is the corrupted party. At this point the parties interact with each honest party behaving as instructed by Π. At the end of the protocol, adversary $\mathcal{A}$ gives to $\mathcal{Z}$ an output which is an arbitrary function of $\mathcal{A}$'s view. Additionally, $\mathcal{Z}$ gets the output of the honest parties. Finally, environment $\mathcal{Z}$ outputs a bit. We denote as $\mathsf{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda)$ the random variable that represents the value of this bit.

The Ideal Model. In this model there is a trusted party that computes $\mathcal{F}_{\text{S-approx}}$ on behalf of the parties. Similar to the real model, environment $\mathcal{Z}$ gives inputs $(1^\lambda, v_A)$ and $(1^\lambda, v_B)$ to parties C_A and C_B, respectively. It gives input $(1^\lambda, r_{cmn})$ to S, and also gives z and X to $\mathcal{A}'$ where $X \in \{C_A, C_B, S\}$ indicates the corrupted party. All the parties send their input to the trusted party. The trusted party computes $\mathcal{F}_{\text{S-approx}}$ and sends $\hat{d}(v_A, v_B)$ to all the parties. In the next step $\mathcal{A}'$ outputs to $\mathcal{Z}$ an arbitrary function of the view of $\mathcal{A}'$. The honest parties also give their output to $\mathcal{Z}$. As a final step $\mathcal{Z}$ outputs a bit. We denote as $\mathsf{IDEAL}_{\Pi,\mathcal{A}',\mathcal{Z}}(\lambda)$ the random variable that represents the value of this bit.

Definition 4. *Let* Π *be a three-party protocol for computing* $\mathcal{F}_{S\text{-}approx}$ *functionality. We say that* Π securely computes $\mathcal{F}_{S\text{-}approx}$ *in the presence of semi-honest adversaries corrupting one party if for any PPT semi-honest adversary* $\mathcal{A}$ *there exists a PPT semi-honest adversary* $\mathcal{A}'$ *such that, for every polynomial size circuit family* $\mathcal{Z} = \{\mathcal{Z}_\lambda\}$ *corrupting at most one party, the following is negligible:*

$$|\Pr[\mathit{REAL}_{\Pi,\mathcal{A},\mathcal{Z}}(\lambda) = 1] - \Pr[\mathit{IDEAL}_{\Pi,\mathcal{A}',\mathcal{Z}}(\lambda) = 1]|.$$

Notice that if the adversary were to corrupt both a client and the server then she would have access to the common random input, and thus become capable of mounting a perturbation attack. We note here that the server-aided approach has been successfully deployed [20, 97, 98] in various

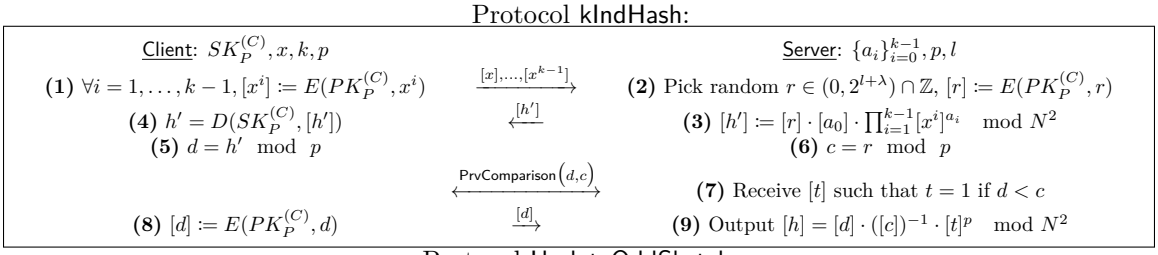
other problems. The proposed perturbation attacks of the previous Section are based on the fact that all clients have offline and direct access to the common random input r_{cmn}. Under our server-aided design an adversary can only attempt an *online* attack, hoping to infer the r_{cmn} from the value of $\hat{d}(\cdot)$, by performing a series of Sketching and Reconstruct executions. Using rate-limiting techniques (e.g., [98]) one can mitigate such an online attack. This scenario, however, is beyond the scope of this thesis.

Table 2.3: An overview of the protocols. For brevity we assume that the public keys of the server $PK_P^{(S)}, PK_{GM}^{(S)}$ and the client $PK_P^{(C)}, PK_{GM}^{(C)}, PK_{DGK}^{(C)}$ are publicly available and thus not passed as an input to the protocols.

Protocol	Client (C) Input	Server (S) Input	Client (C) Output	Server (S) Output	Summary
PrvComparison*	a	b	-	$[t]$	[t=1] if $a < b$, [0] otherwise
EncComparison*	$SK_P^{(C)}, SK_{GM}^{(C)}, l$	$[a], [b], l$	t	-	t=1 if $a < b$, 0 otherwise
EncComparison2	$SK_P^{(C)}, SK_{GM}^{(C)}, l$	$[a], [b], l$	-	$\|t\|$	$\|t = 1\|$ if $a < b$, $\|0\|$ otherwise
ChangePartyEnc*	$SK_{GM}^{(C)}$	$SK_{GM}^{(S)}, \|b\|$	$\|b\|$	-	Encrypts $\|b\|$ under $SK_{GM}^{(S)}$
kIndHashing	$SK_P^{(C)}, x, k, p$	$\{a_i\}_{i=0}^{k-1}, p$	-	$[h]$	$[(\sum_{i=0}^{k-1} a_i x^i) \mod p]$
EncHashing	$SK_P^{(C)}, k, p$	$[x], \{a_i\}_{i=0}^{k-1}, p$	-	$[h]$	$[(\sum_{i=0}^{k-1} a_i x^i) \mod p]$
FindMin	$SK_{GM}^{(C)}, SK_P^{(C)}, l$	$\{[y_i]\}_{i=1}^{n}, l$	-	$[min]$	$[\min_i y_i]$
UpdateOddSketch	$SK_{GM}^{(C)}, SK_P^{(C)}, SK_{DGK}^{(C)}, u, k$	$[x], \{a_i\}_{i=0}^{k-1}, u, (\|skt_0\|, \ldots, \|skt_{u-1}\|)$	-	$(\|skt'_0\|, \ldots, \|skt'_{u-1}\|)$	Update odd sketch with x
SketchingCosine	$\vec{v}, SK_P^{(C)}, SK_{GM}^{(C)}$	$\{\vec{w}_i\}_{i=1}^{\kappa}, SK_{GM}^{(S)}$	$(\|\sigma_1\|, \ldots, \|\sigma_\kappa\|)$	-	Encr. cosine signature
SketchingOdd	$S, k, u, SK_{GM}^{(C)}, SK_P^{(C)}, SK_{DGK}^{(C)}$	$\{h_i^{min}\}_{i=1}^{\kappa}, h_{odd}, p, SK_P^{(S)}, SK_{GM}^{(S)}$	$(\|\sigma_1\|, \ldots, \|\sigma_\kappa\|)$	-	Encr. odd-minhash signature

Composition of Building Blocks. We define separate building blocks that can be combined and the proof of security for the overall construction can be derived using modular composition [31]. The model is called *hybrid model with ideal access to functions* $f_1, \ldots, f_m$ or simply $(f_1, \ldots, f_m)$-hybrid model . In the real life experiment we assume the existence of an incorruptible trusted party T for evaluating $f_1, \ldots, f_m$; all parties hand their input to T and they receive the corresponding output. As a next step, the ideal evaluation of f at each step is replaced with the invocation of a protocol—we refer the reader to [31] for a detailed exposition. In case the function returns an encrypted output, a party passes a public key as an input and we assume that the necessary encryption algorithm is hardwired to the corresponding function. Table 2.3 summarizes all the two-party protocols, which in our case are executed between the server and the client. Using the above building blocks we construct a secure two-party analogue for minhashing (via odd sketches) and cosine skething. Protocols that are marked with $*$ in Table 2.3 are simple modification of already proposed protocols [15, 24, 153]. We note that we follow the protocol and encryption notation established by the work of Bost *et al.* [24].

Protocol kIndHash:



Protocol UpdateOddSketch:

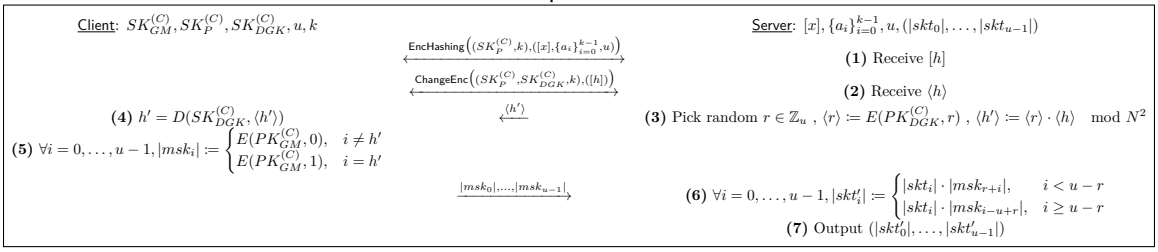


Figure 2.3: Two-party protocols between a client and the server that are used as building blocks for sketching.

2.4.1 Building Blocks

A Note on the Security Proofs. Our security proofs take the classic simulation based approach for semi-honest adversaries on the hybrid model with ideal access to functions [31] and show that a party's view in a protocol execution is simulatable given its input, its output (if any), and access to a series of ideal functionalities. On the one hand we have the hybrid world were protocols have access to functions that are invoked by specific step of the protocol and on the other hand we have the ideal world where the simulator lives. Thus, the participating parties learn nothing from the protocol's execution beyond what can be derived from their input. For the sake of brevity we don't denote the public keys, whenever there is an encryption we indicate which public key is used.

k-Independent Hashing over Encrypted Data. The functionality of $\mathcal{F}_{\text{kIndHash}}$ is as follows. The input of the server is the bit length l and the set of parameters of a k-independent hash function—i.e., the coefficients $\{a_i\}_{i=0}^{k-1}$, the prime p of a $(k-1)$ degree polynomial on $\mathbb{Z}_p$. The client has the input x which is used to evaluate the polynomial on $\mathbb{Z}_p$. The degree of the polynomial as well as the modulo p are considered to be known to both parties. At the end of the protocol the server receives the evaluation of the polynomial $a_0 + a_1 x + \ldots + a_{k-1} x^{k-1} \mod p$ that is encrypted with the client's public key. We do not use a private polynomial evaluation due to the fact that we require the output to be *encrypted*. The server should not learn any information about the client's input x and the client should not learn any information about the coefficients $\{a_i\}_{i=0}^{k-1}$ of the polynomial.

Protocol EncHashing

Client: $SK_P^{(C)}, k, p$		Server: $[x], \{a_i\}_{i=0}^{k-1}, p$
(2) $\forall i = 2, \ldots, k-1, h_i = D(SK_P^{(C)}, [h_i])$	$\xleftarrow{[h_2],\ldots,[h_{k-1}]}$	**(1)** $\forall i = 2, \ldots, k-1$, Pick $r_i \in (0, 2^l) \cap \mathbb{Z}$, $[h_i] \coloneqq [x]^{r_i} \mod N^2$
(3) $\forall i = 2, \ldots, k-1, [h'_i] \coloneqq E(PK_P^{(C)}, h_i^i)$	$\xrightarrow{[h'_2],\ldots,[h'_{k-1}]}$	**(4)** $\forall i = 2, \ldots, k-1, [x_i] \coloneqq [h'_i]^{r_i^{-i}} \mod N^2$
(6) $h' = D(SK_P^{(C)}, [h'])$	$\xleftarrow{[h']}$	**(5)** Pick $r \in \mathbb{Z}_u$, $[h'] \coloneqq [r] \cdot [a_0] \cdot [x]^{a_1} \prod_{i=2}^{k-1} [x_i^i]^{a_i} \mod N^2$
(7) $d = h' \mod p$		**(8)** $c = r \mod p$
	$\xleftrightarrow{\mathsf{PrvComparison}(d,c)}$	Receive $[t]$ such that $t = 1$ if $d < c$
(9) $[d] \coloneqq E(PK_P^{(C)}, d)$	$\xrightarrow{[d]}$	**(10)** Output $[h] = [d] \cdot ([c])^{-1} \cdot [t]^p$

Protocol EncComparison

Client: $SK_P^{(S)}, SK_{GM}^{(S)}, l$		Server: $[a], [b], l$
		(1) $[x] \coloneqq [2^l] \cdot [b] \cdot [a]^{-1} \mod N^2$
		(2) Pick a random $r \in (0, 2^l) \cap \mathbb{Z}$
(4) $z = D(SK_P^{(C)}, [z])$	$\xleftarrow{[z]}$	**(3)** $[z] \coloneqq [x] \cdot [r] \mod N^2$
(5) $d \coloneqq z \mod 2^l$		
		(6) $c \coloneqq r \mod 2^l$
	$\xleftrightarrow{\mathsf{PrvComparison}(d,c)}$	Receive $\lvert t' \rvert$ such that $t' = 1$ if $c < d$
(7) $\lvert z_l \rvert \leftarrow E(PK_{GM}^{(C)}, z_l)$	$\xrightarrow{\lvert z_l \rvert}$	
		(8) $\lvert r_l \rvert \coloneqq E(PK_{GM}^{(C)}, r_l)$
(10) Output $t = D(SK_{GM}^{(C)}, \lvert t \rvert)$	$\xleftarrow{\lvert t \rvert}$	**(9)** $\lvert t \rvert \coloneqq \lvert z_l \rvert \cdot \lvert r_l \rvert \cdot \lvert t' \rvert \cdot \lvert 1 \rvert$

Protocol FindMin:

Client: $SK_{GM}^{(C)}, SK_P^{(C)}, l$		Server: $\{[y_i]\}_{i=1}^n, l$
		(1) Pick a rand. permutation π over $\{1, \ldots, n\}$
		(2) $[min] \coloneqq [y_{\pi(1)}]$
for $i = 2$ **to** n **do**		
(3) Receive bit t_i s.t. $t_i = 1$ if $min < y_{\pi(i)}$	$\xleftarrow{\mathsf{EncComparison}((SK_P, SK_{GM}, l), ([min], [y_{\pi(i)}], l))}$	
		(4) Pick random $r_i, s_i \in (0, 2^{l+\lambda}) \cap \mathbb{Z}$
	$\xleftarrow{[b_i],[c_i]}$	**(5)** $[b_i] \coloneqq [min] \cdot [r_i] \mod N^2$, $[c_i] \coloneqq [y_{\pi(i)}] \cdot [s_i] \mod N^2$
if t_i is 1 **then**		
(6a) $[c_i] \coloneqq \text{Refresh}([b_i])$		
else		
(6b) $[c_i] \coloneqq \text{Refresh}([c_i])$		
end if	$\xrightarrow{[c_i],[t_i]}$	**(7)** $[min] \coloneqq [c_i] \cdot ([t_i] \cdot [-1])^{s_i} \cdot [t_i]^{-r_i} \mod N^2$
end for		
		(8) Output $[min]$

Figure 2.4: Protocol EncComparison is a slight modification of the comparison protocol found in [24, 153]. Protocol EncComparison2 is the same up to step **(9)** where it terminates by outputting $|t|$ to the Server.

Lemma 1. *Protocol kIndHash correctly and securely computes $\mathcal{F}_{kIndHash}$ in the ($\mathcal{F}_{PrvComp}$)-hybrid model.*

Proof. Let's consider first the case where Client is corrupted, denoted as $\mathcal{A}$; notice that Client has no output. Thus we only need to show that a simulator can generate the view of incoming messages received by the $\mathcal{A}$.

Adversary (or simulator) $\mathcal{A}'$ is given $(SK_P^{(C)}, x, k, p)$ and 1^λ and works as follows:

- $\mathcal{A}'$ starts by simulating $\mathcal{A}$.
- $\mathcal{A}'$ receives $[x], \ldots, [x^{k-1}]$ from $\mathcal{A}$.

- $\mathcal{A}'$ picks a random $\widetilde{h'} \in (0, 2^{l+\lambda}) \cap \mathbb{Z}$, encrypts it to get $[h']$ using $PK_P^{(C)}$, and sends it to $\mathcal{A}$.
- $\mathcal{A}'$ receives $\widetilde{d}$ and $PK_P^{(C)}$ from $\mathcal{A}$ which are sent to $\mathcal{F}_{\text{PrvComp}}$.
- $\mathcal{A}'$ receives $[\widetilde{d}]$ from $\mathcal{A}$.
- $\mathcal{A}'$ outputs $\perp$.

Now we show that the view of $\mathcal{A}$ in the simulation with $\mathcal{A}'$ is indistinguishable from its view in a hybrid execution using a series of games.

- *Game-0:* Same as the hybrid execution.
- *Game-1:* Same as Game-0 except that h' in step **(3)** is replaced by with $\widetilde{h'} \in (0, 2^{l+\lambda}) \cap \mathbb{Z}$. In Game-0 h' is the blinded value of $\prod_{i=1}^{k-1} \alpha_i x^i$ with random r, but in this game $\widetilde{h'}$ is picked uniformly at random from $(0, 2^{l+\lambda}) \cap \mathbb{Z}$. Thus the distribution of $h' = r + \prod_{i=1}^{k-1} \alpha_i x^i$ and $\widetilde{h'}$ are computational indistinguishable.

Thus, the view of $\mathcal{A}$ in the simulation with $\mathcal{A}'$ is indistinguishable from its view in a hybrid execution.

Let's consider the case where Server is corrupted. Simulator $\mathcal{A}'$ is given $(\{a_i\}_{i=0}^{k-1}, p)$, 1^λ, output $[h]$, and works as follows:

- $\mathcal{A}'$ starts by simulating $\mathcal{A}$.
- $\mathcal{A}'$ generates $k-1$ distinct encryptions of 1, namely $\{[\widetilde{x_i}]\}_{i=1}^{k-1}$, and sends them to $\mathcal{A}$.
- $\mathcal{A}'$ receives $[h']$.
- $\mathcal{A}'$ encrypts bit $\widetilde{t} = 1$ with $PK_P^{(C)}$ and sends $[\widetilde{t}]$ to $\mathcal{A}$.
- $\mathcal{A}'$ receives $\widetilde{c}$ from $\mathcal{A}$ which is sent to $\mathcal{F}_{\text{PrvComp}}$.
- $\mathcal{A}'$ encrypts the bit $[\widetilde{d}] = 1$ with $PK_P^{(C)}$ and sends it to $\mathcal{A}$.
- Finally $\mathcal{A}'$ outputs $[h]$.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.

- *Game-1:* Same as Game-0 except that the messages $x, \ldots, x^{k-1}$ in step **(1)** are replaced with $\{\widetilde{x_i}\}_{i=1}^{k-1}$ where all of them have value 1. By semantic security of the Paillier's cryptosystem the distribution of the ciphertexts $\{[x^i]\}_{i=1}^{k-1}$ and $\{[\widetilde{x_i}]\}_{i=1}^{k-1}$ are computationally indistinguishable.

- *Game-2:* Same as Game-1 except the encrypted bit $[t]$ returned by $\mathcal{F}_{\text{PrvComp}}$ in step **(7)** is replaced by the encryption of $\widetilde{t} = 1$. Since Server receives the ciphertext of Paillier, the messages $[t]$ and $[\widetilde{t}]$ are computationally indistinguishable.

- *Game-3:* Same as Game-2 except the encrypted value $[d]$ sent to the Server in step **(8)** is replaced by the encryption of $\widetilde{d} = 1$. As before, due to CPA security of Paillier the messages $[d]$ and $[\widetilde{d}]$ are computationally indistinguishable.

Thus, the view of $\mathcal{A}$ in the simulation with $\mathcal{A}'$ is indistinguishable from its view in a hybrid execution. □

Update Encrypted Odd Sketch. The functionality of $\mathcal{F}_{\text{UpdateOddSketch}}$ is as follows. The input of the server consists of i) the bits of an odd sketch $(skt_0, \ldots, skt_{u-1})$ encrypted with the client's public key, ii) the parameters of the $(k-1)$-degree polynomial that is used as the hash function h_{odd}, and iii) the input x of the polynomial encrypted with client's public key. The input of the client is the set of secret keys. At the end of the protocol the server receives an updated odd sketch where the bit in location $h_{odd}(x)$ of the sketch is flipped, while the client receives no output. The server and the client should not learn which bit of the odd sketch is flipped or the input x of the polynomial.

As a first step protocol EncHashing is invoked, which returns to the server the hash value $h_{odd}(x) = (\sum_{i=0}^{k-1} a_i x^i) \mod u$ encrypted with the client's public key. As a next step we change the encryption from Paillier to DGK, since DGK embeds reductions modulo u to its homomorphic operations (see Section 2.1.3). Thus, the result of the blinding in step **(3)** is a random element from the range $[0, u-1]$. Then the server sends the blinded ciphertext $\langle h' \rangle$ to the client who decrypts it (step **(4)**) and prepares an encrypted bit mask for the server. The mask is a bit string of length u where every bit has value 0 except the bit in position h' which has value 1. We note here that the value of u is relatively small since it represents the length of the *succinct* odd sketch. Blinding the value h with r at step **(3)** has the effect of shifting the only "1" value of the bit mask by r positions. Therefore, as a final step, the server has to re-arrange the encrypted mask so as to remove the effect

of the blinding with r. In order to cancel-out the effect of blinding, the server has to "pull" the 1 value back by r positions by applying the following transformation:

$$(|msk_0|, \ldots, |msk_{r-1}|, |msk_r|, \ldots, |msk_u|)$$
$$\rightsquigarrow (|msk_r|, \ldots, |msk_u|, |msk_0|, \ldots, |msk_{r-1}|).$$

The ciphertexts of the updated bit mask are multiplied with the GM ciphertexts of the input sketch, resulting in a homomorphic XOR operation—and, therefore, the desired bit is flipped.

A similar blinding process with Paillier, as opposed to DGK, would give an element r from the range $[0, 2^{\lambda+l}]$. For a standard instantiation $\lambda = 1024$, which gives an impractical number of ciphertexts in steps **(5)** and **(6)**, consequently making the protocol completely impractical. Protocol UpdateOddSketch invokes two protocols (EncHashing, ChangeEnc), performs $2u + 3$ homomorphic operations, and one roundtrip.

Lemma 2. *Protocol UpdateOddSketch correctly and securely computes $\mathcal{F}_{UpdateOddSketch}$ in the ($\mathcal{F}_{EncHashing}$, $\mathcal{F}_{ChangeEnc}$)-hybrid model.*

Proof. Let's assume that $\mathcal{A}$ corrupts the Client. Adversary $\mathcal{A}'$ is given

$(PK_{GM}^{(C)}, SK_{GM}^{(C)}, PK_{P}^{(C)}, SK_{P}^{(C)}, PK_{DGK}^{(C)}, SK_{DGK}^{(C)}, u, k)$

,and 1^{λ} and works as follows:

- $\mathcal{A}'$ starts by simulating $\mathcal{A}$.
- $\mathcal{A}'$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\text{EncHashing}}$.
- $\mathcal{A}'$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\text{ChangeEnc}}$.
- $\mathcal{A}'$ picks a random value $\widetilde{h'}$ from space $\mathbb{Z}_u$ and sends its DGK encryption $\langle \widetilde{h'} \rangle$ with key $PK_{DGK}^{(C)}$ to $\mathcal{A}$.
- $\mathcal{A}'$ receives $|msk_0|, \ldots, |msk_{u-1}|$ from $\mathcal{A}$.
- $\mathcal{A}'$ outputs $\perp$.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.

- *Game-1:* Same as Game-0 except that h' of step **(3)** is replaced with a random value $\widetilde{h'}$ from space $\mathbb{Z}_u$. In Game-1 h' is the blinded value of h with random r, but in this game $\widetilde{h'}$ is picked uniformly at random from $\mathbb{Z}_u$. Recall that in the DGK cryptosystem the homomorphic operations are performed modulo u in the plaintext space. Thus the distribution of h' and $\widetilde{h'}$ is identical.

Let's assume that $\mathcal{A}$ corrupts the Server. Simulator $\mathcal{A}'$ is given $(PK_{GM}^{(C)}, PK_{P}^{(C)}, PK_{DGK}^{(C)}, [x], \{\alpha_i\}_{i=0}^{k-1}, u, (|skt_0|, \ldots, |skt_{u-1}|))$,the output $(|skt'_0|, \ldots, |skt'_{u-1}|)$, and 1^λ and works as follows:

- $\mathcal{A}'$ starts by simulating $\mathcal{A}$.
- $\mathcal{A}'$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\text{EncHashing}}$.
- $\mathcal{A}'$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\text{ChangeEnc}}$.
- $\mathcal{A}'$ receives $\langle h' \rangle$ which is encrypted with $PK_{DGK}^{(C)}$ from $\mathcal{A}$.
- $\mathcal{A}'$ picks uniformly at random u random bits $\widetilde{msk_0}, \ldots, \widetilde{msk_{u-1}}$. Then $\mathcal{A}'$ encrypts them with key $PK_{GM}^{(C)}$ and sends $|\widetilde{msk_0}|, \ldots, |\widetilde{msk_{u-1}}|$ to $\mathcal{A}$.
- $\mathcal{A}'$ outputs $(|skt'_0|, \ldots, |skt'_{u-1}|)$.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.
- *Game-1:* Same as Game-0 except that the bits $msk_0, \ldots, msk_{u-1}$ of step **(5)** are replaced with a random random bits $\widetilde{msk_0}, \ldots, \widetilde{msk_{u-1}}$. Notice that server gets to see bits $\widetilde{msk_0}, \ldots, \widetilde{msk_{u-1}}$ encrypted using GM. From the CPA security of the GM cryptosystem the distribution of $|\widetilde{msk_i}|$ and $|msk_i|$ is computationally indistinguishable, for all $i \in \{0, \ldots, u-1\}$.

□

Find Minimum. Initially the server assigns the first encrypted value as the current minimum $[min]$. Then we compare the current minimum with the next encrypted value using the protocol EncComp, which outputs the result of the comparison without revealing the encrypted values to the key holder (i.e., the client). Notice, however, that if the server iterates through the ciphertexts in

the originally given order then the client can learn the index of the minimum value. To overcome this the server picks a random permutation π that is applied before any pairwise comparison (step **(1)**). Thus the client learns the index of the minimum value with respect to the secret random permutation that the server applied. After the execution of the comparison protocol the client returns a re-encryption $[c_i]$ of the smallest among the input values $[min], [y_{\pi(i)}]$, so as not to reveal to the server which of the two ciphertexts is smaller. Re-encryption (denoted as Refresh) can be achieved by either decrypting and re-encrypting the ciphertext, or by using the homomorphic properties of the cryptosystem to refresh the randomness. Since the client can decrypt $[min]$ and $[y_{\pi(i)}]$, the server blinds the ciphertexts using r_i and s_i so as to create the blinded ciphertexts $[b_i]$ and $[c_i]$. In the final step we deal with two cases. If the result of the comparison is $min < y_{\pi(i)}$ (i.e., $t_i = 1$) the server subtracts the blinding r_i from the value that the client returned. Otherwise the server subtracts s_i. Protocol FindMin performs $n-1$ encrypted comparisons of l bit integers, $8(n-1)$ homomorphic operations and $n-1$ roundtrips.

Lemma 3. *Protocol FindMin correctly and securely computes $\mathcal{F}_{FindMin}$ in the ($\mathcal{F}_{EncComp}$)-hybrid model.*

Proof. Let's assume that $\mathcal{A}$ corrupts the Client. Adversary (or simulator) $\mathcal{A}'$ is given $((PK_P^{(C)}, SK_P^{(C)}, PK_{GM}^{(C)}, SK_{GM}^{(C)}, l)$ and 1^λ and works as follows:

- $\mathcal{A}'$ starts by simulating $\mathcal{A}$.
- $\mathcal{A}'$ receives the input of $\mathcal{A}$ to $\mathcal{F}_{\text{EncComp}}$.
- $\mathcal{A}'$ sends to the $\mathcal{A}$ a random bit $\widetilde{t_i}$.
- $\mathcal{A}'$ picks a pair of random values $\widetilde{b_i}$ and $\widetilde{c_i}$ from the range $(0, 2^{l+\lambda}) \cap \mathbb{Z}$ and sends their encryption with $PK_P^{(C)}$ to $\mathcal{A}$.
- $\mathcal{A}'$ receives the ciphertexts $[c_i]$ and $[t_i]$.
- $\mathcal{A}'$ repeats the above four steps $n-1$ times.
- $\mathcal{A}'$ outputs $\perp$.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.

- *Game-1:* Same as Game-0 except that the output bit of the ideal function $\mathcal{F}_{\text{EncComp}}$ t_i in step **(3)** is replaced with the random bit $\widetilde{t_i}$. From the security of $\mathcal{F}_{\text{EncComp}}$ the client does not learn any information about the values that the Server compares. Since there is no prior information about the result of the comparison and no inferred information from ideal function, from Client's perspective any output of $\mathcal{F}_{\text{EncComp}}$ is equally probable.

- *Game-2:* Same as Game-1 except the values b_i and s_i in step **(4)** are replaced with a pair of random values $\widetilde{b_i}$ and $\widetilde{c_i}$ from the range $(0, 2^{l+\lambda}) \cap \mathbb{Z}$. In Game-1 b_i is the blinded value of min with random r_i, but in this game $\widetilde{b_i}$ is picked uniformly at random from $0, 2^{l+\lambda}) \cap \mathbb{Z}$. Thus the distribution of b_i and $\widetilde{b_i}$ is computationally indistinguishable. Similarly in Game-1, c_i is the blinded value of $y_{\pi(i)}$ with random s_i, but in this game $\widetilde{c_i}$ is picked uniformly at random from $(0, 2^{l+\lambda}) \cap \mathbb{Z}$. Thus the distribution of c_i and $\widetilde{c_i}$ is computationally indistinguishable.

Thus, the view of $\mathcal{A}$ in the simulation with $\mathcal{A}'$ is indistinguishable from its view in a hybrid execution.

Now let's assume that $\mathcal{A}$ corrupts the Server. Simulator $\mathcal{A}'$ is given $(\{[y_i]\}_{i=1}^{n}, l)$, 1^{λ}, the output $[min]$, and works as follows:

- $\mathcal{A}'$ starts by simulating $\mathcal{A}$.

- $\mathcal{A}'$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\text{EncComp}}$.

- $\mathcal{A}'$ receives $[b_i]$ and $[c_i]$.

- $\mathcal{A}'$ picks a pair of random values $\widetilde{c_i}$ and $\widetilde{t_i}$ from the range $(0, 2^{l+\lambda}) \cap \mathbb{Z}$ and sends their encryption with $PK_P^{(C)}$ to $\mathcal{A}$.

- $\mathcal{A}'$ repeats the above three steps $n-1$ times.

- $\mathcal{A}'$ outputs $[min]$.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.

- *Game-1:* Same as Game-0 except that the transferred ciphertexts c_i and t_i before step **(7)** are replaced with the encryption of the random values $\widetilde{c_i}$ and $\widetilde{t_i}$ with the key $PK_P^{(C)}$. From the

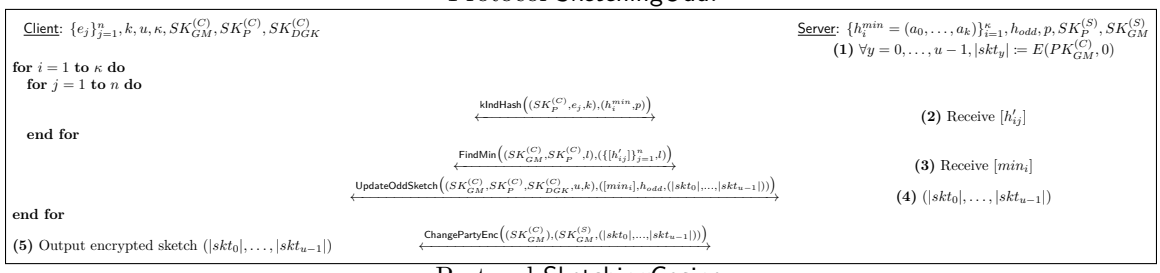
CPA security of the Paillier cryptosystem the distribution of $[c_i]$ and $[\widetilde{c_i}]$ is computationally indistinguishable. A similar argument holds for the distribution of $[t_i]$ and $[\widetilde{t_i}]$.

□

2.4.2 Protocols for the Server-Aided Model

Approximating Jaccard Distance via Odd Sketches. We employ the protocols of the previous subsection as building blocks to *securely* approximate Jaccard distance using the approach by Mitzenmacher *et al.* [122]. As denoted in Figure 2.5, the input of the server consists of the set of κ minhash functions $\{h_i^{min}\}_{i=1}^{\kappa}$, the hash function for the creation of the odd sketch h_{odd}, as well as the corresponding secret keys. Recall that $\{h_i^{min}\}_{i=1}^{\kappa}$ and h_{odd} are generated using the common randomness r_{cmn} that can only be accessed by the server. The input of the client consists of her data, denoted as the elements $\{e_j\}_{j=1}^{n}$, as well as the publicly known moduli p, u, and the secret keys. At the end of the protocol the client receives the odd sketch encrypted with the server's public key.

Protocol SketchingOdd:



Protocol SketchingCosine:

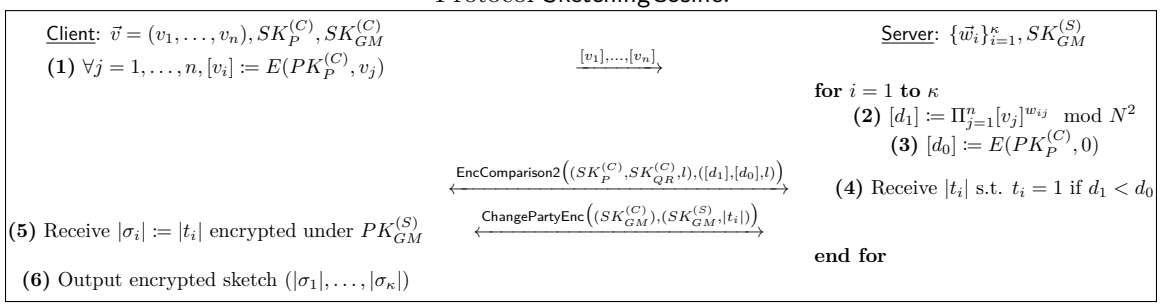


Figure 2.5: The sketching protocols between the server and the client for the server-aided model.

Lemma 4. *Protocol* SketchingOdd *correctly and securely computes* $\mathcal{F}_{SketchingOdd}$ *in the* ($\mathcal{F}_{kIndHashing}$, $\mathcal{F}_{FindMin}$, $\mathcal{F}_{UpdateOddSketch}$, $\mathcal{F}_{ChangePartyEnc}$)*-hybrid model.*

Proof. Let's assume that $\mathcal{A}$ corrupts the Client. Simulator $\mathcal{S}_{\mathrm{C}}$ is given

$(\{e_j\}_{j=1}^{n}, k, u, PK_{GM}^{(C)}, SK_{GM}^{(C)}, PK_{P}^{(C)}, SK_{P}^{(C)}, PK_{DGK}^{(C)}, SK_{DGK}^{(C)})$

$,1^{\lambda}$, the output $(|\sigma_1|, \ldots, |\sigma_\kappa|)$ and works as follows:

- $\mathcal{S}_{\mathrm{C}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{kIndHash}}$.
- $\mathcal{S}_{\mathrm{C}}$ repeats the above step n times.
- $\mathcal{S}_{\mathrm{C}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{FindMin}}$.
- $\mathcal{S}_{\mathrm{C}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{UpdateOddSketch}}$.
- $\mathcal{S}_{\mathrm{C}}$ repeats the above four steps κ times.
- $\mathcal{S}_{\mathrm{C}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{ChangePartyEnc}}$.
- $\mathcal{S}_{\mathrm{C}}$ outputs $(|\sigma_1|, \ldots, |\sigma_\kappa|)$.

The security proof based on the above simulator is straight-forward. The simulator only accesses the ideal functionalities that are provided to $\mathcal{F}_{\text{Sketching-Odd}}$.

Now let's assume that $\mathcal{A}$ corrupts the Server. Simulator $\mathcal{S}_{\mathrm{S}}$ is given

$(\{h_i^{min} = (a_0, \ldots, a_k)\}_{i=1}^{\kappa}, h_{odd}, p, PK_{P}^{(S)}, SK_{P}^{(S)}, PK_{GM}^{(S)}, SK_{GM}^{(S)})$

$,1^{\lambda}$, and works as follows:

- $\mathcal{S}_{\mathrm{S}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{kIndHash}}$.
- $\mathcal{S}_{\mathrm{S}}$ picks uniformly at random a value $\widetilde{h'_{ij}}$ from the range $(0, p) \cap \mathbb{Z}$. Then it encrypts it using $PK_{P}^{(C)}$ so as to get $[\widetilde{h'_{ij}}]$ which is sent to $\mathcal{A}$.
- $\mathcal{S}_{\mathrm{S}}$ repeats the above two step n times.
- $\mathcal{S}_{\mathrm{S}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{FindMin}}$.
- $\mathcal{S}_{\mathrm{S}}$ picks uniformly at random a value $\widetilde{min_i}$ from the range $(0, p) \cap \mathbb{Z}$. Then it encrypts it using $PK_{P}^{(C)}$ so as to get $[\widetilde{min_i}]$ which is sent to $\mathcal{A}$.
- $\mathcal{S}_{\mathrm{S}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{UpdateOddSketch}}$.
- $\mathcal{S}_{\mathrm{S}}$ picks uniformly at random u bits, namely $\widetilde{skt_0}, \ldots, \widetilde{skt_{u-1}}$. Then it encrypts them using $PK_{GM}^{(C)}$ so as to get $(|\widetilde{skt_0}|, \ldots, |\widetilde{skt_{u-1}}|)$ which is sent to $\mathcal{A}$.
- $\mathcal{S}_{\mathrm{S}}$ repeats the above seven steps κ times.

- $\mathcal{S}_{\mathrm{S}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{ChangePartyEnc}}$.
- $\mathcal{S}_{\mathrm{S}}$ outputs $\perp$.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.
- *Game-1:* Same as Game-0 except that the transferred ciphertext $[h'_{ij}]$ before step **(2)** is replaced with the encryption of a random value $\widetilde{h'_{ij}}$ from the range $(0, p) \cap \mathbb{Z}$. From the CPA security of the Paillier cryptosystem the distribution of $[h'_{ij}]$ and $[\widetilde{h'_{ij}}]$ is computationally indistinguishable.
- *Game-2:* Same as Game-1 except that the transferred ciphertext $[min_i]$ before step **(3)** is replaced with the encryption of a random value $\widetilde{min_i}$ from the range $(0, p) \cap \mathbb{Z}$. From the CPA security of the Paillier cryptosystem the distribution of $[min_i]$ and $[\widetilde{min_i}]$ is computationally indistinguishable.
- *Game-3:* Same as Game-2 except that the transferred ciphertexts $(|skt_0|, \ldots, |skt_{u-1}|)$ before step **(4)** is replaced with the encryption of a random bits $(|\widetilde{skt_0}|, \ldots, |\widetilde{skt_{u-1}}|)$. From the CPA security of the GM cryptosystem, for every $i \in \mathbb{Z}_u$ the distribution of $|skt_i|$ and $|\widetilde{skt_i}|$ is computationally indistinguishable.

□

Approximating Cosine Distance via Cosine Sketching. We approximate cosine distance as follows. The input of the server consists of the vectors $\vec{w}_i$, that are sampled uniformly at random from the $(n-1)$-sphere. The input of the client consists of her data which is represented by the vector $\vec{v}$. Note that vectors $\vec{w}_i$ are generated using the common randomness r_{cmn} that can only be accessed by the server. At the end of the protocol the client receives the cosine sketch encrypted with the server's public key.

Lemma 5. *Protocol* SketchingCosine *correctly and securely computes* $\mathcal{F}_{SketchingCosine}$ *in the* ($\mathcal{F}_{EncComparison2}$, $\mathcal{F}_{ChangePartyEnc}$)*-hybrid model.*

Proof. Let's assume that $\mathcal{A}$ corrupts the Client. Simulator $\mathcal{S}_{\mathrm{C}}$ is given

$(\vec{v} = (v_1, \ldots, v_n), PK_P^{(C)}, SK_P^{(C)}, PK_{GM}^{(C)}, SK_{GM}^{(C)}, PK_{GM}^{(S)})$

,1^λ, the output $(|\sigma_1|, \ldots, |\sigma_\kappa|)$ and works as follows:

- $\mathcal{S}_{\mathrm{C}}$ receives $[v_1], \ldots, [v_n]$ from $\mathcal{A}$.
- $\mathcal{S}_{\mathrm{C}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{EncComp2}}$.
- $\mathcal{S}_{\mathrm{C}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{ChangePartyEnc}}$.
- $\mathcal{S}_{\mathrm{C}}$ picks a random bit $\widetilde{t_i}$ and it encrypts it with $PK_{GM}^{(S)}$. Then it sends $|\widetilde{t_i}|$ to $\mathcal{A}$.
- $\mathcal{S}_{\mathrm{C}}$ repeats the above three steps $k-1$ times.
- $\mathcal{S}_{\mathrm{C}}$ outputs $(|\sigma_1|, \ldots, |\sigma_\kappa|)$.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.
- *Game-1:* Same as Game-0 except that the transferred ciphertext $|t_i|$ before step **(5)** is replaced with the encryption of a random bit $|\widetilde{t_i}|$ encrypted with $PK_{GM}^{(S)}$. From the CPA security of the GM cryptosystem the distribution of $|t_i|$ and $|\widetilde{t_i}|$ is computationally indistinguishable.

Now let's assume that $\mathcal{A}$ corrupts the Server. Simulator $\mathcal{S}_{\mathrm{S}}$ is given $(\{\vec{w}_i\}_{i=1}^{\kappa}, PK_{GM}^{(S)}, SK_{GM}^{(S)}, PK_{P}^{(C)}, PK_{GM}^{(C)})$, 1^λ, and works as follows:

- $\mathcal{S}_{\mathrm{S}}$ picks uniformly at random values $\widetilde{v_1}, \ldots, v_n$ from the range $(0, 2^l) \cap \mathbb{Z}$. Then it encrypts it using $PK_{P}^{(C)}$ so as to get $[\widetilde{v_1}], \ldots, [v_n]$ which is sent to $\mathcal{A}$.
- $\mathcal{S}_{\mathrm{S}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{EncComp2}}$.
- $\mathcal{S}_{\mathrm{S}}$ picks uniformly at random bit $\widetilde{t_i}$. Then it encrypts it using $PK_{GM}^{(C)}$ so as to get $|\widetilde{t_i}|$ which is sent to $\mathcal{A}$.
- $\mathcal{S}_{\mathrm{S}}$ receives the input of $\mathcal{A}$ to function $\mathcal{F}_{\mathrm{ChangePartyEnc}}$.
- $\mathcal{S}_{\mathrm{S}}$ repeats the above four steps k times.
- $\mathcal{S}_{\mathrm{S}}$ outputs $\perp$.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.

- *Game-1:* Same as Game-0 except that the transferred ciphertexts $[v_1], \ldots, [v_n]$ after step **(1)** are replaced with the encryption of a random values $\widetilde{v_1}, \ldots, \widetilde{v_n}$ from the range $(0, 2^l) \cap \mathbb{Z}$ with the key $PK_P^{(C)}$. From the CPA security of the Paillier cryptosystem the distribution of $[v_i]$ and $[\widetilde{v_i}]$ is computationally indistinguishable.

- *Game-2:* Same as Game-1 except that the transferred ciphertext $|t_i|$ after step **(4)** is replaced with the encryption of a random bit $\widetilde{t_i}$ with the key $PK_{GM}^{(C)}$. From the CPA security of the GM cryptosystem the distribution of $|t_i|$ and $[\widetilde{t_i}]$ is computationally indistinguishable.

□

Protocol Reconstruct:

$\underline{\mathsf{Client}_A}$: $\lvert\vec{\sigma}^A\rvert$		$\underline{\mathsf{Client}_B}$: $\lvert\vec{\sigma}^B\rvert$		$\underline{\mathsf{Server}}$: $SK_{GM}^{(S)}$
	$\xrightarrow{\lvert\vec{\sigma}^A\rvert}$	**(1)** Receive sketch $\lvert\vec{\sigma}^A\rvert$ **(2)** $\forall i \in \{0, \ldots, \kappa-1\}, \lvert\sigma'_i\rvert = \lvert\sigma_i^A\rvert \cdot \lvert\sigma_i^B\rvert$ **(3)** Pick a rand. perm. π over $\{0, \ldots, \kappa-1\}$ **(4)** Permute $\lvert\vec{\sigma}'\rvert$ w.r.t. π		
			$\xrightarrow{\lvert\vec{\sigma}'\rvert}$	**(5)** Decrypt all $\lvert\vec{\sigma}'\rvert$ **(6)** $c \leftarrow$Count 1s in σ'
(9) Output c/κ	$\xleftarrow{c}$	**(8)** Output c/κ	$\xleftarrow{c}$	**(7)** Output c/κ

Figure 2.6: The reconstruction of SketchingCosine between the server and the clients. The reconstruction for SketchingOdd is the same for steps **(1)**-**(6)**; steps **(7)**, **(8)** follow the reconstruction of Equation (2.3).

Reconstruct Protocol. The power of the sketching techniques that we chose for approximating Jaccard distance and cosine distance lies in the fact that their reconstruction function is *simple* and *efficient*. Both techniques follow the same reconstruction process which performs an exclusive-or operation between the two sketches, and then counts the number of 1 values (see Equations (2.3) and (2.5)). Taking advantage of the homomorphic properties of the GM cryptosystem we build an efficient Reconstruct protocols. See Figure 2.6.

Lemma 6. *Protocol Reconstruct is correct and secure in the semi-honest model.*

Proof. Notice that this is a tree party protocol between Client_A, Client_B, and the Server. Let's assume that $\mathcal{A}$ corrupts the Client_A. Simulator $\mathcal{S}_{\text{CA}}$ is given $\big(|\vec{\sigma}^A| = (|\sigma_0^A|, \ldots, |\sigma_{\kappa-1}^A|), PK_{GM}^{(S)}, \kappa\big)$, the output c/κ, 1^λ, and works as follows:

- $\mathcal{S}_{\text{CA}}$ receives $|\vec{\sigma}^A| = (|\sigma_0^A|, \ldots, |\sigma_{\kappa-1}^A|)$ from $\mathcal{A}$.

- $\mathcal{S}_{\text{CA}}$ performs the operation $(c/\kappa) \cdot \kappa = c$.
- $\mathcal{S}_{\text{CA}}$ sends c to $\mathcal{A}$.
- $\mathcal{S}_{\text{CA}}$ outputs c/κ.

The security proof based on the above simulator is straight-forward.

Let's assume that $\mathcal{A}$ corrupts the Client_B. Simulator $\mathcal{S}_{\text{CB}}$ is given $\big(|\vec{\sigma}^B| = (|\sigma_0^B|, \ldots, |\sigma_{\kappa-1}^B|), PK_{GM}^{(S)}, \kappa\big)$, the output c/κ, 1^λ, and works as follows:

- $\mathcal{S}_{\text{CB}}$ picks κ random bits so as to create $\widetilde{\vec{\sigma}^A} = (\widetilde{\sigma_0^A}, \ldots, \widetilde{\sigma_{\kappa-1}^A})$.
- $\mathcal{S}_{\text{CB}}$ encrypts the individual bits $\widetilde{\vec{\sigma}^A}$ with $PK_{GM}^{(S)}$ and sends the encrypted vector to $\mathcal{A}$.
- $\mathcal{S}_{\text{CB}}$ receives $|\vec{\sigma}'|$ from $\mathcal{A}$.
- $\mathcal{S}_{\text{CB}}$ performs the operation $(c/\kappa) \cdot \kappa = c$.
- $\mathcal{S}_{\text{CB}}$ sends c to $\mathcal{A}$.
- $\mathcal{S}_{\text{CB}}$ outputs c/κ.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.
- *Game-1:* Same as Game-0 except that the transferred ciphertexts $\vec{\sigma}^A = (\sigma_0^A, \ldots, \sigma_{\kappa-1}^A)$ before step **(1)** are replaced with the encryption of a random bits $\widetilde{\vec{\sigma}^A} = (\widetilde{\sigma_0^A}, \ldots, \widetilde{\sigma_{\kappa-1}^A})$ with the key $PK_{GM}^{(S)}$. From the CPA security of the GM cryptosystem the distribution of $|\sigma_i^A|$ and $[\widetilde{\sigma_i^A}]$ is computationally indistinguishable.

Finally let's assume that $\mathcal{A}$ corrupts the Server. Simulator $\mathcal{S}_{\text{S}}$ is given $(PK_{GM}^{(S)}, \kappa)$, the output c/κ, 1^λ, and works as follows:

- $\mathcal{S}_{\text{S}}$ performs the operation $(c/\kappa) \cdot \kappa = c$.
- $\mathcal{S}_{\text{S}}$ defines c bits with value 1, i.e., $\widetilde{\sigma_0'}, \ldots, \widetilde{\sigma_{c-1}'}$, and $\kappa - c$ bits with value 0, i.e., $\widetilde{\sigma_c'}, \ldots, \widetilde{\sigma_{\kappa-c}'}$. Thus, $\mathcal{S}_{\text{S}}$ creates $\widetilde{\vec{\sigma}'} = (\widetilde{\sigma_0'}, \ldots, \widetilde{\sigma_{\kappa-1}'})$, where the first c bits have value 1.
- $\mathcal{S}_{\text{S}}$ picks a random permutation π' over the set $\{0, \ldots, \kappa\}$. Then it permutes the values of $\widetilde{\vec{\sigma}'}$ with respect to π'.

- $\mathcal{S}_{\mathrm{S}}$ sends the permuted $\widetilde{\vec{\sigma}'}$ to $\mathcal{A}$.
- $\mathcal{S}_{\mathrm{S}}$ receives c from $\mathcal{A}$.
- $\mathcal{S}_{\mathrm{S}}$ outputs c/κ.

The games for the security proof are the following:

- *Game-0:* Same as the hybrid execution.
- *Game-1:* Same as Game-0 except that the κ bits $\vec{\sigma}' = \sigma'_0, \ldots, \sigma'_{\kappa-1})$ which are the result of step **(4)** are replaced with the permutation of c bits with value 1 and $\kappa - c$ bits with value zero denoted as $\widetilde{\vec{\sigma}'} = (\widetilde{\sigma'_0}, \ldots, \widetilde{\sigma'_{\kappa-1}})$. Notice that the number of bits set to 1 is the same in both $\vec{\sigma}'$ and $\widetilde{\vec{\sigma}'}$. Also notice that the values of the vectors $\vec{\sigma}'$ and $\widetilde{\vec{\sigma}'}$ are permuted according to random permutations π and π', respectively. Thus the distribution of 1s over the dimensions of the vectors is identical.

□

On the Choice of Building Blocks. Since our protocols follow a modular design, one can substitute the proposed building blocks with protocols that follow other MPC techniques so as to further optimize the performance of our constructions. The work presented in this thesis is meant to present to principles of this modular design and is not representative of a highly-optimized implementation. According to the work of Bost *et al.* [24], comparison protocols that utilize specialized homomorphic cryptosystems [62, 153] are more efficient when the input is encrypted. Thus, our implementation invokes variations of the above protocols, namely EncComparison and EncComparison2. For the comparison protocol on unencrypted inputs, Bost *et al.* [24] denote that a garbled circuit approach [18] results in a more efficient implementation. In our implementation we followed the work of Veugen [153], and therefore one can further speedup our implementation by invoking a garbled circuit design instead. We note that well-known protocols that are purely based on garbled circuits for functionality such as FindMin can not be deployed because the input of the FindMin is a set of *encrypted* inputs (see Table 2.3). A similar argument holds for the output of kIndHashing which is encrypted. Thus, to the best of our knowledge, the most promising speedup opportunity would be opting for garbled circuit designs for the simplest building blocks, such as comparison.

2.5 Scalability Evaluation

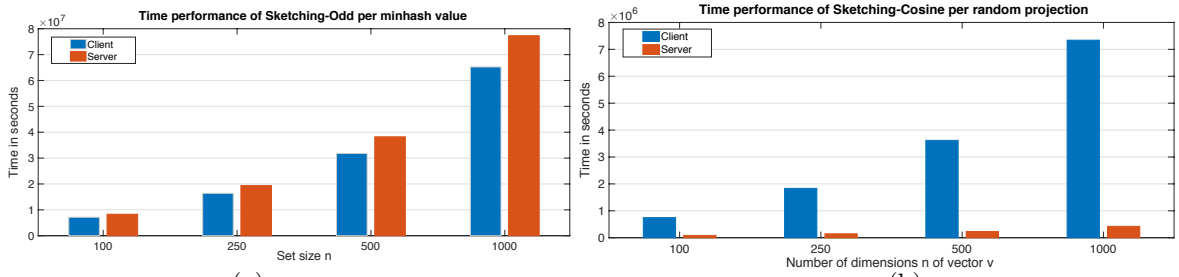
Implementation Setup. We implemented the proposed protocols in C++ using existing libraries as well as newly implemented building blocks. For serializing the communication between the server and client we use Protocol Buffers [78]. All the arithmetic operations are performed with the gmp multiple precision library [55]. We use the Advanced Crypto Software Collection [19] implementation of the Paillier cryptosystem, and an open-source implementation of the GM cryptosystem. We implemented the DGK cryptosystem in C++ following the design principles of [19] and the directions of the original work [47, 48].

For the minhashing via odd sketching protocols we choose the security parameter $\lambda = 100$. Given the scale of our experiments the k-independent hashing setup is the following: we choose $k = 4$ and a prime p that is at least an order of magnitude larger than the size of the set—i.e., $p \geq 10n$. As explained in the description of protocol UpdateOddSketch, prime u of the DGK cryptosystem is set to have the same value as the length of the odd sketch. As it is also noted in [24] the parameterization of Paillier has to be such that the homomorphic operations do not overflow the message space. To accomplish this instantiation we analyze the two phases of the protocol. The first phase is the kIndHashing computation; let l' be the maximum bit-length of the inputs x. In step **(1)** of protocol kIndHashing involves $(k-1)$ exponentiations among which the plaintext x^{k-1} can have the maximum length of $l'_{\max} = (k-1)l'$ bits. Step **(3)** of protocol kIndHashing involves $(k-1)$ multiplications and $(k+1)$ additions of numbers that are at most $l'_{\max}$ bits long. Therefore it is sufficient for N to be such that $\log N \geq (k^2 - k - 2)(l'/2) + 2 + \lambda$. After the execution of kIndHashing the numbers involved in protocols PrvComparison and EncComparison are $\log p$ bits long, since they are hash values. Thus protocols PrvComparison and EncComparison operate on integers that are at most $l = \log p$ bits long. Consequently, it is sufficient for N to be such that $\log N > \log p + \lambda + 1$. We satisfy the above inequalities by choosing $\log N \geq 1024$.

Regarding the protocols for cosine sketching, we also choose a security parameter $\lambda = 100$. Recall that vectors $\vec{w_i} = (w_{i1}, \ldots, w_{in})$ are sampled uniformly at random from the $(n-1)$-sphere, so each value w_{ij} is a real number. We can transform the above real numbers to integers by multiplying with a constant K and rounding, allowing us to interpret w_{ij} as part of Paillier's message space. The purpose of the random projection is to compute the sign of the inner product thus one can choose a relatively small K. In our implementation we choose $K = 1000$. Similarly to the previous

instantiation, the parameterization of Paillier should not overflow by the homomorphic operations of the encrypted inner product that is performed in step **(2)** of protocol Sketching-Cosine. Let l be the maximum length in bits of the entries in $\vec{v}$. Then step **(2)** of protocol Sketching-Cosine involves the multiplication of a $\log K$ bit long integer with an l bit long integer. Thus, it is sufficient for N to be such that $\log N \geq \log K + l + n$. Finally, in our implementation, both GM and DGK have moduli that are at least 1024 bits long. The implementation of the protocols and the serialization of the server is around 1400 lines, while the client is around 1100 lines.

Scalability. We evaluate the scalability of the server-aided design based on the described implementation setup. In Figure 2.7 we present the recorded computation time for the sketching protocols on a commercial laptop with 2.6 GHz Intel Core i5 CPU and 8GB DDR3 RAM.



(a) (b)

Figure 2.7: Subfigure (a): Time performance for varied set size of the SketchingOdd protocol. Time averaged for a single minhash over five runs. Subfigure (b): Time performance for varied number of vector dimensions of the SketchingCosine protocol. Time averaged for a single random projection over five runs.

The client and server have similar time performance for the SketchingOdd protocol. This is mostly because both parties are subject to a slowdown by a similar number of encrypt/decrypt operations. The time performance presented in Figure 2.7-(a) is for a a single minhash value (i.e., $\kappa = 1$), and an odd sketch of 151 bits (i.e., $u = 151$). Note that the computational overhead scales linearly with κ: for $\kappa > 1$ we have the same computational overhead as the one depicted in Figure 2.7-(a), only κ times larger. Notice, however, that the computation for each of the κ dimensions of the sketch is *independent* of each other, thus the overall task is parallelizable. On the other, hand the computational overhead of the client in protocol SketchingCosine is significantly higher than the one of the server. This is mainly caused by the encryption of each dimension of $\vec{v}$, which translates to a large number of exponentiations taking place in step **(1)** of the protocol. Furthermore, the performance of the server (time) is measured when we have a single random projection, i.e., $\kappa = 1$,

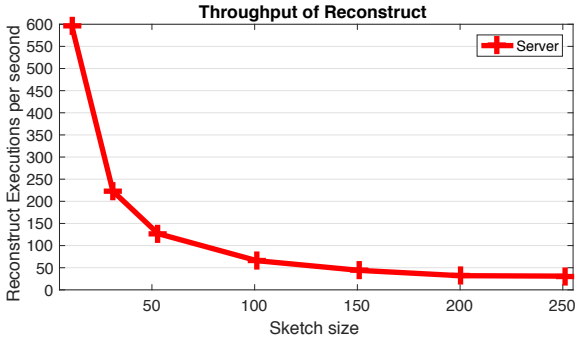


Figure 2.8: Throughput of the server Reconstruct protocol for varied sketch sizes. Average values over 5 runs.

thus steps **(2)**-**(4)** of SketchingCosine are repeated only once. Similar to the case of SketchingOdd, for $\kappa > 1$ the overall task is highly parallelizable into κ tasks. The communication overhead of the sketching protocols for various values of n is depicted in Table 2.4.

Table 2.4: Communication Overhead of Sketching. Average over 5 runs.

Protocol	n			
	100	250	500	1000
Sketch-Odd	1112 KB	2930 KB	6218 KB	13201 KB
Sketch-ShimHash	69 KB	165 KB	324 KB	644 KB

In our design we prioritize the *speedup the reconstruction protocol*, since it is the protocol that is executed multiple times throughout the lifetime of the system—once for every pairwise approximation. On the contrary, the sketching protocol is invoked only once for every high-dimensional data point, so as to create the sketch. Thus, using odd sketches (rather than regular minhashing) introduced, indeed, some overhead in the overall sketching protocol but resulted in a fast and more scalable reconstruction protocol. Generally, the reconstruction protocol from the server's perspective is the same, regardless of whether we are approximating Jaccard or cosine similarity, since the only task performed by the server is to decrypt κ ciphertexts encrypted under GM. The end result is a rather scalable performance illustrated in Figure 2.8.

2.6 Related Work

Aside from the secure sketching protocols mentioned earlier, there is a rich body of protocol that devise a combination of semi-homomorphic cryptosystems and garbled circuits to operate on encrypted data [15, 24, 100, 123, 157]. The work by Mironov *et al.* [120] introduces the model of *sketching*

in adversarial environments which is different in certain ways from what we consider in our work. Specifically, the work in [120] studies a model where a *single* party adversarially chooses the input *for all other parties* while they approximate joint functions on the adversarially chosen input. In their model, the adversarial inputs are provided to the parties in an *on-line* manner and thus the users update the sketch incrementally without being able to store the original information, much like in one-pass streaming algorithms. In our work, each party uses her own data which is stored locally. Our model is different from the data stream model, and follows more closely the published work on privacy-preserving sketches discussed above. The work by Naor *et al.* [127] introduces a new *adversarial model for Bloom filters*. The threat model of [127] is somewhat similar to our model, in the sense that both adversaries exploit the used randomness so as to violate the correctness of the computation. In terms of differences, our adversary has direct access to the randomness used, whereas for the case of [127] the adversary has only oracle access via the responses of the Bloom filter. Furthermore in our work sketching is just the first phase of the computation and the second phase consists of a secure computation protocol; on the contrary the work of [127] does not involve any form of secure computation.

There is a significant body of research focusing on the attack vectors that lay in the intersection of machine learning and privacy-preserving mechanisms [16, 34, 46, 65, 66, 118]. The line of research closer to our proposed attack is the work on Deep Learning in adversarial settings. Some works [12, 33, 134, 150] show how an adversary can *craft her input* so as to maximize the prediction error of a deep neural network (DNN). Interestingly, in this work we show that adversarial inputs are very effective not only with learning and classification mechanisms, e.g., DNN, but also with simple randomized algorithms, e.g., sketching.

Chapter 3

Leakage of k-NN Queries in Encrypted Databases

In this chapter we start by performing a theoretical feasibility study on exact reconstruction, i.e., recovery of the exact plaintext values of the encrypted database. For ordered responses, we show that exact reconstruction is feasible if the attacker has additional access to some auxiliary information that is normally not available in practice. For unordered responses, we prove that exact reconstruction is impossible due to the infinite number of valid reconstructions. As a next step, we propose practical and more realistic approximate reconstruction attacks so as to recover an approximation of the plaintext values. For ordered responses, we show that after observing enough query responses, the attacker can approximate the client's encrypted database with considerable accuracy. For unordered responses we characterize the set of valid reconstructions as a convex polytope in a k-dimensional space and present a rigorous attack that reconstructs the plaintext database with bounded approximation error. As multidimensional spatial data can be efficiently processed by mapping it to one dimension via Hilbert curves, we demonstrate our approximate reconstruction attacks on privacy-sensitive geolocation data. Our experiments on real world datasets show that our attacks reconstruct the plaintext values with relative error ranging from 2.9% to 0.003%.

3.1 Preliminaries

Database and its Organization. A database is a collection DB of n records. Let $\alpha, \beta \in \mathbb{R}$. We consider records with one-dimensional values in the continuous range $[\alpha, \beta]$ on which one-dimensional k-nearest neighbor (k-NN) queries are performed. Thus, each record has two fields: (1) a unique identifier, id_i; and (2) a value $\mathbf{val}(id_i) \in [\alpha, \beta]$, for $i = 0, \cdots, n-1$. We denote with $S = (s_0, \ldots, s_{n-1})$ the sequence of record ids sorted in increasing order with respect to their values. Also, we write $v_i = \mathbf{val}(s_i)$. We denote with $\mathsf{pos}(id_i)$ the position of record id_i in sequence S. Finally we assume that a database responds to a k-NN for a fixed k decided at setup-time. For the sake of simplicity of the analysis, we assume that the mapping from records to values is injective, that is, there is a single record in the database associated with a value.

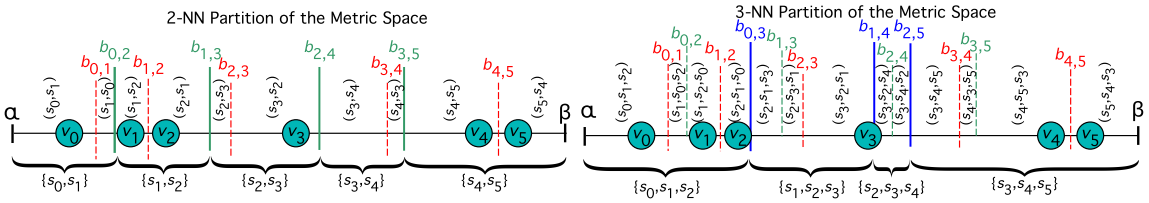


Figure 3.1: The partition of $[\alpha, \beta]$ in Voronoi segments of ordered and unordered responses for $k = 2$ (left) and $k = 3$ (right). The curly brackets on the bottom indicate the *unordered responses* that correspond to each Voronoi segments. The vertically written k-tuples indicate the *ordered responses*. The term $b_{i,j}$ denotes the bisector between v_i and v_j which is also the Voronoi endpoint that separates the corresponding neighboring Voronoi segments.

High-Order Voronoi Diagrams in One-Dimension. Given two values v_i and v_j of database DB, the *bisector* $b_{i,j}$ of v_i and v_j is the point $(v_i + v_j)/2$. For a value v_i of DB, the locus of points of $[\alpha, \beta]$ for which v_i is the nearest neighbor among the values of DB is called the *Voronoi segment* of v_i, denoted $V(v_i)$. The endpoints of $V(v_i)$ are $b_{i-1,i}$ and $b_{i,i+1}$, where we conventionally define $b_{-1,0} = \alpha$ and $b_{n-1,n} = \beta$. The *Voronoi diagram* $V(DB)$ is the partition of range $[\alpha, \beta]$ into regions associated with the Voronoi segments of DB.

The notions of Voronoi segment and Voronoi diagram can be *extended* to sets and tuples of values in DB. Given a *set* H of k values, we define Voronoi segment $V_k(H)$ as the locus of points for which the k-nearest values of every query that lands in this segment comprise set H. If H is a *tuple* of k values, we define the Voronoi segment $V_k(H)$ as the locus of points whose k-nearest values sorted from closest to furthest comprise the tuple H. Thus, for a query that lands in the a locus $V_k(H)$ the server returns the corresponding identifiers of the values of H. We define the *Voronoi diagram of*

order k of DB, denoted with $V_k(DB)$, as the collection of all nonempty Voronoi segments $V_k(H)$, for all k-sized subsets (or tuples) H of values in DB. Finally, we denote with R the set of all possible responses for k-NN queries on DB.

k-NN Responses. We consider two variants of k-NN queries. If the returned response is a *set*, then we have an *unordered response*, denoted with r, which does not differentiate between closeness among the values of r to the query point. In case the response is a *k-tuple* where the order of the components indicates the closeness to the query point (from closest to furthest), then we have an *ordered response*. In our analysis, both type of responses are denoted with r and the exact meaning is either explicitly stated or can be inferred from the context. Figure 3.1 illustrates Voronoi segments for ordered and unordered responses on a database. In our analysis we consider k that takes values from the following range: $2 \leq k \leq \lceil \frac{n}{2} \rceil$. In case $k = 1$ it is not possible to reconstruct the order of the record identifiers due to absence of overlap in the responses. In case $k \geq \lceil \frac{n}{2} \rceil + 1$ there is at least one pair of records that appears in *all possible responses*, thus order reconstruction is not possible.

We denote with $\mathsf{Len}(r)$ the length of the Voronoi segment $V_k(r)$ associated with response r. For the case of unordered responses the set of Voronoi endpoints of $V_k(DB)$ is $\{b_{0,k}, b_{1,k+1}, \ldots, b_{n-k-1,n-1}\}$. The above set of bisectors is also denoted as B_k because each bisector refers to values that are k-positions apart wrt the ordering of S. For the case of ordered responses, the set of Voronoi endpoints of $V_k(DB)$ consists of the union of the sets of bisectors $B_1, B_2, \ldots, B_k$.

In Sections 3.2.2 through 3.2.4 and 3.3.1 we study attacks on ordered responses and in Sections 3.2.5 and 3.3.2 through 3.3.4 we study attacks on unordered responses.

Adversarial Model. In our analysis, we assume that the adversary is *passive* and *persistent*, that is, the adversary sees all the communication between the client and the server. The goal of the adversary is to reconstruct the plaintext value of each record of the encrypted database by just observing the encrypted identifiers returned as responses to k-NN queries. If the attacker recovers the exact values, then the attack is called *exact reconstruction*. If the attacker recovers an approximation of the values, then the attack is called *approximate reconstruction* and in our work is accompanied by rigorous approximation guarantees. Our adversary does not have the power to issue queries or inject data and has no prior knowledge about the distribution of the data.

Leakage Profile Under Attack. To design generic attacks that are applicable to a family of present solutions, e.g. [101, 154], for k-NN queries, we consider a leakage profile that is typical in this line of work. Given a fixed k the *only* information that our adversary sees is the *query*

leakage $\mathcal{L}_Q(DB)$ which is either the set (unordered) or the k-tuple (ordered) of the deterministically encrypted identifiers that are retrieved for an issued query. For simplicity in the rest of the chapter we refer to the deterministically encrypted identifiers as 'records'. The only setup leakage $\mathcal{L}_S(DB)$ that we assume is the *number* of encrypted records, n. We note here that leaking the encrypted record ids returned as responses to queries is a standard approach in the vast majority of encrypted search constructions [23, 36, 45, 63, 90, 96, 101, 138, 149, 154] and to the best of our knowledge, it can only be avoided with heavier cryptographic primitives such as ORAM [75] and response-hiding STE [41], which negatively affect the running time and storage of the overall construction. From this leakage profile the attacker can detect which ids correspond to the two extreme values but it is not possible to differentiate between the ids of the first and the last value. Thus, all our reconstructions, similarly to [99, 106], are *correct up to reflection*.

Assumptions for Our Attacks. For our attacks, we have three assumptions:

1. The queries observed are generated uniformly at random.
2. The database is static, no data is updated after the setup.
3. The boundaries α and β of the values are known.

Assumption 1, uniform query generation, appears in other leakage-abuse attacks [99, 106] and is crucial for our proposed estimation techniques. An application where assumption 2 holds is the historical geo-location trace of a user for a fixed time period, similar to the dataset in our evaluation.

Access to Auxiliary Information. In Section 3.2, we show that an attacker who has additional knowledge can achieve *exact reconstruction*. In particular, for the results of Section 3.2, the adversary is given the following *auxiliary information*, Aux:

- The set of *all possible ordered (resp. unordered) responses* to k-NN queries on DB, denoted with R.
- The *exact length of the Voronoi segment* for every response in R, with is modeled by oracle access to function $\mathsf{Len}(r)$ for a response r in R.

Note that set R has size $n - k + 1$ for unordered responses. The following lemma shows that set R has size $k(n - (k+1)/2) + 1$ for ordered responses.

Lemma 7. *The number of ordered responses for k-NN queries in a database DB with n unique values is $k(n - (k+1)/2) + 1$.*

Proof. We start the proof with a relation regarding the location of bisectors with different gap. The bisector $b_{i,i+(\lambda-1)}$ concerns the ids s_i and $s_{i+(\lambda-1)}$ and the gap is $\lambda-1$, similarly the bisector $b_{i,i+\lambda}$ concerns the ids s_i and $s_{i+\lambda}$. Therefore we have:

$$v_{i+(\lambda-1)} < v_{i+\lambda} \Rightarrow \frac{v_i + v_{i+(\lambda-1)}}{2} < \frac{v_i + v_{i+\lambda}}{2} \Rightarrow b_{i,i+(\lambda-1)} < b_{i,i+\lambda}$$

, where the first inequality comes from the definition of the sequence of sorted ids S. With a similar argument we have:

$$v_i < v_{i+1} \Rightarrow \frac{v_i + v_{i+\lambda}}{2} < \frac{v_{i+1} + v_{i+\lambda}}{2} \Rightarrow b_{i,i+\lambda} < b_{i+1,i+\lambda}$$

Therefore we have

$$b_{i,i+(\lambda-1)} < b_{i,i+\lambda} < b_{i+1,i+1+(\lambda-1)} \tag{3.1}$$

, where $0 \leq i \leq n-k-1$, and $1 < \lambda \leq k$.

Let us start with the partition that is implied by introducing the bisectors of gap k, i.e., $b_{0,k}, b_{1,1+k}, \ldots, d_{n-1-k,n-1}$. These bisectors form a partition of the universe of possible values, furthermore this partition has $n-k+1$ segments. The above $n-k+1$ segments are further divided by the introduction of the bisectors of gap $k-1$, i.e., $b_{0,k-1}, b_{1,1+(k-1)}, b_{2,2+(k-1)}, \ldots, b_{n-1+(k-1)+1,n-1}$.

If we apply the Equation (3.1) to the possible values of λ, i.e., $1 < \lambda \leq k$, we get the following set of inequalities:

$$b_{0,k-1} \leq b_{0,k} \leq b_{1,1+(k-1)} \leq b_{1,1+k} \leq b_{2,2+(k-1)} \leq \ldots \leq b_{n-1-(k-1),n-1}$$

$$b_{0,k-2} \leq b_{0,0+(k-1)} \leq b_{1,1+(k-2)} \leq b_{1,1+(k-1)} \leq b_{2,2+(k-2)} \leq \ldots \leq b_{n-1-(k-2),n-1}$$

$$\vdots$$

$$b_{0,1} \leq b_{0,2} \leq b_{1,2} \leq b_{1,3} \leq b_{2,3} \leq \ldots \leq b_{n-2,n-1}.$$

Thus, when we insert the set of bisectors with gap $\lambda-1$ to the already formed partition we introduce new segments. The exact number of new segments is equal to the number of bisectors with gap $\lambda-1$ since every new bisector cuts a segment in two. By applying the above observation to all

bisectors we get:

$$|R| = \#\text{Segments}B_k + \ldots + \#\text{Segments}B_1 = (n-k+1) + (n-(k-1)) + (n-(k-2)) + \ldots + (n-1)$$

$$= (n-k+1) + \sum_{j=1}^{k-1}(n-j) = (n-k+1) + \frac{(k-1)(2n-k)}{2} = k\left(n - \frac{k+1}{2}\right) + 1.$$

□

One might say that knowledge of the above auxiliary information by the attacker is too much to assume. Indeed, the results of Section 3.2 are primarily of *theoretical interest*. Nevertheless, they provide a sufficient condition that makes exact reconstruction feasible. Also, the attack of Section 3.2 can be modified to achieve approximate reconstruction without access to the auxiliary information. Indeed, as we show in Section 3.3 the auxiliary information *can be approximated* by an attacker who observes a sufficently large number of queries. In particular, the attacker can (1) analyze the probability of the event of observing all the possible responses and (2) rigorously estimate the lengths of the Voronoi segments from the frequency of each response.

3.2 Exact Reconstruction

In this section, we consider *exact* reconstruction attacks for k-NN queries on a one-dimensional encrypted database DB. An exact reconstruction attack is one that always and correctly retrieves the values of the underlying encrypted database by just accessing the leakage. We assume that the attacker has access to the auxiliary information, which we recall consists of the set R (all possible responses to k-NN queries) and oracle access to the function $\mathsf{Len}(r)$ that returns the length of the Voronoi segment associated with a response r in R. The auxiliary information subsumes Assumption 1, which is not necessary for the results in this section. However, we still rely on Assumptions 2 (static database), and 3 (knowledge of the range $[\alpha, \beta]$ of database values).

First, in Section 3.2.1, we present an algorithm that reconstructs the order of the records by value given the set of all the possible responses, R, which is part of the auxiliary information. This algorithm only needs unordered responses. Next, we study the complete exact reconstruction attack for two cases: (i) ordered responses, for which we present an exact reconstruction attack (Sections 3.2.2 through 3.2.4); (ii) unordered responses for which we show that exact reconstruction

is impossible under this leakage profile (Section 3.2.5). The following two theorems summarize the findings of this section.

Theorem 2. *Let DB be an encrypted database consisting of n records with values in the range* $[\alpha, \beta]$. *Assume the adversary is given the set R of all possible ordered responses to k-NN queries and oracle access to the length* $\mathsf{Len}(r)$ *of the Voronoi segment of each response r in R. Algorithm* AttackOrdered *achieves exact reconstruction of the values of DB, up to reflection, in* $O(k\,n \log n)$ *time.*

Theorem 3. *Let DB be an encrypted database with n records, and let* $k \geq 2$. *Given only the leakage of unordered responses to k-NN queries, it is impossible for any attacker (even computationally unbounded) to achieve exact reconstruction.*

Proof. Let DB have n values $v_0, v_1, \ldots, v_{n-1}$. Recall that in the case of k-NN unordered responses, the bisectors that define the Voronoi segments are $b_{0,k}$, $b_{1,k+1}$, $b_{2,k+2}$ up to $b_{n-1-k,n-1}$. To construct DB' we will change the values v_i in a way that the bisectors stay put. This implies that Voronoi segments will not change. Therefore the leakage $\mathcal{L}_Q(DB) = \mathcal{L}_Q(DB')$ while $DB \neq DB'$. To compute DB', we set $v'_0 = v_0 + \epsilon$ and $v'_k = v_k - \epsilon$, thus not affecting bisector $b_{0,k}$. Changing v'_k by $-\epsilon$, however, requires a change on v'_{2k} by $+\epsilon$ so that bisector $b_{k,2k}$ is not affected either. This cascading effect continues until we finally adjust $v'_{\lfloor n/k \rfloor k}$. Note that the range of values that ϵ can take is easily computable. Specifically, it is the range from 0 to the minimum distance between any v_i and v_{i+1} or v_{i-1} for all $i = 0, k, 2k, \ldots, \lfloor n/k \rfloor k$. Since there are arbitrarily many values in this range, we have an arbitrarily number of potential DB' with the same Voronoi diagram.

□

3.2.1 Reconstructing the Order of Records

Consider a database that consists of three points x, y, z and where the set of possible unordered responses to 2-NN queries is $R = \{\{x, z\}, \{y, x\}\}$. Clearly, the only possible order is $z < x < y$ (up to reflection) since x appears in both responses, i.e., overlaps, and thus x is the intermediate value. Our algorithm ReconstructOrder is a generalization of the above idea.

In particular, Algorithm 3 initially finds the identifiers for the largest and smallest values—this is easy since these are the only ones appearing in a *single* k-NN response. Then we construct the order sequence S by finding the response r that overlaps with the $k-1$ most-recently discovered entries of

S, denoted in the algorithm as seq. The single remaining identifier is the one that finally extends the discovered S. See Algorithm 3 for the detailed pseudocode.

Algorithm 3: ReconstructOrder

Input: Set R of unordered responses
Output: Sequence of ordered records $(s_0, \ldots, s_{n-1})$
1 Let Responses$[j]$ be the set of responses containing identifier j;
2 Let id', id'' be the identifiers that are part of *only one* response in R;
3 Set $s_0 \leftarrow id'$ and $s_{n-1} \leftarrow id''$;
4 **for all** $p_i \in$ Responses$[s_0] - \{s_0\}$ **do**
5 $\quad$ ind $\leftarrow |$Responses$[p_i]|$;
6 $\quad$ $s_{\text{ind}} \leftarrow p_i$;
7 **end**
8 **while** $k - 1 \leq$ ind $< n - 2$ **do**
9 $\quad$ seq $\leftarrow \{s_{\text{ind}-k+1}, \ldots, s_{\text{ind}}\}$;
10 $\quad$ Find response r from Responses$[s_{\text{ind}}]$ s.t. $|r \cap \text{seq}| = k - 1$;
11 $\quad$ $s_{\text{ind}+1} \leftarrow r -$ seq;
12 $\quad$ ind$\leftarrow$ ind $+1$;
13 **end**
14 **return** $(s_0, \ldots, s_{n-1})$

Theorem 4. *Given the set R of all possible unordered responses to k-NN queries on an encrypted database DB with n records, Algorithm* ReconstructOrder *computes the order of the records of DB with respect to their values, up to reflection, in time $O(k^2\, n)$.*

Proof. The correctness of the algorithm is straight-forward. As for the complexity, assuming that the data structure Responses is given as an input then Step 2 takes time equal to a linear scan of the DS, i.e., $O(n)$. The 'for' loop of Step 4 runs $k - 1$ times, inside the loop we compute the size of the set Responses$[p_i]$ which naively takes $O(k)$ so overall $O(k^2)$. The 'while' loop of Step 8 is repeated $O(n)$ times and inside the loop we perform the following: Step 9 is $O(k)$, Step 10 needs to go through the k unordered responses of Responses$[s_{\text{ind}}]$ and compute the intersection, an action that takes time proportional to the size of the largest set, so k repeats for $O(k)$ intersection gives us $O(k^2)$. Therefore, overall the 'while' loop has time complexity $O(nk^2)$. As a side note, if one wants to build the data structure Responses from scratch then the complexity is a function of m which is the size of the set of unordered responses. □

Prior Work on Order Reconstruction. The work of Lacharité *et al.* [106] also uses order reconstruction as a step for their attack on range queries leakage. In particular, the "sorting step" proposed in [106] can be directly applied to the case of k-NN queries[1]. But just this step in [106] takes $O(k\, n^3)$ time whereas our algorithm takes $O(k^2\, n)$ time *overall*.

[1]Specifically, Lines 9-15 of Algorithm 2 in [106] iteratively build a set of responses that covers the entire set of records except a single record.

3.2.2 Overview of the Attack For Ordered Responses

Our proposed attack reconstructs the Voronoi diagram of the database values as an intermediate step. This task consists of finding the order of the Voronoi segments and finding the location of the Voronoi endpoints that separate the segments. As we will see, this is enough for total reconstruction. Our attack consists of five steps, which are illustrated in Figure 3.2.

Step-1: Reconstruct Order of Records and Relabel. We find the order of the records *with respect to their corresponding (unknown) values* by executing Algorithm ReconstructOrder, presented in Section 3.2.1. This algorithm takes as input unordered responses, thus ignoring the order of the ids in the response tuples. The output of this step is the n-tuple of ids of DB sorted by value, denoted $S = (s_0, \ldots, s_{n-1})$.

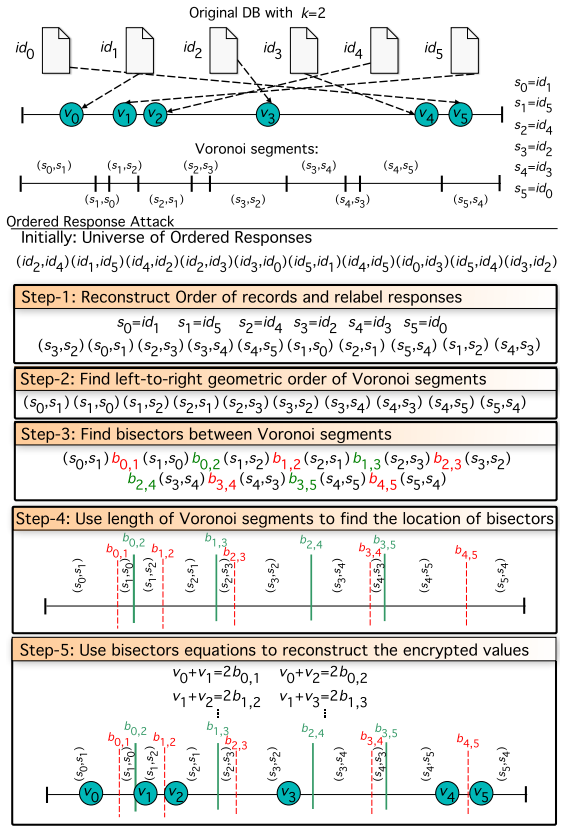


Figure 3.2: An overview of the attack based on *ordered responses* where $k = 2$.

Step-2: Find Left-to-Right Geometric Order of Voronoi Segments. We sort lexicographically the response tuples of R using the order S from the previous step. As shown in Lemma 8, the resulting sorted sequence of responses yields the left-to-right geometric order of the Voronoi segments.

Step-3: Find Bisectors Between Voronoi Segments. By definition, except for α and β, each endpoint of a Voronoi segment is a bisector of two values from DB. In the previous step, we discovered the *neighboring relation* between Voronoi segments, in this step, we further discover *which bisector corresponds to which Voronoi segment endpoint.* Towards this goal, we use Lemma 9, which shows that by comparing the ordered responses of two neighboring Voronoi segments, we can infer which bisector separates them.

Step-4: Use Voronoi Segments' Length to Find the Location of Bisectors. Starting

from α, we use the left-to-right order of the Voronoi segments, and "expand" each segment by its length so as to find the exact location of each bisector.

Step-5: Use Bisector Equations to Reconstruct Encrypted Values. At this point, we have reconstructed the exact Voronoi diagram. In the final step of the attack, we take advantage of the fact that bisectors impose *constraints* on the location of the associated values. Specifically, by the definition of the bisector, the following equality holds $b_{i,j} = (v_i + v_j)/2$. Notice that as long as $k \geq 2$ then the bisectors $B_1 = \{b_{0,1}, b_{1,2}, \ldots, b_{n-2,n-1}\}$ and $B_2 = \{b_{0,2}, b_{1,3}, \ldots, b_{n-3,n-1}\}$ appear as Voronoi endpoints (see Preliminaries). Additionally, the locations of these bisectors are known from the previous steps. Therefore, by forming a system of $|B_1| + |B_2| = 2n - 3$ linear equations with the n unknowns $v_0, \ldots, v_{n-1}$, the adversary reconstructs the encrypted values. Standard algorithms for solving such a system take $O(n^c)$ time, where $c \approx 3$. In Section 3.2.4, we prove that there is a unique solution to this system and by taking advantage of the structure of the equations, we derive a *significantly faster* reconstruction in $O(n)$ time.

3.2.3 Ordering Voronoi Segments and Computing Bisectors

To complete the attack the attacker must order the Voronoi segments and compute the locations of the bisectors separating them. As a reminder, the ordering of the underlying identifiers is derived from Step-1 of the attack.

Lemma 8. *For a database DB with n records, let S be the sequence of identifiers sorted by increasing value. Let R be the universe of all ordered responses for k-NN queries on DB, where each response is a k-tuple of ids of DB. We have that the left-to-right geometric order of the Voronoi segments of the values of DB is given by the lexicographic order of the tuples of R with respect to the ordering of identifiers given by S.*

Finally, two neighboring Voronoi segments are separated by a Voronoi endpoint which is a bisector between two values. The next lemma explains how an attacker can infer *which bisector* separates two neighboring Voronoi segments.

Lemma 9. *Let r_{left} and r_{right} be ordered responses to k-NN queries associated with consecutive Voronoi segments. We have that k-tuples r_{left} and r_{right} differ in either:*

- *the last position, k, where the bisector that separates their segments refers to the values of the record $r_{left}(k)$ and the record $r_{right}(k)$; or*

- *two consecutive positions, l and $l+1$, where the bisector that separates their segments refers to the values of records $r_{left}(l)$ and $r_{left}(l+1)$.*

Proof. Let $\mathbf{val}(r)$ denote the value of record r. Let $h(\mathbf{val}(r(i)), \mathbf{val}(r(j)))$ be the locus of points that are close to $\mathbf{val}(r(i))$ than $\mathbf{val}(r(j))$. Geometrically the Voronoi segment of ordered responses is expressed as:

$$V_k(\mathbf{val}(r)) = \big(\bigcap_{id \in DB - \{r(1)\}} h(\mathbf{val}(r(1)), \mathbf{val}(id)) \big) \cap \dots \tag{3.2}$$

$$\dots \cap \big(\bigcap_{id \in DB - \{r(1),\dots,r(k)\}} h(\mathbf{val}(r(k)), \mathbf{val}(id)) \big). \tag{3.3}$$

Notice that the resulting line segment is the intersection of a series of loci. Since the metric space is bounded, every Voronoi segment is bounded. Specifically, all internal Voronoi segments , i.e., we exclude the first and last, are upper and lower bounded by two bisectors associated with two loci that appear in the definition of $V_k(\mathbf{val}(r))$. The connection between the bisectors and their loci is a fact that we will use later in our proof. From the geometric interpretation of $V_k(\mathbf{val}(r))$ we can easily see that every locus h concerns a pair values where the first value is a record from the ordered response $r = (r(1), \dots, r(k))$ and the second value is from a record id that either appears in r or not. Given that every locus h is defined by a bisector, the above observation can be restated as: the bisectors that upper- and lower-bound $V_k(\mathbf{val}(r))$ are between a record that belongs to the ordered response r and A) a record that does not appear in r or B) a record that appears in r. We proceed with proof by using the above two cases as well as an *inductive argument on the number of bisectors* of a k-th order Voronoi diagram when scanned from left-to-right.

Base Case. Since each record corresponds to a unique value the first bisector, which is also a Voronoi endpoint , that is $b_{0,1}$. Therefore, when scanning from left-to-right the first ordered response is $(s_0, s_1, s_2, \dots, s_{k-1})$ and the second ordered response is $(s_1, s_0, s_2, \dots, s_{k-1})$. The claims of the lemma are true.

Inductive Step. Let's assume that our statement holds for m bisectors, we will prove that the statement holds for $m+1$ bisectors as well. Let r_{left} be the tuple of ordered responses which is lower-bounded by the m-th bisector where the statements of our lemma are true. Let r_{right} be the next tuple of ordered responses that is lower-bounded by the $(m+1)$-th bisector $b_{i,j}$. Thus, bisector

$b_{i,j}$ separates r_{left} from r_{right}. Since the values are unique it follows that we do not have an overlap between two distinct bisectors, which in turn implies that two consecutive Voronoi segments can differ in at most two positions of their tuples. Additionally, one of the values associated with $b_{i,j}$ is part of the left ordered response and the other part of the right, more formally either $s_i \in r_{\text{left}}$ and $s_j \in r_{\text{right}}$ or $s_i \in r_{\text{right}}$ and $s_j \in r_{\text{left}}$. Without loss of generality we proceed assuming $s_i \in r_{\text{left}}$ and $s_j \in r_{\text{right}}$. So the question now is whether s_j is also in r_{left} or not, we proceed with case analysis.

Case $s_j \notin r_{left}$. We know that s_i is part of r_{left} but s_j is not, so there are $k-1$ other ids that appear in r_{left} and together with s_i comprise the ordered response r_{left}. Since the only bisector between the segment of $V_k(\mathbf{val}(r_{\text{left}}))$ and $V_k(\mathbf{val}(r_{\text{right}}))$ is $b_{i,j}$ then the above $k-1$ ids that belong to r_{left} also appear in the same positions in r_{right}. Due to a previous observation we also know that s_j belongs to r_{right}, so s_j together with the $k-1$ ids comprise r_{right}. As a result the two ordered responses only differ in a single position. This is only possible when the two ordered responses differ in the last, i.e., k-th position. Thus, the Voronoi endpoint between them is the bisector of $\mathbf{val}(r_{\text{left}}(k))$ and $\mathbf{val}(r_{\text{right}}(k))$.

Case $s_j \in r_{left}$. We know that both s_i and s_j are part of r_{left}, so there are $k-2$ other ids that appear in r_{left} and together with s_i and s_j comprise the ordered response r_{left}. Since the only bisector between the segment of $V_k(\mathbf{val}(r_{\text{left}}))$ and $V_k(\mathbf{val}(r_{\text{right}}))$ is $b_{i,j}$ then the above $k-2$ ids that belong to r_{left} also appear in the same positions in r_{right}. Due to a previous observation we also know that s_j belongs to r_{right}, so s_j together with the $k-2$ ids appear in r_{right}. The last remaining id from r_{right} can only be s_i. The final piece of the proof is to find the relative order of s_i and s_j in k-tuples r_{left} and r_{right}. Since the line segment $V_k(\mathbf{val}(r_{\text{left}}))$ lies on the side of the locus $h(s_i, s_j)$, we have that the position of s_i in r_{left} is smaller than the position of s_j in r_{left}. A similar argument for the line segment $V_k(\mathbf{val}(r_{\text{right}}))$ shows that the position of s_j in r_{right} is smaller than the position of s_i in r_{right}. Therefore the two ids appear in the same consecutive positions in both r_{left} and r_{right}, let them be l and $l+1$, but in different order. This shows that Voronoi endpoint between them is the bisector of $\mathbf{val}(r_{\text{left}}(l))$ and $\mathbf{val}(r_{\text{left}}(l+1))$ which concludes the proof. □

3.2.4 From Exact Bisectors to Full Reconstruction

Given the length of each Voronoi segment, given via Aux that the attacker has access to in this section, it is easy to compute the exact location of each bisector. In particular, starting from point α, we use the left-to-right order of the Voronoi segments, and "expand" each Voronoi segment by its

length. Since we found which bisector separates which Voronoi segments, we can compute the exact location of every bisector. In the final step of the attack the adversary utilizes the exact locations of bisectors $B_1 = \{b_{0,1}, b_{1,2}, \ldots, b_{n-2,n-1}\}$ $B_2 = \{b_{0,2}, b_{1,3}, \ldots, b_{n-3,n-1}\}$ so as to reconstruct the *exact values* of DB. We use the relation between the bisector and the corresponding values in order to form linear equations where the unknowns are the encrypted values of DB. Specifically, from the locations of the bisectors of set B_1 we can formulate a set of $n-1$ equations, whereas from the locations of the bisectors of set B_2 we can formulate a set of $n-2$ equations. The above two sets of equations are labeled as $L_1 \ldots, L_{2n-3}$ and are depicted in the following:

$$\begin{aligned} L_1 &: v_0 + v_1 = 2b_{0,1} & \qquad L_n &: v_0 + v_2 = 2b_{0,2} \\ L_2 &: v_1 + v_2 = 2b_{1,2} & \qquad L_{n+1} &: v_1 + v_3 = 2b_{1,3} \\ &\vdots & \qquad &\vdots \\ L_{n-1} &: v_{n-2} + v_{n-1} = 2b_{n-2,n-1} & \qquad L_{2n-3} &: v_{n-3} + v_{n-1} = 2b_{n-3,n-1} \end{aligned}$$

Lemma 10. *The above linear system has a unique solution.*

Proof. Using elementary row operations on the augmented matrix we compute the echelon form and thus compute the rank of the matrix. Specifically we introduce two types of echelon transformations:

Transformation (1)	*Transformation (2)*
$L_{n-1+i} - L_i \to L'_{i+2}$	$L_{n-1+i} - L_{i+1} \to L'_i$
$L'_{i+2} + L_{i+1} \to L'_{i+2}$	$L'_i + L_i \to L'_i$
$\frac{1}{2}L'_{i+2} \to L'_{i+2}$	$\frac{1}{2}L'_i \to L'_i$

where Transformation (1) takes values $1 \leq i \leq (n-2)$, and Transformation (2) takes values $1 \leq i \leq 2$. For example, for Transformation (1) and $i = 1$ we have: $L_n - L_1 \to L'_3, L'_3 + L_2 \to L'_3, \frac{1}{2}L'_3 \to L'_3$. By applying the above two transformations to all possible values of i we can construct the new set of linear equations, $L'_1, L'_2, \ldots, L'_n$ and get the following augmented matrix.

$$\begin{array}{c} L'_1: \\ L'_2: \\ L'_3: \\ \ldots \\ L'_n: \end{array} \left[\begin{array}{cccccc|c} 1 & 0 & 0 & \ldots & 0 & 0 & b_{0,2} - b_{1,2} + b_{0,1} \\ 0 & 1 & 0 & \ldots & 0 & 0 & b_{1,3} - b_{2,3} + b_{1,2} \\ 0 & 0 & 1 & \ldots & 0 & 0 & b_{0,2} - b_{0,1} + b_{1,2} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 0 & 1 & b_{n-3,n-1} - b_{n-3,n-2} + b_{n-2,n-1} \end{array}\right]$$

From the resulting augmented matrix we can verify that *the rank of the matrix is* n, therefore the derived solution of the overdetermined system is unique. □

Notice that each equation of the derived augmented matrix gives an expression of the corresponding value in terms of three bisectors. For example, $v_0 = b_{0,2} - b_{1,2} + b_{0,1}$ and $v_1 = b_{1,3} - b_{2,3} + b_{1,2}$ etc. As a result in AttackOrdered we don't have to solve the system of linear equations derived by the set of bisectors B_1 and B_2 which would take $O(n^c)$ time, where $c \approx 3$. Instead we use directly the derived formulas to fully reconstruct all values, which requires $O(n)$ time, as it is captured in Lines 22-25 of AttackOrdered. In terms of time complexity, Step 1 takes $O(nk^2)$, Step-2 takes[2] $O(kn\log(n))$, Step-3 & 4 take $O(k^2n)$, and Step-5 takes $O(n)$ time.

Algorithm 4: AttackOrdered

Input: Auxiliary information Aux=(R, Len), where R corresponds to the ordered responses, and $\mathsf{Len} : r \to \mathbb{R}$ is the length function where $r \in R$.
Output: Reconstructed encrypted values $v_0, \ldots, v_{n-1}$

1 $R_{set} \leftarrow$Transform each k-tuple of R to a set of size k; // Step-1
2 $(s_0, \ldots, s_{n-1}) \leftarrow \mathsf{ReconstructOrder}(R_{set})$;
3 Create an empty array VoronoiOrder ; // Step-2
4 Iterate through all $r \in R$ and add each k-tuple $\big(\mathsf{pos}(r(1)), \ldots, \mathsf{pos}(r(k))\big)$ in VoronoiOrder;
5 VoronoiOrder$\leftarrow$ Sort(VoronoiOrder,'ascending');
6 left$\leftarrow$ VoronoiOrder[1] ; // Step-3 & Step-4
7 $current\_r \leftarrow (s_{\mathsf{left}(1)}, s_{\mathsf{left}(2)}, \ldots, s_{\mathsf{left}(k)})$;
8 $covered\_area \leftarrow \alpha + \mathsf{Len}(current\_r)$;
9 **for** ***all*** $2 \le i \le |VoronoiOrder|$ **do**
10 left$\leftarrow$ VoronoiOrder$[i-1]$, right$\leftarrow$ VoronoiOrder$[i]$;
11 **if** *k-tuples left and right differ in only one position* **then**
12 $j \leftarrow \mathsf{left}(k)$;
13 $b_{j,j+k} \leftarrow covered\_area$;
14 **else**
15 Let x be the smallest position left and right differ;
16 $j \leftarrow \mathsf{left}(\mathsf{x})$, $j' \leftarrow \mathsf{left}(\mathsf{x}+1)$;
17 $b_{j,j'} \leftarrow covered\_area$;
18 **end**
19 $current\_r \leftarrow (s_{\mathsf{right}(1)}, s_{\mathsf{right}(2)}, \ldots, s_{\mathsf{right}(k)})$;
20 $covered\_area \leftarrow covered\_area + \mathsf{Len}(current\_r)$;
21 **end**
22 $v_0 \leftarrow b_{0,2} - b_{1,2} + b_{0,1}$, $v_1 \leftarrow b_{1,3} - b_{2,3} + b_{1,2}$; // Step-5
23 **for** ***all*** $2 \le i \le n-1$ **do**
24 $v_i \leftarrow b_{i-2,i} - b_{i-2,i-1} + b_{i-1,i}$;
25 **end**
26 **return** $v_0, \ldots, v_{n-1}$

[2]Since the total number of ordered responses is $k(n-(k+1)/2)+1$ the sorting step of that many items takes $O(kn\log(n))$

3.2.5 Exact Reconstruction Impossibility for Unordered Responses

We sketch here the proof of the impossibility of exact reconstruction for the case of *unordered responses* (Theorem 3). We show that for any fixed $k \geq 2$, there exist arbitrarily many distinct databases with same unordered-responses query leakage, and thus the leakage is not enough to distinguish among them. From the leakage, we can derive the Voronoi diagram and thus the location of all the bisectors. In our proof we demonstrate how to "displace" a carefully chosen subset of values so as to create arbitrarily many distinct databases, one for every possible displacement value, while maintaining the location of the bisectors.

3.3 Approximate Reconstruction

We now turn our attention to attacks that **approximate** the plaintext values when there is *no guarantee* that all possible responses are observed by the adversary and the exact Voronoi segment lengths are *not available*, i.e., *no auxiliary information*. Again, we consider ordered and unordered responses. In both cases, our approximate reconstruction fails if the adversary has not observed the set of all possible responses, R. The probability of this happening (over m uniformly distributed queries) is summarized in the following lemma.

Lemma 11. *The probability that the set of responses to m uniform k-NN queries from $[\alpha, \beta]$ does not contain the set of all possible ordered (unordered) responses, R, is at most*

$$|R| e^{-\frac{m}{\beta-\alpha} \min_{r \in R} \mathsf{Len}(r)},$$

where $\mathsf{Len}(r)$ is the length of the Voronoi segment of r.

Proof. Let Z be a random variable defined to be the number of queries required to see at least one *of each* Voronoi segments. Let p_i be the probability that i-th leftmost Voronoi segment is observed by a uniformly generated query. Let E_i^l be the even that the i-th leftmost Voronoi segment (wrt to their left-to-right order) is not observed in the first l queries.

$$\Pr(E_i^l) = (1 - p_i)^l \leq e^{-p_i l}$$

Using the union bound we get:

$$\Pr(Z > m) = \Pr\left(\cup_{i=1}^{|R|} E_i^m\right) \leq \sum_{i=1}^{|R|} \Pr(E_i^m) \leq \sum_{i=1}^{|R|} e^{-p_i m} \leq \sum_{i=1}^{|R|} e^{-m \cdot \min_{1 \leq j \leq |R|} p_j}$$
$$= |R| e^{-m \cdot \min_{1 \leq j \leq |R|} p_j} = |R| e^{-\frac{m}{\beta - \alpha} \min_{r \in R} \mathsf{Len}(r)}$$

Thus, $\Pr(Z < m) \geq 1 - |R| e^{-\frac{m}{\beta - \alpha} \min_{r \in R} \mathsf{Len}(r)}$ □

With reference to Lemma 11, recall that the size of the set R of all possible responses to k-NN queries on a database with n records is $|R| = k(n - (k+1)/2) + 1$ for ordered responses and $|R| = n - k + 1$ for for unordered responses. The attacker can verify whether all responses are observed since we know n from the setup leakage and k from the query leakage. Note that for a fixed number of queries, the smaller the length of the minimum Voronoi segment, the larger a probability of failure of the attacks. Namely, our approximate reconstruction attack fails with the probability given in Lemma 11 due to not having observed all the responses. However, for unordered responses, the attack can fail for another reason as well. In particular, as discussed in Section 3.3.4, the attacker picks its output based on an estimated k-dimensional polytope. Thus, if the estimated polytope is *empty*, the attack fails.

3.3.1 Ordered Responses: Estimating Voronoi Segment Lengths

Given all possible responses R have been observed with m uniformly generated queries in $[\alpha, \beta]$, our approximate reconstruction attack is a simple modification of attack AttackOrdered presented in the Section 3.2.4. In particular, instead of assuming oracle access to function $\mathsf{Len}(r)$ at Line 20 of AttackOrdered, we estimate $\mathsf{Len}(r)$ as

$$(\beta - \alpha) \cdot \frac{m_r}{m}, \tag{3.4}$$

where m_r is the number of queries (out of m total queries) that returned r as a response. The resulting reconstruction attack achieves *approximate reconstruction (up to reflection) with rigorous guarantees*:

Theorem 5. *Let DB be an encrypted database with n records whose values are in the range $[\alpha, \beta]$. Suppose the attacker observes the responses to m k-NN queries that are uniformly generated from $[\alpha, \beta]$ (Assumption 1) and which contain all possible ordered responses, R. For any $0 < \epsilon < \beta - \alpha$*

and $0 < \delta < 1$*, the variation of Algorithm* AttackOrdered *which estimates Voronoi segment lengths using Equation 3.4 computes in* $O(m + k\,n \log n)$ *time a sequence of reconstructed values such that each reconstructed value differs from its original value by at most* ϵ *with probability at least* $1 - \delta$*, provided* m *is at least*

$$\max\left\{\frac{180(\beta-\alpha)^2(k(n-(k+1)/2)+1)}{\epsilon^2},,\frac{225(\beta-\alpha)^2(\ln 3-\ln\delta)}{\epsilon^2}\right\}.$$

Proof. Let $(Y_1, \ldots, Y_{|R|})$ be a multinomial distribution, that is $|R|$ is a fixed number and we have $|R|$ mutually exclusive outcomes with corresponding probabilities $p_1, \ldots, p_{|R|}$, and m independent trials. Due to the fact that the $|R|$ outcomes are mutually exclusive and one must occur every probability is non-zero and $\sum_{i=1}^{|R|} p_i = 1$. Additionally, the expectation of every Y_i is $E[Y_i] = mp_i$. From Lemma 3 in [56] we have that for all $\epsilon_1 \in (0, 1)$ and all R satisfying $|R|/m \leq \epsilon_1^2/20$ we have:

$$\Pr\big(\sum_{i=1}^{|R|}|Y_i - E[Y_i]| > m\epsilon_1) \leq 3e^{-m\epsilon_1^2/25} \Rightarrow \Pr\big(\sum_{i=1}^{|R|}|Y_i - E[Y_i]| \leq m\epsilon_1) > 1 - 3e^{-m\epsilon_1^2/25}.$$

Let us now analyze the above probability event:

$$\sum_{i=1}^{|R|}|Y_i - E[Y_i]| \leq m\epsilon_1 \Rightarrow |\sum_{i=1}^{|R|}(Y_i - E[Y_i])| \leq \sum_{i=1}^{|R|}|Y_i - E[Y_i]| \leq m\epsilon_1$$

$$\Rightarrow -m\epsilon_1 \leq \sum_{i=1}^{|R|}\big(Y_i - E[Y_i]\big) \leq m\epsilon_1 \Rightarrow \big(\sum_{i=1}^{|R|}\frac{E[Y_i]}{m}\big) - \epsilon_1 \leq \sum_{i=1}^{|R|}\frac{Y_i}{m} \leq \big(\sum_{i=1}^{|R|}\frac{E[Y_i]}{m}\big) + \epsilon_1 \tag{3.5}$$

Transformation to Bisector Estimation. The multinomial formulation can be adjusted to assist to the task of bisector approximation. Let each trial correspond to a uniformly chosen query point from $[\alpha, \beta]$, then the outcome of the trial corresponds to an ordered response from the Voronoi diagram $V_k(DB)$. The number of trials for the case of a multinomial was denoted by m which in the bisector estimation is the number of queries. Therefore the number of possible outcomes $|R|$ is the number of Voronoi segments of order k. Notice that the probability that a response is r_i is equal to the probability p_i that a uniformly chosen query point lands to the Voronoi segment $V_k(r_i)$, which in turn is equal to the ratio of the length of the corresponding Voronoi segment to the length of the entire bounded metric space. Specifically $p_i = \mathsf{Len}(V_k(r_i))/(\beta - \alpha)$.

Let $P_{i,j}$ be the set of Voronoi segments that precede $b_{i,j}$, then we know that $b_{i,j} = \alpha +$

$\sum_{l\in P_{i,j}} \mathsf{Len}(P_{i,j}(l))$. Since $|P_{i,j}| < R$ we have:

$$|\sum_{l\in P_{i,j}} (Y_i - E[Y_i])| \leq \sum_{l\in P_{i,j}} |Y_l - E[Y_l]| \leq \sum_{i=1}^{R} |Y_i - E[Y_i]| \leq m\epsilon_1$$

, and similarly to the summation of (3.5) we can derive:

$$(\sum_{l\in P_{i,j}} \frac{E[Y_l]}{m}) - \epsilon_1 \leq \sum_{l\in P_{i,j}} \frac{Y_l}{m} \leq (\sum_{l\in P_{i,j}} \frac{E[Y_l]}{m}) + \epsilon_1 \Rightarrow (\sum_{l\in P_{i,j}} p_l) - \epsilon_1 \leq \sum_{l\in P_{i,j}} \frac{Y_l}{m} \leq (\sum_{l\in P_{i,j}} p_l) + \epsilon_1$$

$$\Rightarrow (\sum_{l\in P_{i,j}} \frac{\mathsf{Len}(P_{i,j}(l))}{(\beta-\alpha)}) - \epsilon_1 \leq \sum_{l\in P_{i,j}} \frac{Y_l}{m} \leq (\sum_{l\in P_{i,j}} \frac{\mathsf{Len}(P_{i,j}(l))}{(\beta-\alpha)}) + \epsilon_1$$

$$\Rightarrow \big(\sum_{l\in P_{i,j}} \mathsf{Len}(P_{i,j}(l))\big) - (\beta-\alpha)\epsilon_1 \leq (\beta-\alpha)\sum_{l\in P_{i,j}} \frac{Y_l}{m} \leq \big(\sum_{l\in P_{i,j}} \mathsf{Len}(P_{i,j}(l))\big) + (\beta-\alpha)\epsilon_1$$

$$\Rightarrow b_{i,j} - (\beta-\alpha)\epsilon_1 \leq \alpha + (\beta-\alpha)\sum_{l\in P_{i,j}} \frac{Y_l}{m} \leq b_{i,j} + (\beta-\alpha)\epsilon_1$$

$$\Rightarrow b_{i,j} - (\beta-\alpha)\epsilon_1 \leq \widetilde{b_{i,j}} \leq b_{i,j} + (\beta-\alpha)\epsilon_1 \Rightarrow |b_{i,j} - \widetilde{b_{i,j}}| \leq (\beta-\alpha)\epsilon_1$$

Therefore the overall probability expression becomes:

$$\Pr\left(|b_{i,j} - \widetilde{b_{i,j}}| \leq (\beta-\alpha)\epsilon_1\right) > 1 - 3e^{-m\epsilon_1^2/25}.$$

We define $\epsilon_2 = (\beta-\alpha)\epsilon_1$ and get:

$$\Pr\left(|b_{i,j} - \widetilde{b_{i,j}}| \leq \epsilon_2\right) > 1 - 3e^{-\frac{m\epsilon_2^2}{(\beta-\alpha)^2 25}}.$$

With further analysis we get:

$$\left.\begin{aligned} b_{i-2,i} - \epsilon_2 \leq \widetilde{b_{i-2,i}} \leq b_{i-2,i} + \epsilon_2 \\ b_{i-2,i-1} - \epsilon_2 \leq \widetilde{b_{i-2,i-1}} \leq b_{i-2,i-1} + \epsilon_2 \\ -b_{i-1,i} - \epsilon_2 \leq -\widetilde{b_{i-1,i}} \leq -b_{i-1,i} + \epsilon_2 \end{aligned}\right\}$$

$$\Rightarrow b_{i-2,i} + b_{i-2,i-1} - b_{i-1,i} - 3\epsilon_2 \leq \widetilde{b_{i-2,i}} + \widetilde{b_{i-2,i-1}} - \widetilde{b_{i-1,i}} \leq b_{i-2,i} + b_{i-2,i-1} - b_{i-1,i} + 3\epsilon_2$$

$$\Rightarrow v_i - 3\epsilon_2 \leq \widetilde{b_{i-2,i}} + \widetilde{b_{i-2,i-1}} - \widetilde{b_{i-1,i}} \leq v_i + 3\epsilon_2 \Rightarrow v_i - 3\epsilon_2 \leq \widetilde{v_i} \leq v_i + 3\epsilon_2$$

, where for the last substitution we used the augmented matrix described in Section 3.2.4. Finally, we define $\epsilon = 3\epsilon_2$. We also define δ as $\delta \geq 3e^{-\frac{m\epsilon_2^2}{25(\beta-\alpha)^2}}$ and get:

$$\delta \geq 3e^{-\frac{m\epsilon_2^2}{25(\beta-\alpha)^2}} = 3e^{-\frac{m\epsilon^2}{225(\beta-\alpha)^2}} \Rightarrow \ln\delta \geq \ln 3 - \frac{m\epsilon^2}{225(\beta-\alpha)^2} \Rightarrow m \geq \frac{225(\ln 3 - \ln\delta)(\beta-\alpha)^2}{\epsilon^2}$$

Also with some algebraic manipulation of the inequality about m from Lemma 3 in [56] we get:

$$\frac{|R|}{m} \leq \frac{\epsilon_1^2}{20} \Rightarrow \frac{|R|}{m} \leq \frac{\epsilon_2^2}{20(\beta-\alpha)^2} \Rightarrow m \geq \frac{|R|(\beta-\alpha)^2 20}{\epsilon_2^2} \Rightarrow m \geq \frac{180(\beta-\alpha)^2|R|}{\epsilon^2}$$

Since $|R| = k(n-(k+1)/2)+1$ we get

$$m \geq \frac{180(\beta-\alpha)^2(k(n-(k+1)/2)+1)}{\epsilon^2}.$$

By using the two derived inequalities about m we get:

$$m = \max\left\{\frac{180(\beta-\alpha)^2(k(n-(k+1)/2)+1)}{\epsilon^2}, \frac{225(\beta-\alpha)^2(\ln 3 - \ln\delta)}{\epsilon^2}\right\}$$

Then the out put $\tilde{v}$ of the algorithm satisfies the following probability expression:

$$\Rightarrow \Pr\left(|v_i - \widetilde{v_i}| \leq \epsilon\right) > 1 - \delta.$$

□

3.3.2 Unordered Responses: Defining the Reconstruction Space

As we saw in Section 3.2.5, in the case of unordered responses, there are more than one databases DB that map to the same query leakage $\mathcal{L}_Q(DB)$. Our first step in developing an attack is to study the space of potential reconstructions.

We first define the *universe* of all reconstructions. Let $\mathcal{V}^n$ be the set of n-tuples $v \in \mathcal{V}^n$ such that $v_0, \ldots, v_{n-1} \in [\alpha, \beta]$ and $v_0 < v_1 < \ldots < v_{n-1}$. A reconstruction algorithm returns an n-tuple from the set $\mathcal{V}^n$. We show how $\mathcal{V}^n$ can be partitioned based on the concept of Voronoi diagrams of order k.

Definition 5. *Let $\mathcal{V}^n$ be the set of ordered n-tuples with distinct values from the range $[\alpha, \beta]$. We define the* ***reconstruction relation*** $\mathcal{P}_k$ ***with respect to*** k *as a subset of $\mathcal{V}^n \times \mathcal{V}^n$ such that if $(v, v') \in \mathcal{P}_k$, also denoted as $v \sim_k v'$, then $V_k(v) = V_k(v')$ where $V_k(\cdot)$ is the k-th order Voronoi diagram.*

To put it simply, if we have two n-tuples v, v' where the reconstruction relation $\mathcal{P}_k$ holds then they have the same k-th order Voronoi diagram. A corollary of the above definition is that the reconstruction relation is an *equivalence relation.*

Definition 6. *Given the reconstruction relation $\sim_k$ and $v \in \mathcal{V}^n$, we define $[v]$ the* ***reconstruction class of*** v*, as :*

$$[v] = \{v' \in \mathcal{V}^n : v \sim_k v'\}.$$

It is easy to prove that the reconstruction class of v is also an *equivalence class.* From the properties of equivalence classes we know that every two equivalence classes $[v]$ and $[v']$ are either equal or disjoint. Additionally, the set of all equivalence classes of $\mathcal{V}^n$ forms a partition of $\mathcal{V}^n$, i.e., every n-tuple $v \in \mathcal{V}^n$ belongs to one and only one equivalence class. Given an equivalence class $[v]$, a *representative* for $[v]$ is an n-tuple of $[v]$, i.e., it is a $v' \in \mathcal{V}^n$ such that $v' \sim_k v$.

One of the n-tuples of the reconstruction class is the *original database*, but given the considered leakage profile our attacker *does not know* which one. We focus on attacks that return a representative of the correct[3] reconstruction class. The success of the reconstruction attack is measured using the L_∞ distance metric, also known as Chebyshev distance, so as to capture the largest error among all reconstructed values. More formally:

$$d_{L_\infty}(v, v') = \max_{0 \le i < n}(|v_i - v'_i|).$$

The next step of our analysis is to characterize the n-tuples of a given reconstruction class.

Characterization via Offsets ξ from Bisectors. In order for two n-tuples to be in the same reconstruction class they must have the same k-th order Voronoi diagram. Therefore the Voronoi endpoints (i.e., the bisectors) must be in the same fixed location. In this approach we model the n-tuples of the reconstruction class using *only* k unknown offsets from the above fixed bisectors, as opposed to n unknowns of a naive approach. To give an intuitive explanation of our attack we

[3]Other attack techniques might be possible if the attacker is willing to output an n-tuple from *any* (and possibly incorrect) reconstruction class.

proceed with an illustration of our result for $k = 2$, the general case where $k > 2$ is addressed in Section 3.3.4.

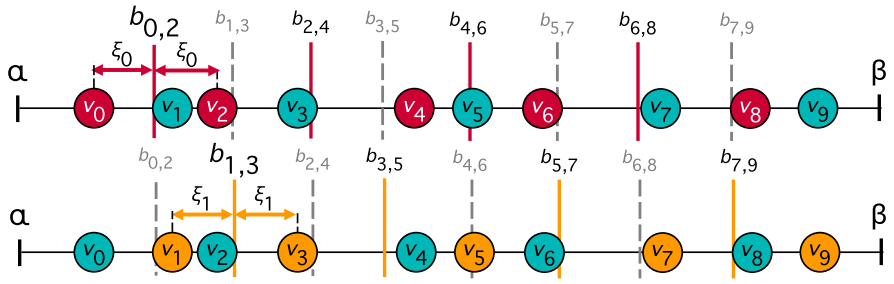


Figure 3.3: Values v_0 and v_2 must be equidistant, specifically ξ_0 afar, from $b_{0,2}$. Using the derived equations we can express the locations of v_2, v_4, v_6, v_8 because the locations of bisectors $b_{0,2}, b_{2,4}, b_{4,6}, b_{6,8}$ stay fixed. Similarly, by using offset ξ_1 we can express the locations of v_3, v_5, v_7, v_9.

Analyzing Case $k = 2$. Our goal is to *characterize the space* of n-tuples of a reconstruction class given its 2nd order Voronoi diagram. We will demonstrate that we need to define unknowns for the location of only two values and the rest of the $n - 2$ values can be expressed as function of these two. The unknown variables are $\xi = (\xi_0, \xi_1)$ and their geometric description follows. Let the first value v_0 be ξ_0 to the left of the bisector $b_{0,2}$. Since the location of $b_{0,2}$ is fixed, it follows that the value v_2 must be ξ_0 to the right of the bisector $b_{0,2}$. Using the formulated equation $v_2 = b_{0,2} + \xi_0$ we can express v_4 so as v_4 and v_2 are equidistant from the fixed location of $b_{2,4}$. Using the same reasoning let v_1 be ξ_1 to the left of the bisector $b_{1,3}$. The location of values $v_3, v_5, v_7, \ldots$ can be expressed as a function of the offset ξ_1 and the location of the relevant bisectors. As one can easily see, by picking a value for the unknown ξ_0 we fix the location of v_0 (resp. v_1) which in turn has *a domino effect* on the location of $v_2, v_4, v_6, \ldots$ (resp. $v_3, v_5, v_7, \ldots$). Figure 3.3 highlights which values can be expressed as a function of the unknown offsets ξ_0, ξ_1. Specifically for $n = 10$ we have:

$$
\begin{aligned}
v_0 &= b_{0,2} - \xi_0 & v_1 &= b_{1,3} - \xi_1 \\
v_2 &= b_{0,2} + \xi_0 & v_3 &= b_{1,3} + \xi_1 \\
v_4 &= 2b_{2,4} - v_2 = 2b_{2,4} - b_{0,2} - \xi_0 & v_5 &= 2b_{3,5} - v_3 = 2b_{3,5} - b_{1,3} - \xi_1 \\
v_6 &= 2b_{4,6} - v_4 = 2b_{4,6} - 2b_{2,4} + b_{0,2} + \xi_0 & v_7 &= 2b_{5,7} - v_5 = 2b_{5,7} - 2b_{3,5} + b_{1,3} + \xi_1 \\
v_8 &= 2b_{6,8} - v_6 = 2b_{6,8} - 2b_{4,6} + 2b_{2,4} - b_{0,2} - \xi_0 & v_9 &= 2b_{7,9} - v_7 = 2b_{7,9} - 2b_{5,7} + 2b_{3,5} - b_{1,3} - \xi_1
\end{aligned}
$$

The next lemma describes the closed-form of each value as function of the appropriate bisectors and offset for any $k \geq 2$.

Lemma 12. *Let $V_k(v)$ be the Voronoi diagram of the reconstruction class $[v]$ with Voronoi endpoints $b_{0,k}, \ldots, b_{n-k-1,n-1}$. If an n-tuple v' belongs to $[v]$, there exists a k-tuple denoted as ξ where $\xi_0, \ldots, \xi_{k-1} \geq 0$ such that for all $0 \leq i \leq n-1$:*

$$v'_i = \begin{cases} b_{i,i+k} - \xi_i & , for\ 0 \leq i < k \\ b_{i \bmod k, i \bmod k + k} + \xi_{i \bmod k} & , for\ k \leq i < 2k \\ (-1)^{\lfloor i/k - 1 \rfloor}(b_{i \bmod k,(i \bmod k)+k} + \xi_{i \bmod k}) + & , for\ 2k \leq i \leq n-1 \\ + \sum_{2 \leq j \leq \lfloor i/k \rfloor} (-1)^{j + \lfloor i/k \rfloor} 2 b_{(i \bmod k)+(j-1)k,(i \bmod k)+jk}, & \end{cases} \tag{3.6}$$

We call the k-tuple $\xi = (\xi_0, \ldots, \xi_{k-1})$ the ***offset vector of*** *v'.*

After characterizing the n-tuples of $[v]$ using 2 unknowns we address the following question: *What values can ξ take so as to give an n-tuple that belongs to $[v]$?*

Ordering Constraints. We define the *ordering constraints* as the inequalities that guarantee that two consecutive values do not surpass each other, e.g., $v_0 \leq v_1$, $v_1 \leq v_2$ etc. By substituting the formulas for $v_0, \ldots, v_{n-1}$ defined in Lemma 12, we get inequalities with the unknowns ξ_0 and ξ_1. As an example, for $n = 10$ we have the following inequalities:

$$\begin{aligned}
&v_0 < v_1 \Rightarrow -\xi_0 + \xi_1 < c_{0,1} \text{ , where } c_{0,1} = (b_{1,3} - b_{0,2}) \\
&v_1 < v_2 \Rightarrow -\xi_0 - \xi_1 < c_{1,2} \text{ , where } c_{1,2} = -(b_{1,3} - b_{0,2}) \\
&v_2 < v_3 \Rightarrow \xi_0 - \xi_1 < c_{2,3} \text{ , where } c_{2,3} = (b_{1,3} - b_{0,2}) \\
&v_3 < v_4 \Rightarrow \xi_0 + \xi_1 < c_{3,4} \text{ , where } c_{3,4} = (b_{2,4} - b_{1,3}) + (b_{2,4} - b_{0,2}) \\
\hline
&v_4 < v_5 \Rightarrow -\xi_0 + \xi_1 < c_{4,5} \text{ , where } c_{4,5} = 2(b_{3,5} - b_{2,4}) - (b_{1,3} - b_{0,2}) \\
&v_5 < v_6 \Rightarrow -\xi_0 - \xi_1 < c_{5,6} \\
&\quad \text{, where } c_{5,6} = 2(b_{4,6} - b_{3,5}) - (b_{2,4} - b_{0,2}) - (b_{2,4} - b_{1,3}) \\
&v_6 < v_7 \Rightarrow \xi_0 - \xi_1 < c_{6,7} \\
&\quad \text{, where } c_{6,7} = 2(b_{5,7} - b_{4,6}) - 2(b_{3,5} - b_{2,4}) + (b_{1,3} - b_{0,2}) \\
&v_7 < v_8 \Rightarrow \xi_0 + \xi_1 < c_{7,8} \\
&\quad \text{, where } c_{7,8} = 2(b_{6,8} - b_{5,7}) - 2(b_{4,6} - b_{3,5}) + (b_{2,4} - b_{1,3}) + (b_{2,4} - b_{0,2}) \\
\hline
&v_8 < v_9 \Rightarrow -\xi_0 + \xi_1 < c_{8,9} \\
&\quad \text{, where } c_{8,9} = 2(b_{7,9} - b_{6,8}) - 2(b_{5,7} - b_{4,6}) + 2(b_{3,5} - b_{2,4}) - (b_{1,3} - b_{0,2})
\end{aligned}$$

By examining the first four inequalities one can see that the first $\xi_1 \leq \xi_0 + c_{0,1}$ and the third $\xi_1 \geq \xi_0 - c_{2,3}$ concern half-planes based on *parallel lines* with positive slope. The y-intercept of the first is positive while the y-intercept of the third is negative. Similarly, the second inequality $\xi_1 \geq -\xi_0 - c_{1,2}$ and the fourth $\xi_1 \leq -\xi_0 + c_{3,4}$, concern half-planes based on parallel lines with negative slope and y-intercepts that are negative and positive, respectively. If we extend this reasoning to the rest of the constraints we see that we can partition the ordering constraints in four categories of half-planes, i.e., 1) *slope* $= 1$ and positive y-intercept, 2) *slope* $= 1$ and negative y-intercept 3) *slope* $= -1$ and positive y-intercept, and 4) *slope* $= -1$ and negative y-intercept. From each of the four categories *all but one constraint are redundant.* By omitting redundant constraints we do not change the region that satisfies the constraints. To find the *non-redundant* one can go through the y-intercepts of each category and remove the overlapping constraints, can be accomplished in time $O(n)$.

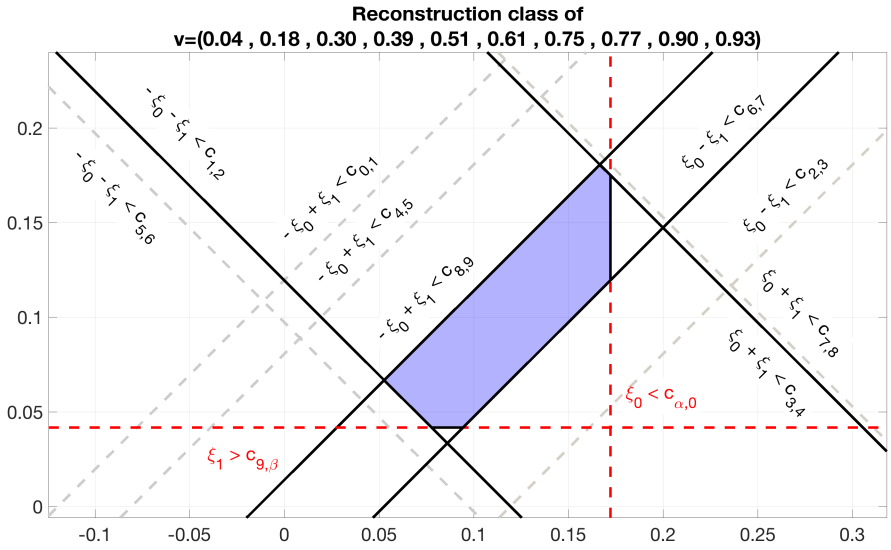


Figure 3.4: The possible values of ξ_0, ξ_1 in X- and Y-axis respectively for a given $[v]$. The feasible region is colored with blue. The set of redundant ordering constraints are depicted with gray dotted-lines, and the non-redundant ordering constraints with black bold-lines. The boundary constraints are depicted with red dotted-lines.

Boundary Constraints. We need two inequalities to guarantee that the first/last value do not surpass the boundary of the range of values, i.e., $[\alpha, \beta]$. For the example where $n = 10$ we have 1) $\alpha < v_0 \Rightarrow \xi_0 < c_{\alpha,0}$ where $c_{\alpha,0} = b_{0,2} - \alpha$, and 2) $v_9 < \beta \Rightarrow \xi_1 > c_{9,\beta}$ where $c_{9,\beta} = 2b_{7,9} - 2b_{5,7} + 2b_{3,5} - b_{1,3} - \beta$.

The Reconstruction Class: A Convex Polygon. The pairs of feasible offset values (ξ_0, ξ_1) is the set

of values that satisfy: 1) the four non-redundant ordering constraints as well as 2) the two boundary constraints. Figure 3.4 gives a detailed geometric illustration of the feasible region for the running example. Depending on the values of the database the boundary constraints can be redundant. Generally, we denote the **feasible region of reconstruction class** $[v]$ as $\mathcal{F}_{[v]} = \{\xi' \in \mathbb{R}^k : A \cdot \xi' \leq c\}$, where each row of $A \cdot \xi' \leq c$ represents a constraint on ξ. Overall, we have:

(1) **Ordering constraints:** $n-1$ in number,

(2) **Boundary constraints:** two in number,

(3) **Positive-offset constraints:** k constraints to guarantee that the offsets are positive.

Therefore, A is a $(n+k+1) \times k$ matrix of coefficients for the inequalities, ξ is a column vector with k offsets, and c is the column vector with the $n+k+1$ constants (i.e., $c_{a,0}, c_{n-1,b}, c_{0,1}, c_{1,2}, \ldots, c_{n-2,n-1}$). Since we only have *linear* inequalities in $\mathcal{F}_{[v]}$, the region is a *convex polytope*.

Diameter of the Feasible Region. Given the feasible region $\mathcal{F}_{[v]}$ of the reconstruction class $[v]$, the L_∞ distance between a pair of n-tuples $v', v'' \in [v]$ of the class is:

$$d_{L_\infty}(v', v'') = \max_{0 \leq i \leq n-1}(\{|v'_i - v''_i|\}) = \max_{0 \leq i \leq k-1}(\{|\xi'_i - \xi''_i|\} = d_{L_\infty}(\xi', \xi'') \leq d_{L_2}(\xi', \xi'') \leq diam(\mathcal{F}_{[v]}) \tag{3.7}$$

where the second equality is derived by substituting the values with the offset formulas of Lemma 12. The *polytope diameter* $diam(\cdot)$, or simply diameter, is the largest Euclidean distance between any pair of vertices of the polytope. Therefore if the attacker is able to compute $\mathcal{F}_{[v]}$ he can compute an upper-bound of the distance of any pair of n-tuples in the reconstruction class. We note here that the final output of the reconstruction attack is a representative v^* of the reconstruction class $[v]$, and that the original database can be *any* n-tuple of the reconstruction class. The last key observation of the attack is that if the attacker outputs v^* for which the offset vector is the *mean* of the offsets ξ', ξ'' of the diameter, then *all* the potential original database n-tuples are at most $diam(\mathcal{F}_{[v]})/2$ distance afar.

3.3.3 Overview of the Unordered Response Attack

We give an overview of the approximate reconstruction for $k = 2$. See Section 3.3.4 for a generalization.

Step 1. The attacker reconstructs the order of the records with respect to their (unknown) values by using the algorithm ReconstructOrder. After relabeling the record ids using $S = (s_0, \ldots, s_{n-1})$

the attacker computes the left-to-right order of the Voronoi segments. For the case of unordered responses this step is straightforward and can be done by just "shifting" a k-length window over the sequence S, e.g., the left-to-right order is $\{s_0, \ldots, s_{k-1}\}, \{s_1, \ldots, s_k\}, \ldots, \{s_{n-k-1}, \ldots, s_{n-1}\}$. For the case of unordered responses assigning the bisectors to Voronoi endpoints is straightforward as well. The corresponding left-to-right order of the bisectors is $b_{0,k}, b_{1,k+1}, \ldots, b_{n-k-1,n-1}$. This attack differs significantly from the the Ordered Responses Attack in the next two steps.

Step 2: Estimate the Constraints of the Feasible Region. There are infinitely many value n-tuples for DB that can give a fixed k-th order Voronoi diagram. The next step of our attack characterizes the set of *all such n-tuples* using only k unknowns, namely the offsets $\xi = (\xi_0, \ldots, \xi_{k-1})$. We define a set of linear constraints, namely the ordering, the boundary, and the positive-offset constraints, imposed on the unknowns ξ so as to find the offsets assignments that correspond to a valid n-tuple of the reconstruction class. Each constraint imposed on ξ is a half-space and the intersection of these constraints defines the *feasible region* $\mathcal{F}_{[v]}$. Geometrically, $\mathcal{F}_{[v]}$ is a bounded convex k-dimensional polyhedron, i.e., a polytope. But since the constraints' constants *are not known to the attacker* we propose a way to estimate them. In particular the new algorithm estimates the right-hand side constant of each constraint, e.g., estimation of terms $c_{\alpha,0}, c_{0,1}, \ldots, c_{4,5}, c_{5,\beta}$ in Figure 3.5.

The key observation is that each c term can be expressed as the *linear combination of lengths* of Voronoi segments, e.g., in Figure 3.5 term $c_{4,5}$ involves $\mathsf{Len}(\{s_1, s_2\})$ and $\mathsf{Len}(\{s_3, s_4\})$. Our estimator uses the frequency of each unordered response to estimate the appropriate linear combination of lengths of each constraint with rigorous probabilistic guarantees.

Step 3: Compute Convex Polytope & Output the Mean of the Polytope Diameter. At this point we have estimated the feasible region of the offset vector ξ, depicted in Figure 3.5. As a next step the attack utilizes a solver for the *Vertex Enumeration Problem* [13] which takes as an input the linear inequalities (i.e., the constraints) and outputs the *vertices* on the boundary of the feasible region $\mathcal{F}_{[v]}$. Having the coordinates of the vertices of $\mathcal{F}_{[v]}$, our attack can compute the *diameter* of the convex polytope. As it is shown in Theorem 6, the offset ξ^*, which is defined as the mean of a pair of polytope-vertices that constitute the diameter, gives a representative v^* that has distance at most $diam(\mathcal{F}_{[v]})/2$ from *all the n-tuples of the reconstruction class*, including the (unknown) original database.

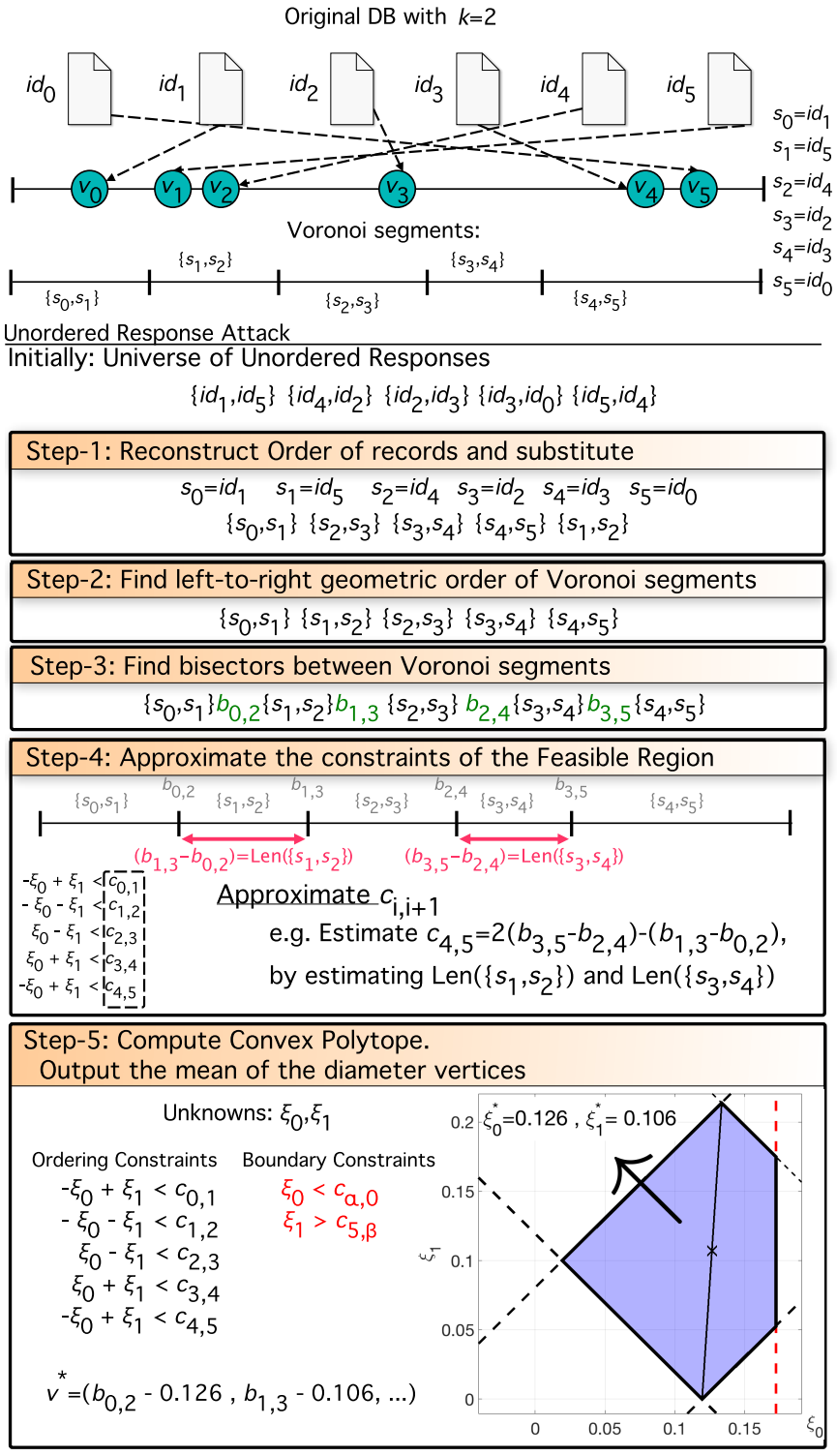


Figure 3.5: An overview of the attack based on *unordered responses* for $k = 2$.

3.3.4 Unordered Responses: Reconstruction for $k \geq 2$

Estimating the Constraints. We define the tuple L as:

$$L = \big(\mathsf{Len}(\{s_0, \ldots, s_{k-1}\}), \ldots, \mathsf{Len}(\{s_{n-k}, \ldots, s_{n-1}\})\big)$$

, where $\mathsf{Len}(\cdot)$ indicates the length of the Voronoi segment that is given as an input. Our goal is to estimate the expression of each constraint and to achieve this the next (simplified) lemma is of great importance. Specifically, the following lemma shows that each of the ordering constraints (same argument holds for the boundary) can be expressed as a simple linear combination of ξ and L where the coefficients are known. Since we can estimate the lengths of L using Equation (3.4) we have a way to estimate the constraints as well. The proof performs a case analysis of the inequality $v_i < v_{i+1}$ based on the value of i with respect to the formulas of Lemma 12.

Lemma 13. *The ordering constraint $v_i < v_{i+1}$ can be expressed as a function of A) the offsets $\xi = (\xi_0, \ldots, \xi_{k-1})$ and B) the lengths of a subset of Voronoi segments. Specifically by using the expressions of v_i from Lemma 12 we get the following cases:*

- *if $0 \leq i < k-1$, then $v_i < v_{i+1}$ can be written as:*

$$-\xi_i + \xi_{i+1} < c_{i,i+1}, \text{ where } c_{i,i+1} = \mathsf{Len}(\{s_{i+1}, \ldots, s_{i+k}\})$$

- *if $i = k-1$, then $v_i < v_{i+1}$ can be written as:*

$$-\xi_{k-1} - \xi_0 < c_{k-1,k}, \text{ where } c_{k-1,k} = -\sum_{1 \leq l \leq k-1} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\})$$

- *if $k \leq i < 2k-1$, then $v_i < v_{i+1}$ can be written as:*

$$\xi_{i \bmod k} - \xi_{i \bmod k+1} < c_{i,i+1}, \text{ where } c_{i,i+1} = \mathsf{Len}(\{s_{i \bmod k+1}, \ldots, s_{i \bmod k+k}\})$$

- *if $i = 2k-1$, then $v_i < v_{i+1}$ can be written as:*

$$\xi_{k-1} + \xi_0 < c_{2k-1,2k}, \text{ where } c_{2k-1,2k} = \mathsf{Len}(\{s_k, \ldots, s_{2k-1}\}) + \sum_{1 \leq l \leq k} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\})$$

- *if* $2k \leq i < n-1$ *and* $(i+1) \bmod k \neq 0$*, then* $v_i < v_{i+1}$ *can be written as:*

$$(-1)^{\lfloor i/k-1 \rfloor}(\xi_{i \bmod k} - \xi_{(i+1) \bmod k}) < c_{i,i+1}$$

$$\text{, where } c_{i,i+1} = (-1)^{\lfloor i/k-1 \rfloor}(\mathit{Len}(\{s_{(i+1) \bmod k}, \ldots, s_{(i+1) \bmod k+k-1}\})$$
$$+ \sum_{2 \leq j \leq \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2\mathit{Len}(\{s_{i \bmod k+(j-1)k+1}, \ldots, s_{(i+1) \bmod k+jk-1}\})$$

- *if* $2k \leq i < n-1$ *and* $(i+1) \bmod k = 0$*, then* $v_i < v_{i+1}$ *can be written as:*

$$(-1)^{\lfloor i/k \rfloor+1}(\xi_{i \bmod k} + \xi_{(i+1) \bmod k}) < c_{i,i+1}$$

$$\text{, where } c_{i,i+1} = (-1)^{\lfloor i/k \rfloor+1}\Big(\sum_{1 \leq l \leq k} \mathit{Len}(\{s_l, \ldots, s_{l+k-1}\})\Big)$$
$$+ (-1)^{\lfloor i/k \rfloor+1}\mathit{Len}(\{s_k, \ldots, s_{2k-1}\}) + \sum_{2 \leq j \leq \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2(\mathit{Len}(\{s_{jk}, \ldots, s_{jk+k-1}\}))$$

The first three cases the term $c_{i,i+1}$ *consists of the length of a single Voronoi segment. For the fourth case the term* $c_{i,i+1}$ *is a linear combination of* $2k-1$ *length terms. For the fifth case the term* $c_{i,i+1}$ *is a linear combination of at most* $\lfloor (n-1)/k \rfloor$ *length terms. Finally for the last case* $c_{i,i+1}$ *is a linear combination of at most* $\lfloor (n-1)/k \rfloor + k$ *length terms.*

Proof. We proceed by doing a case analysis on the value of i using the formulas of Lemma 12. A useful observation for the rest of the analysis is that the Voronoi segment between consecutive bisectors $b_{l,l+k}$ and $b_{l+1,l+1+k}$ is labeled as $\{s_{l+1}, \ldots, s_{l+k}\}$. So the difference $b_{l+1,l+1+k} - b_{l,l+k}$ is equal to $\mathsf{Len}(\{s_{l+1}, \ldots, s_{l+k}\})$.

Case (1) if $0 \leq i < k-1$**:** Then we have:

$$v_i < v_{i+1} \Rightarrow b_{i,i+k} - \xi_i < b_{i+1,i+1+k} - \xi_{i+1} \Rightarrow -\xi_i + \xi_{i+1} < b_{i+1,i+1+k} - b_{i,i+k}$$

The bisectors $b_{i,i+k}$ and $b_{i+1,i+1+k}$ are consecutive bisectors. Therefore, the difference of their location is equal to the *length of the Voronoi segment* that is in-between.

$$-\xi_i + \xi_{i+1} < \mathsf{Len}(\{s_{i+1}, \ldots, s_{i+k}\}) \tag{3.8}$$

Case (2) if $i = k - 1$: Then we have:

$$v_{k-1} < v_k \Rightarrow b_{k-1,2k-1} - \xi_{k-1} < b_{0,k} + \xi_0 \Rightarrow -\xi_{k-1} - \xi_0 < -(b_{k-1,2k-1} - b_{0,k})$$

In this case the difference is equivalent with the *the sum of the lengths of the Voronoi segments that reside in-between the two bisectors.* Therefore:

$$(b_{k-1,2k-1} - b_{0,k}) = \sum_{1 \leq l \leq k-1} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\})$$

Case (3) if $k \leq i < 2k - 1$: Then we have:

$$\begin{aligned} v_i < v_{i+1} &\Rightarrow b_{i \bmod k, i \bmod k+k} + \xi_{i \bmod k} < b_{(i+1) \bmod k,(i+1) \bmod k+k} + \xi_{(i+1) \bmod k} \\ &\Rightarrow \xi_{i \bmod k} - \xi_{i \bmod k+1} < b_{i \bmod k+1, i \bmod k+1+k} - b_{i \bmod k, i \bmod k+k} \end{aligned}$$

The bisectors $b_{i \bmod k+1, i \bmod k+1+k}$ and $b_{i \bmod k, i \bmod k+k}$ are consecutive bisectors. Therefore, the difference of their location is equal to the length of the Voronoi segment that is in-between.

$$\xi_{i \bmod k} - \xi_{i \bmod k+1} < \mathsf{Len}(\{s_{i \bmod k+1}, \ldots, s_{i \bmod k+k}\}) \tag{3.9}$$

Case (4) if $i = 2k - 1$:

Then we have:

$$\begin{aligned} v_{2k-1} < v_{2k} &\Rightarrow b_{(2k-1) \bmod k,(2k-1) \bmod k+k} + \xi_{(2k-1) \bmod k} < -(b_{0,k} + \xi_0) + 2b_{k,2k} \\ &\Rightarrow \xi_{k-1} + \xi_0 < 2b_{k,2k} - b_{0,k} - b_{k-1,2k-1} = (b_{k,2k} - b_{0,k}) + (b_{k,2k} - b_{k-1,2k-1}) \end{aligned}$$

Notice that in the difference of bisectors $(b_{k,2k} - b_{0,k})$ does not concern consecutive bisectors. In this case the difference is equivalent with the *the sum of the lengths of the Voronoi segments that reside in-between the two bisectors.* Therefore:

$$(b_{k,2k} - b_{0,k}) = \sum_{1 \leq l \leq k} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\})$$

Using a similar argument we have:

$$(b_{k,2k} - b_{k-1,2k-1}) = \mathsf{Len}(\{s_k, \ldots, s_{2k-1}\})$$

So overall we have:

$$\xi_{k-1} + \xi_0 < \mathsf{Len}(\{s_k, \ldots, s_{2k-1}\}) + \sum_{1 \le l \le k} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\}) \tag{3.10}$$

Remaining Cases: For the last cases we have the following expression.

$$\begin{aligned}
&v_i < v_{i+1} \Rightarrow (-1)^{\lfloor i/k-1 \rfloor} \xi_{i \bmod k} - (-1)^{\lfloor (i+1)/k-1 \rfloor} \xi_{(i+1) \bmod k} \\
&< (-1)^{\lfloor (i+1)/k-1 \rfloor} b_{(i+1) \bmod k, ((i+1) \bmod k)+k} \\
&- (-1)^{\lfloor i/k-1 \rfloor} b_{i \bmod k, (i \bmod k)+k} + \\
&+ \sum_{2 \le j \le \lfloor (i+1)/k \rfloor} (-1)^{j+\lfloor (i+1)/k \rfloor} 2 b_{((i+1) \bmod k)+(j-1)k, ((i+1) \bmod k)+jk} \\
&- \sum_{2 \le j \le \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2 b_{(i \bmod k)+(j-1)k, (i \bmod k)+jk}
\end{aligned}$$

If term $\lfloor (i+1)/k \rfloor$ is different from $\lfloor i/k \rfloor$ then the first sum has an extra term. The only way these two differ is for $i+1$ to be a multiple of k, i.e., $(i+1) \bmod k = 0$. We analyze the remaining two cases in the remaining of the proof.

Case (5) if $i \ge 2k$ and $(i+1) \bmod k \neq 0$: then we have that $\lfloor i/k \rfloor = \lfloor (i+1)/k \rfloor$. Consequently, $\lfloor i/k - 1 \rfloor = \lfloor (i+1)/k - 1 \rfloor$. Thus:

$$\begin{aligned}
&(-1)^{\lfloor i/k-1 \rfloor} (\xi_{i \bmod k} - \xi_{(i+1) \bmod k}) < (-1)^{\lfloor i/k-1 \rfloor} (b_{(i+1) \bmod k, ((i+1) \bmod k)+k} - b_{i \bmod k, (i \bmod k)+k}) \\
&+ \sum_{2 \le j \le \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2 \big(b_{((i+1) \bmod k)+(j-1)k, ((i+1) \bmod k)+jk} - b_{(i \bmod k)+(j-1)k, (i \bmod k)+jk} \big)
\end{aligned}$$

Since $((i+1) \bmod k) + (j-1)k$ is equal to $\big((i \bmod k) + (j-1)k\big) + 1$, the two bisectors of the sum $b_{((i+1) \bmod k)+(j-1)k, ((i+1) \bmod k)+jk}$ and $b_{(i \bmod k)+(j-1)k, (i \bmod k)+jk}$ are consecutive bisectors. Similarly, since $(i+1) \bmod k = i \bmod k + 1$, the two bisectors $b_{(i+1) \bmod k, ((i+1) \bmod k)+k}$ and $b_{i \bmod k, (i \bmod k)+k}$ are consecutive bisectors. Therefore, the difference of their location is equal to the

length of the Voronoi segment that is in-between. Thus, we have:

$$(-1)^{\lfloor i/k-1\rfloor}(\xi_{i \bmod k} - \xi_{(i+1) \bmod k}) < (-1)^{\lfloor i/k-1\rfloor}(\mathsf{Len}(\{s_{(i+1) \bmod k}, \ldots, s_{((i+1) \bmod k)+k-1}\}) + \sum_{2\le j\le\lfloor i/k\rfloor} (-1)^{j+\lfloor i/k\rfloor} 2\mathsf{Len}(\{s_{(i \bmod k)+(j-1)k+1}, \ldots, s_{((i+1) \bmod k)+jk-1}\}) \tag{3.11}$$

Case (6) if $i \ge 2k$ and $(i+1) \bmod k = 0$: then we have that $\lfloor i/k\rfloor + 1 = \lfloor (i+1)/k\rfloor$. We also have that, $\lfloor i/k-1\rfloor + 1 = \lfloor (i+1)/k-1\rfloor$. Since $(i+1) \bmod k = 0$ we also have that $i \bmod k = k-1$.

$$\begin{aligned}
&(-1)^{\lfloor i/k-1\rfloor}(\xi_{i \bmod k} + \xi_{(i+1) \bmod k}) \\
&< (-1)^{\lfloor (i+1)/k-1\rfloor} b_{(i+1) \bmod k,((i+1) \bmod k)+k} - (-1)^{\lfloor i/k-1\rfloor} b_{i \bmod k,(i \bmod k)+k} + \\
&+ \sum_{2\le j\le\lfloor (i+1)/k\rfloor} (-1)^{j+\lfloor (i+1)/k\rfloor} 2b_{(j-1)k,jk} - \sum_{2\le j\le\lfloor i/k\rfloor} (-1)^{j+\lfloor i/k\rfloor} 2b_{k-1+(j-1)k,k-1+jk} \\
&= (-1)^{\lfloor i/k-1\rfloor+1} b_{0,k} - (-1)^{\lfloor i/k-1\rfloor} b_{k-1,k-1+k} + \\
&+ \sum_{2\le j\le\lfloor (i+1)/k\rfloor} (-1)^{j+\lfloor (i+1)/k\rfloor} 2b_{(j-1)k,jk} - \sum_{2\le j\le\lfloor i/k\rfloor} (-1)^{j+\lfloor i/k\rfloor} 2b_{jk-1,jk-1+k} \\
&= -(-1)^{\lfloor i/k-1\rfloor} b_{0,k} - (-1)^{\lfloor i/k-1\rfloor} b_{k-1,k-1+k} + (-1)^{2+\lfloor i/k\rfloor+1} 2b_{k,2k} \\
&+ \sum_{2\le j\le\lfloor i/k\rfloor} (-1)^{j+1+\lfloor i/k\rfloor+1} 2b_{jk,jk+k} - (-1)^{j+\lfloor i/k\rfloor} 2b_{jk-1,jk-1+k} \\
&= (-1)^{\lfloor i/k\rfloor} b_{0,k} + (-1)^{\lfloor i/k\rfloor} b_{k-1,k-1+k} - (-1)^{\lfloor i/k\rfloor} 2b_{k,2k} + \sum_{2\le j\le\lfloor i/k\rfloor} (-1)^{j+\lfloor i/k\rfloor} 2(b_{jk,jk+k} - b_{jk-1,jk-1+k}) \\
&= -(-1)^{\lfloor i/k\rfloor}(b_{k,2k} - b_{0,k}) - (-1)^{\lfloor i/k\rfloor}(b_{k,2k} - b_{k-1,k-1+k}) + \sum_{2\le j\le\lfloor i/k\rfloor} (-1)^{j+\lfloor i/k\rfloor} 2(b_{jk,jk+k} - b_{jk-1,jk-1+k})
\end{aligned}$$

The bisectors $b_{jk,jk+k}$ and $b_{jk-1,jk-1+k}$ are consecutive bisectors. Therefore, the difference of their location is equal to the length of the Voronoi segment that is in-between:

$$(b_{jk,jk+k} - b_{jk-1,jk-1+k}) = \mathsf{Len}(\{s_{jk}, \ldots, s_{jk+k-1}\})$$

The remaining two bisector differences can also be expressed as lengths of Voronoi segments as

follows:

$$(b_{k,2k} - b_{0,k}) = \sum_{1 \le l \le k} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\})$$

$$(b_{k,2k} - b_{k-1,k-1+k}) = \mathsf{Len}(\{s_k, \ldots, s_{2k-1}\})$$

Therefore the overall expression is:

$$\begin{aligned} &- (-1)^{\lfloor i/k \rfloor}(\xi_{i \bmod k} + \xi_{(i+1) \bmod k}) < -(-1)^{\lfloor i/k \rfloor}\Big(\sum_{1 \le l \le k} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\})\Big) \\ &- (-1)^{\lfloor i/k \rfloor}\mathsf{Len}(\{s_k, \ldots, s_{2k-1}\}) + \sum_{2 \le j \le \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2(\mathsf{Len}(\{s_{jk}, \ldots, s_{jk+k-1}\})) \end{aligned} \tag{3.12}$$

□

A similar lemma can be formed for the boundary constraints. The values of the coefficients of f_i^{Left} and f_i^{Right} can be easily computed since they only depend on i, n, k. The ConstraintEstimation algorithm focuses on estimating $c_{i,i+1} = f_i^{Right} \cdot L^T$ and performs the following series of actions for each boundary and ordering constraint: given i, n, k compute the coefficients f_i^{Left}, f_i^{Right}, scan the multiset of unordered responses U that come from uniformly generated queries and record the observed frequency of each relevant entry of L, use the coefficients f_i^{Right} and the values in L^T to finalize the estimation of the terms $c_{i,i+1}$.

Attack Algorithm. The attack algorithm utilizes algorithm ConstraintEstimation to approximate the inequalities of the feasible region, i.e., get the estimates $\tilde{c}$. The final set of inequalities is captured by the expression $A \cdot \xi \le \tilde{c}$ where by a linear scan the attacker can remove the redundant constraints (Line 3 of the algorithm). Notice that the ordering constraints concern a pair of consecutive values and by substituting from Lemma 12 we finally get a constraint on a pair of offsets that appear consecutively in the cyclical ordering $\hookrightarrow \xi_0 \to \xi_1 \to \ldots \to \xi_{k-1}$. Due to the periodicity on the cyclical ordering we have $2k$ non-redundant ordering constraints among the total $n-1$. For $n > 2k$ the polytope is bounded and thus the solver of the vertex enumeration problem [13] returns the vertices of the k-dimensional polytope formed by $A \cdot \xi \le \tilde{c}$ in $O(k^2\, z)$ time, where z is the number of vertices of the polytope. In general, z could be as large as 2^k. Thus, our approach is suitable for small values of k, which is typical in practical scenarios where k is often a small constant.

We note here that in case the estimation of the constraints is not "accurate enough", which

High-Level Description of ConstraintEstimation

- **Input:** Multiset U of responses from queries generated uniformly at random. Order S of the ids wrt their values. Boundaries α, β.
- **Step 1:** Let L be the $(n-k+1)$-tuple of the labels of the (unknown) lengths of the Voronoi segments. Calculate the formula of each *ordering constraint* wrt ξ and compute the coefficients f_i^{Right} of $c_{i,i+1}$. Given the analytical formula of each term $c_{i,i+1}$ define $T_{i,i+1}$ to be the set of triplets (lbl, cfc, cnt), where lbl is the label of the participating length from L, cfc is the coefficient of this lbl in the formula, and cnt is a counter initialized to zero.
- **Step 2:** Similarly, calculate the analytical formulas of the terms $c_{\alpha,0}$ and $c_{n-1,\beta}$ of the *boundary constraints*. Define the sets of triplets $T_{\alpha,0}$ and $T_{n-1,\beta}$. Let T be the collection of sets $\{T_{\alpha,0}, T_{0,1}, \ldots, T_{n-2,n-1}, T_{n-1,\beta}\}$.
- **Step 3:** For each response $r \in U$ find the sets from collection T where the term $\mathsf{Len}(r)$ is part of a lbl and increase the corresponding cnt entry by one.
- **Step 4:** Set all estimations $\widetilde{c}_{i,i+1}$ to zero. For each set of collection T, go through all the triplets. For each triplet of $T_{i,i+1}$, multiply cfc with the counter cnt and add the result to $\widetilde{c}_{i,i+1}$.
- **Step 5:** Multiply each of the $\widetilde{c}_{i,i+1}$ by $(\beta-\alpha)$, divide the result by $|U|$, and store the final result at $\widetilde{c}_{i,i+1}$.
- **Step 6:** Output $\widetilde{c}_{\alpha,0}, \widetilde{c}_{0,1}, \ldots, \widetilde{c}_{n-2,n-1}, \widetilde{c}_{n-1,\beta}$.

depends on the distribution of the values, the feasible region might be empty. In this case the solver will return an empty set and the attack will fail since no offset can meet the (not adequately) approximated constraints. Given the vertices we can compute the diameter of the polytope of $\mathcal{F}_{[v]}$ in time quadratic in the number of vertices. So as a last step our attack returns the mean of the diameter vertices which guarantees that all the n-tuples of the class are at most $diam(\mathcal{F}_{[v]})/2$ distance afar.

Theorem 6. *Let DB be an encrypted database with n records whose values are in the range* $[\alpha, \beta]$. *Suppose the attacker observes the responses to m k-NN queries uniformly generated from* $[\alpha, \beta]$ *(Assumption 1) and which contain the set of all possible unordered responses, R. For any* $0 < \epsilon < \beta - \alpha$ *and* $0 < \delta < 1$, *Algorithm* AttackUnordered *runs in time* $O(m + k^2 z + z^2)$, *where z is the number of vertices of the feasible region,* $\mathcal{F}_{[v]}$, *and returns either* $\perp$ *(failure) or a sequence of reconstructed values (success) such that each reconstructed value differs from its original value by at most* $\frac{diam\left(\widetilde{\mathcal{F}_{[v]}}\right)}{2} + \epsilon$

Algorithm 5: AttackUnordered

Input: Response multiset $U = \{r_1, r_2, \ldots\}$, Boundaries α, β
Output: Reconstructed values $(v_0^*, \ldots, v_{n-1}^*)$ or $\perp$

$S \leftarrow \mathsf{ReconstructOrder}(U)$, $\tilde{c} \leftarrow \mathsf{ConstraintEstimation}(U, S, \alpha, \beta)$;
Compute the $(n + k + 1) \times k$ matrix A of coefficients such that each line of $A \cdot \xi < \tilde{c}$ represents a constraint, ξ is the column vector with k offsets, $\tilde{c}$ is the column vector with $(n + k + 1)$ entries of the constants;
Remove the redundant constraints from A;
Deploy an algorithm that solves the 'Vertex Enumeration Problem' with input $A \cdot \xi \leq \tilde{c}$ and output a matrix Ξ of k columns where each row represents a vertex of the convex polytope of the feasible region;
If Ξ is non-empty **then** compute the Euclidean distance between every pair of rows (i.e., vertices) of Ξ and record the pair (ξ', ξ'') with the maximum distance, **else** return $\perp$;
Compute ξ^* as the mean of ξ' and ξ'';
Use the offset $\xi^* = (\xi_0^*, \ldots, \xi_{k-1}^*)$ in the expressions of Lemma 12 to compute the corresponding value $v^* = (v_0^*, \ldots, v_{n-1}^*)$;
return $(v_0^*, \ldots, v_{n-1}^*)$

with probability at least $1 - \delta$*, provided* m *is at least*

$$\max\left\{\frac{25(\beta-\alpha)^2(\ln 3 - \ln \delta)}{\epsilon^2}, \frac{20(\beta-\alpha)^2(n-k+1)}{\epsilon^2}\right\}$$

Proof. The proof of this Theorem is similar to the proof of Theorem 5, in terms of probabilistic analysis. Let $(Y_1, \ldots, Y_{|R|})$ be a multinomial distribution, that is $|R|$ is a fixed number and we have $|R|$ mutually exclusive outcomes with corresponding probabilities $p_1, \ldots, p_{|R|}$, and m independent trials. From Lemma 3 in [56] we have that for all $\epsilon_1 \in (0, 1)$ and all $|R|$ satisfying $|R|/m \leq \epsilon_1^2/20$ we have:

$$\Pr\big(\sum_{i=1}^{|R|} |Y_i - E[Y_i]| > m\epsilon_1\big) \leq 3e^{-m\epsilon_1^2/25}$$

From which we can similarly derive:

$$\big(\sum_{i=1}^{|R|} \frac{E[Y_i]}{m}\big) - \epsilon_1 \leq \sum_{i=1}^{|R|} \frac{Y_i}{m} \leq \big(\sum_{i=1}^{|R|} \frac{E[Y_i]}{m}\big) + \epsilon_1$$

Transformation to Constraint Estimation. The multinomial formulation can be adjusted to assist to the task of constraint estimation. Let each trial correspond to a uniformly chosen query point from $[\alpha, \beta]$, then the outcome of the trial corresponds to an unordered response from the Voronoi diagram $V_k(DB)$. The number of trials for the case of a multinomial corresponds to the number of queries and is denoted by m. Therefore the number of possible outcomes is $|R|$ which is the number

of Voronoi segments of order k. Notice that the probability that a response is r_i is equal to the probability p_i that a uniformly chosen query point lands to the Voronoi segment $V_k(r_i)$, which in turn is equal to the ratio of the length of the corresponding Voronoi segment to the length of the entire bounded metric space. Specifically $p_i = \mathsf{Len}(V_k(r_i))/(\beta - \alpha)$.

We proceed by doing a case analysis on the possible values of i and how it affects the formulation of the $c_{i,i+1}$ (see Lemma 13) of the ordering constraint and consequently the probability expression of the previous paragraph. Each outcome of the multinomial $(Y_1, \ldots, Y_{|R|})$ corresponds to a Voronoi segment in the left-to-right order. Roughly, the calculations of Theorem 5 are performed for summation with $|R|$ terms, using Lemma 13 we show that in this Theorem and for cases (1)-(5) we sum $|C_i|$ terms where $|C_i| \leq |R|$. Therefore the analysis of Theorem 5 holds for these cases as well. Case (6) is different and we explain how the analysis changes.

Case (1) where $0 \leq i < k-1$. In this case the term $c_{i,i+1}$ has the following formulation:

$$c_{i,i+1} = \mathsf{Len}(\{s_{i+1}, \ldots, s_{i+k}\})$$

Let C_i be the set that contains the index of the Voronoi segment with ids $\{s_{i+1}, \ldots, s_{i+k}\}$ wrt the ordering $(Y_1, \ldots, Y_{|R|})$. Since $|C_i| = 1$ we have:

$$\sum_{l \in C_i} |Y_l - E[Y_l]| \leq \sum_{i=1}^{|R|} |Y_i - E[Y_i]| \leq m\epsilon_1$$

Following similar calculations as in the proof of Theorem 5 we get:

$$\Pr\left(|c_{i,i+1} - \widetilde{c_{i,i+1}}| \leq (\beta - \alpha)\epsilon_1\right) > 1 - 3e^{-m\epsilon_1^2/25}.$$

Case (2) where $i = k-1$. In this case the term $c_{i,i+1}$ has the following formulation:

$$c_{k-1,k} = -\sum_{1 \leq l \leq k-1} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\})$$

Let C_i be the set that contains the index of each Voronoi segment of the above expression, i.e., the index of responses $\{s_l, \ldots, s_{i+k-1}\}$ for $1 \leq l \leq k-1$ wrt the ordering $(Y_1, \ldots, Y_{|R|})$. Since $|C_i| = k-1$

and all the participating lengths are unique we have:

$$\sum_{l \in C_i} |Y_l - E[Y_l]| \leq \sum_{i=1}^{|R|} |Y_i - E[Y_i]| \leq m\epsilon_1$$

Following similar calculations as in the proof of Theorem 5 we get:

$$\Pr\left(|c_{i,i+1} - \widetilde{c_{i,i+1}}| \leq (\beta - \alpha)\epsilon_1\right) > 1 - 3e^{-m\epsilon_1^2/25}.$$

Case (3) where $k \leq i < 2k-1$. In this case the term $c_{i,i+1}$ has the following formulation:

$$c_{i,i+1} = \mathsf{Len}(\{s_{i \bmod k+1}, \ldots, s_{i \bmod k+k}\})$$

Let C_i be the set that contains the index of the Voronoi segment of the above expression, i.e., index of $\{s_{i \bmod k+1}, \ldots, s_{i \bmod k+k}\}$ wrt $(Y_1, \ldots, Y_{|R|})$. Since $|C_i| = 1$ we have:

$$\sum_{l \in C_i} |Y_l - E[Y_l]| \leq \sum_{i=1}^{|R|} |Y_i - E[Y_i]| \leq m\epsilon_1$$

Following similar calculations as in the proof of Theorem 5 we get:

$$\Pr\left(|c_{i,i+1} - \widetilde{c_{i,i+1}}| \leq (\beta - \alpha)\epsilon_1\right) > 1 - 3e^{-m\epsilon_1^2/25}.$$

Case (4) where $i = 2k-1$. In this case the term $c_{i,i+1}$ has the following formulation:

$$c_{2k-1,2k} = \mathsf{Len}(\{s_k, \ldots, s_{2k-1}\}) + \sum_{1 \leq l \leq k} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\})$$

Let C_i be the set that contains the index of each Voronoi segment of the above expression, i.e., index of $\{s_l, \ldots, s_{i+k-1}\}$ for $1 \leq l \leq k$ wrt $(Y_1, \ldots, Y_{|R|})$. Notice that in the above expression the length of segment $\{s_k, \ldots, s_{2k-1}\}$ is counted twice. Additionally the size of the set C_i is $k+1$, i.e., less than

$|R|$. So if we define an ϵ_2 such that $\epsilon_2 = 2\epsilon_1$ then we have:

$$\sum_{l \in C_i} |Y_l - E[Y_l]| \leq \sum_{i=1}^{|R|} |Y_i - E[Y_i]| \leq m2\epsilon_1 = m\epsilon_2/2$$

Following similar calculations as in the proof of Theorem 5 we get:

$$\Pr\left(|c_{i,i+1} - \widetilde{c_{i,i+1}}| \leq (\beta - \alpha)\epsilon_2/2\right) > 1 - 3e^{-m\epsilon_2^2/100}.$$

Case (5) where $i \geq 2k$ and $(i+1) \bmod k \neq 0$. In this case the term $c_{i,i+1}$ has the following formulation:

$$c_{i,i+1} = (-1)^{\lfloor i/k-1 \rfloor}(\mathsf{Len}(\{s_{(i+1) \bmod k+1}, \ldots, s_{(i+1) \bmod k+k-1}\}) + \sum_{2 \leq j \leq \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2\mathsf{Len}(\{s_{i \bmod k+(j-1)k+1}, \ldots, s_{(i+1) \bmod k+jk-1}\})$$

Let C_i be the set that contains the index of each Voronoi segment of the above expression, i.e., $\{s_{(i+1) \bmod k+1}, \ldots, s_{(i+1) \bmod k+k-1}\}$ and $\{s_{i \bmod k+(j-1)k+1}, \ldots, s_{(i+1) \bmod k+jk-1}\}$ for $2 \leq j \leq \lfloor i/k \rfloor$, in the ordering of $(Y_1, \ldots, Y_{|R|})$. Notice that in the above expression all the Voronoi segments are unique since we have $(i+1) \bmod k + 1 \neq i \bmod k + (j-1)k + 1$ for integer values of i and j. Therefore, since the size of the set C_i is upper-bounded by $|R|$ and all segments are unique we have:

$$\sum_{l \in C_i} |Y_l - E[Y_l]| \leq \sum_{i=1}^{|R|} |Y_i - E[Y_i]| \leq m\epsilon_1$$

Following similar calculations as in the proof of Theorem 5 we get:

$$\Pr\left(|c_{i,i+1} - \widetilde{c_{i,i+1}}| \leq (\beta - \alpha)\epsilon_1\right) > 1 - 3e^{-m\epsilon_1^2/25}.$$

Case (6) where $i \geq 2k$ and $(i+1) \bmod k = 0$. In this case the term $c_{i,i+1}$ has the following

formulation:

$$c_{i,i+1} = (-1)^{\lfloor i/k \rfloor + 1}\Big(\sum_{1 \le l \le k} \mathsf{Len}(\{s_l, \ldots, s_{l+k-1}\})\Big)$$
$$+ (-1)^{\lfloor i/k \rfloor + 1}\mathsf{Len}(\{s_k, \ldots, s_{2k-1}\})$$
$$+ \sum_{2 \le j \le \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2(\mathsf{Len}(\{s_{jk}, \ldots, s_{jk+k-1}\}))$$

Let C_i be the set that contains the index of each Voronoi segment of the above expression, i.e., index of $\{s_l, \ldots, s_{l+k-1}\}$ for $1 \le l \le k$ and $\{s_k, \ldots, s_{2k-1}\}$ and $\{s_{jk}, \ldots, s_{jk+k-1}\}$ for $2 \le j \le \lfloor i/k \rfloor$ wrt the ordering $(Y_1, \ldots, Y_{|R|})$. Notice that in the above expression of $c_{i,i+1}$ the length of segment $\{s_k, \ldots, s_{2k-1}\}$ is counted twice. Additionally the size of the set C_i is upper-bounded by $|R|$. So if we define an ϵ_2 such that $\epsilon_2 = 2\epsilon_1$ then we have:

$$\sum_{l \in C_i} |Y_l - E[Y_l]| \le \sum_{i=1}^{|R|} |Y_i - E[Y_i]| \le m2\epsilon_1 = m\epsilon_2/2$$

Following similar calculations as in the proof of Theorem 5 we get:

$$\Pr\left(|c_{i,i+1} - \widetilde{c_{i,i+1}}| \le (\beta - \alpha)\epsilon_2/2) > 1 - 3e^{-m\epsilon_2^2/100}.\right.$$

Overall. Let us define $\epsilon = (\beta - \alpha)\epsilon_2/2$. From the above case analysis we conclude that the following expression holds for *all* the cases:

$$\Pr\left(|c_{i,i+1} - \widetilde{c_{i,i+1}}| \le \epsilon) > 1 - 3e^{-\frac{m\epsilon^2}{(\beta-\alpha)^2 25}}.\right.$$

We define δ as $\delta \ge 3e^{-\frac{m\epsilon^2}{(\beta-\alpha)^2 25}}$, and we get:

$$\delta \ge 3e^{-\frac{m\epsilon^2}{(\beta-\alpha)^2 25}} \Rightarrow \ln\delta \ge \ln 3 - \frac{m\epsilon^2}{(\beta-\alpha)^2 25} \Rightarrow m \ge \frac{25(\beta-\alpha)^2(\ln 3 - \ln\delta)}{\epsilon^2}$$

Also with some algebraic manipulation of the inequality about m from Lemma 3 in [56] we get:

$$\frac{|R|}{m} \le \frac{\epsilon_1^2}{20} \Rightarrow \frac{|R|}{m} \le \frac{\epsilon_2^2}{80} \Rightarrow \frac{|R|}{m} \le \frac{4\epsilon^2}{80(\beta-\alpha)^2} \Rightarrow m \ge \frac{20(\beta-\alpha)^2|R|}{\epsilon^2}$$

Since $|R| = n - k + 1$ we get

$$m \geq \frac{20(\beta - \alpha)^2(n - k + 1)}{\epsilon^2}.$$

So with the above analysis we proved the following statement

Let m be the number of uniformly drawn query points from $[\alpha, \beta]$ for a database DB with n unique values. Let's assume that:

$$m \geq \max\left\{\frac{25(\beta - \alpha)^2(\ln 3 - \ln \delta)}{\epsilon^2}, \frac{20(\beta - \alpha)^2(n - k + 1)}{\epsilon^2}\right\}$$

Then for any $\delta \in (0, 1)$ and $\epsilon \in (0, |\beta - \alpha|)$ the Algorithm ConstraintEstimation returns $\widetilde{c_{i,i+1}}$ in $O(kn)$ time such that:

$$\Pr\left(|c_{i,i+1} - \widetilde{c_{i,i+1}}| \leq \epsilon\right) \geq 1 - \delta$$

, for any $0 \leq i \leq n - 2$.

Therefore the hyperplanes derived by the attacker (see the inequalities of Lemma 13) are a function of the *estimations* $\widetilde{c_{i,i+1}}$ rather then actual $c_{i,i+1}$. As a result we have an approximation $\widetilde{\mathcal{F}_{[v]}}$ of the real $\mathcal{F}_{[v]}$. Which implies that the diameter that is computed by the output of the solver of the Vertex Enumeration Problem is an estimate of the actual diameter. Since the location of a vertex of the approximated polytope $\widetilde{\mathcal{F}_{[v]}}$ in the k-dimensional space is within an ϵ-ball of the corresponding vertex of the actual polytope $\mathcal{F}_{[v]}$, the estimated diameter can be at most 2ϵ afar with probability $1 - \delta$ Specifically:

$$\Pr\left(|diam\left(\mathcal{F}_{[v]}\right) - diam\left(\widetilde{\mathcal{F}_{[v]}}\right)| \leq 2\epsilon\right) \geq 1 - \delta$$

Therefore the above analysis about the constraints can be interpreted as:

From the above statement we can derive the following probability expression:

$$\Pr\left(diam\left(\mathcal{F}_{[v]}\right) + 2\epsilon \geq diam\left(\widetilde{\mathcal{F}_{[v]}}\right) \geq diam\left(\mathcal{F}_{[v]}\right) - 2\epsilon\right) \geq 1 - \delta$$
$$\Rightarrow \Pr\left(\frac{diam\left(\mathcal{F}_{[v]}\right)}{2} \leq \frac{diam\left(\widetilde{\mathcal{F}_{[v]}}\right)}{2} + \epsilon\right) \geq 1 - \delta$$

Let m be the number of uniformly drawn query points from $[\alpha, \beta]$ for a database DB with n unique values. Let's assume that:

$$m \geq \max\left\{\frac{25(\beta-\alpha)^2(\ln 3 - \ln \delta)}{\epsilon^2}, \frac{20(\beta-\alpha)^2(n-k+1)}{\epsilon^2}\right\}$$

Then for any $\delta \in (0,1)$ and $\epsilon \in (0, |\beta - \alpha|)$ the Algorithm ConstraintEstimation returns $\widetilde{c_{i,i+1}}$ in $O(kn)$ time such that:

$$\Pr\left(|diam\left(\mathcal{F}_{[v]}\right) - diam\left(\widetilde{\mathcal{F}_{[v]}}\right)| \leq 2\epsilon\right) \geq 1-\delta$$

, for any $0 \leq i \leq n-2$.

Let's assume for a second that the attack could compute the real value of the diameter, i.e., $diam\left(\mathcal{F}_{[v]}\right)$. Then if we denote as v^{DB} the unknown encrypted n-tuple of values of DB we would have the following guarantee for the output v^* of the reconstruction:

$$d_{L_\infty}(v^{DB}, v^*) = \max_{0 \leq i \leq n-1} |v_i^{DB} - v_i^*| \leq \frac{diam\left(\mathcal{F}_{[v]}\right)}{2}.$$

But since the attacker can only computed the diameter of the approximated polytope we derive:

$$\Pr\left(\max_{0 \leq i \leq n-1} |v_i^{DB} - v_i^*| \leq \frac{diam\left(\widetilde{\mathcal{F}_{[v]}}\right)}{2} + \epsilon\right) \geq 1-\delta$$

□

On the Size of the Diameter Since the approximation is a function of a quantity that depends on the distribution of the data, we further study the possible values that $diam(\mathcal{F}_{[v]})$ can take. In the next theorem we show that the $3k$ consecutive values that are within the smallest possible γ range give an upper-bound on the diameter of $\mathcal{F}_{[v]}$. Thus, a small concentrated number of consecutive values affects heavily the diameter of $\mathcal{F}_{[v]}$. We note here that the smaller the γ the higher the number of samples required to achieve meaningful approximation guarantees since the sample size is a function of the length of the smallest Voronoi segment, so there is an inherent trade-off.

Theorem 7. *Let $V_k(v)$ be the Voronoi diagram of reconstruction class $[v]$, and let v' be an n-tuple such that $v' \in [v]$. If there are $3k$ values of v' within an γ range in $[\alpha, \beta]$ then we have $diam(\mathcal{F}_{[v]}) \leq 2\gamma$.*

Proof. The proof is an induction on the number of values that are placed to the left of the group of $3k$ values that are within a γ-distance. Intuitively, we show that $3k$ values are enough to define constraints that form a polytope, and since the distance between this group of $3k$ values is upper bounded by γ then the diameter of the derived polytope is also upper bounded by a function of γ. A useful observation is that if all $3k$ values of the group are within γ-distance, then the *bisectors between any pair of these values* are also within γ-distance.

Base Case. In this case there are zero values to the left of the group. Therefore the group consists of values $G = \{v_0, \ldots, v_{3k-1}\}$ and the bisectors that correspond to Voronoi endpoints, and are also within this γ-range, are $b_{0,k}, b_{1,k+1}, \ldots, b_{2k-1,3k-1}$. We now analyze the first $2k-1$ ordering constraints.

From the first $k-1$ ordering constraint we have:

$$
\begin{aligned}
&v_0 < v_1 \Rightarrow -\xi_0 + \xi_1 < b_{1,k+1} - b_{0,k} \Rightarrow -\xi_0 + \xi_1 < \gamma \\
&v_1 < v_2 \Rightarrow -\xi_1 + \xi_2 < b_{2,k+2} - b_{1,k+1} \Rightarrow -\xi_1 + \xi_2 < \gamma \\
&\vdots \\
&v_{k-2} < v_{k-1} \Rightarrow -\xi_{k-2} + \xi_{k-1} < b_{k-1,2k-1} - b_{1,k+1} \Rightarrow -\xi_{k-2} + \xi_{k-1} < \gamma
\end{aligned}
$$

For the next ordering constraint we have:

$$v_{k-1} < v_k \Rightarrow -\xi_{k-1} + \xi_0 < b_{k,2k+1} - b_{k-1,2k-1} \Rightarrow -\xi_{k-1} + \xi_0 < \gamma$$

For the next $k-1$ ordering constraints we have:

$$
\begin{aligned}
&v_k < v_{k+1} \Rightarrow \xi_0 - \xi_1 < b_{k+1,2k+1} - b_{k,2k} \Rightarrow \xi_0 - \xi_1 < \gamma \\
&v_{k+1} < v_{k+2} \Rightarrow \xi_1 - \xi_2 < b_{k+2,2k+2} - b_{k+1,2k+1} \Rightarrow \xi_1 - \xi_2 < \gamma \\
&\vdots \\
&v_{2k-2} < v_{2k-1} \Rightarrow \xi_{k-2} - \xi_{k-1} < b_{2k-1,3k-1} - b_{2k-2,3k-2} \Rightarrow \xi_{k-2} - \xi_{k-1} < \gamma
\end{aligned}
$$

Each inequality of $-\xi_0 + \xi_1 < \gamma$ and $\xi_0 - \xi_1 < \gamma$ forms a half-space. Additionally, they are based on parallel lines on the sub-space of ξ_0 and ξ_1. The same statement holds for $-\xi_1 + \xi_2 < \gamma$ and $\xi_1 - \xi_2 < \gamma$ and the parallel lines are on subspace of ξ_1 and ξ_2. Following the same process we define pairs of half-spaces and the space that satisfies *all* the inequalities is a polytope in the k-dimensional

space. Additionally since each of the parallel lines in the corresponding subspace has y-intercept that is upper bounded by γ, the diameter of the polytope has diameter that is upper-bounded by 2γ.

Inductive Step. Let's assume that our statement holds for n values to the left of the group, we will prove that the statement holds for $n+1$ values as well. We elaborate on observations that hold for n values and later we show that these observations hold for the case of $n+1$ as well. First, let's assume without loss of generality that the order of the first value of the group is j. Therefore, the group of values is $G = \{v_j, v_{j+1}, \ldots, v_{j+3k-1}\}$. Additionally, the group of the corresponding bisectors is $b_{j,j+k}, b_{j+1,j+k+1}, \ldots, b_{j+2k-1,j+3k-1}$ and they are both within an γ-distance as we observed before. Therefore, we have the following set of inequalities for their ordering constraints:

$$v_j < v_{j+1} \Rightarrow 2b_{j,j+k} - v_{j+k} < 2b_{j+1,j+k+1} - v_{j+k+1}$$

Notice that in the above inequality both the value of the term on the left-side and the value of the term on the right-side *do not change* in case we add an additional new value to the left of the group G.

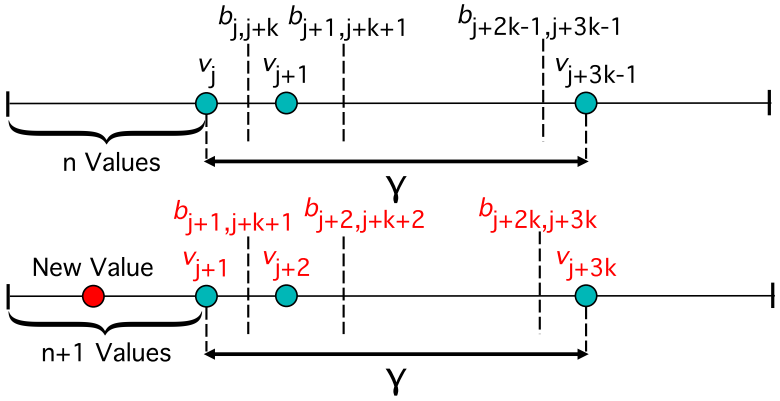


Figure 3.6: The addition of a single new value to the left of the group G does not change the location of the values and their bisectors. The only thing that changes is the labeling, i.e., sub-index, of the values and bisectors and is indicated in red.

Specifically, if we add a new value to the left of G the values v_{j+k} and v_{j+k+1} have now a different label (i.e., different subindex) but their location in $[\alpha, \beta]$ is the same. Similarly the bisectors $b_{j,j+k}$ and $b_{j+1,j+k+1}$ would also have a different subindex in case we add an additional new value to the left of G but the location of the involved values are the same. In other words the value of the left-term of the inequality (resp. right-term) is the same in case of $n+1$ it just *increases the subindices by one*.

We use this observation later in the proof.

Also due to the assumption that the statement holds for n we have:

$$v_j < v_{j+1} \Rightarrow (-1)^{\lfloor i/k-1 \rfloor}\xi_{j \bmod k} - (-1)^{\lfloor (i+1)/k-1 \rfloor}\xi_{j+1 \bmod k} < \gamma$$

, which comes from Lemma 12 and the fact that the left-term (resp. right-term) of the inequality $v_j < v_{j+1}$ can be expressed as a linear combination of the components of ξ and bisectors. We study how the subindices change in case we add a new value. For the case of n values to the left of G the non-zero components of ξ are: $\xi_{j \bmod k}$ for the left term and $\xi_{j+1 \bmod k}$ for the right term. In case we add a new value we have $n+1$ values to the left of G. Thus, the non-zero components of ξ are: $\xi_{j+1 \bmod k}$ for the left-term and $\xi_{j+2 \bmod k}$ for the right-term. So overall, for $n+1$ we have:

$$v_{j+1} < v_{j+2} \Rightarrow (-1)^{\lfloor (j+1)/k-1 \rfloor}\xi_{j+1 \bmod k} - (-1)^{\lfloor (j+2)/k-1 \rfloor}\xi_{j+2 \bmod k} < \gamma$$

, since in the previous paragraph we showed that *the value* of the left term is the same in both n and $n+1$ case (just a different labeling). Therefore, for the case of $n+1$ we can form a half-space that the left-side is a linear combination of ξ with two non-zero coefficients and the right-side is upper bounded by γ. Next we focus on the following ordering constraint for the case of n:

$$v_{j+k} < v_{j+k+1} \Rightarrow 2b_{j+k,j+2k} - v_{j+2k} < 2b_{j+k+1,j+2k+1} - v_{j+2k+1}$$

Similarly to before, notice that both the value on the left-side and the value on the right-side of the above inequality *do not change* in case we add an additional value to the left of the group G. Following the same reasoning for the case of $n+1$ we can form a half-space that the left-side is a linear combination of $\xi_{j+k+1 \bmod k} \equiv \xi_{j+1 \bmod k}$ and $\xi_{j+2k+1 \bmod k} \equiv \xi_{j+k+1 \bmod k}$ and the right-side is upper bounded by γ. Additionally the two discussed half-spaces are parallel since the coefficient of the ξ terms in $v_{j+k+1} < v_{j+k+2}$ are opposite from the coefficients in $v_{j+1} < v_{j+2}$, that is: $(-1)^{\lfloor (j+k+1)/k-1 \rfloor} = -(-1)^{\lfloor (j+1)/k-1 \rfloor}$ and $(-1)^{\lfloor (j+k+2)/k-1 \rfloor} = -(-1)^{\lfloor (j+2)/k-1 \rfloor}$.

If we follow the above reasoning and pair the ordering constraints for the case of n to the ordering

constraints for the case of $n+1$ we get:

$$v_j < v_{j+1} \leftrightarrow v_{j+k} < v_{j+k+1}$$
$$v_{j+1} < v_{j+2} \leftrightarrow v_{j+k+1} < v_{j+k+2}$$
$$\vdots$$
$$v_{j+k-1} < v_{j+k} \leftrightarrow v_{j+2k} < v_{j+2k+1}$$

Therefore we derive the same half-spaces for the case of $n+1$, just applied on differently labeled ξ terms. As a result for the case of $n+1$ we have a polytope in the k-dimensional space that has diameter that is upper bounded by 2γ. □

3.4 Evaluation of Approximate Reconstruction

In our evaluation, we test our reconstruction attacks on encrypted versions of databases, e.g. [101], that reduce their two-dimensional data to one dimension via Hilbert curves [125].

Mapping 2D Data to 1D via Hilbert Curves. Organizing *multidimensional data* for efficient access and indexing is a challenging problem due to the lack of a total ordering that preserves locality. *Space-filling curves* [125], which map points in a high-dimensional space onto one-dimensional points while preserving *locality* and proximity relations, have been thoroughly explored in spatial data management. See, e.g., [108, 130, 145, 161]. These curves essentially span the desired higher-dimensional space, with granularity tuned by the so-called *order* of the curve. The higher the order the better the approximation of locality. The second row of Figure 3.7 shows an example of a Hilbert curve of order 7 that spans a square in the two-dimensional space. In particular, this *single continuous* line of gray color that starts at (0,0) and ends at (0, 2^7) gives a $2^7 \times 2^7$ grid of points. A value of the DB in the two-dimensional space is projected to the closest *segment of the curve*. By "untangling" the curve we get a single straight line segment where all the projected values are within the boundaries $\alpha = 0$ and $\beta = 2^7 \times 2^7 = 2^{14}$. Conceptually, to run a (non-secure) k-NN query it is enough to traverse the one-dimensional points of DB towards the left and the right of the projections of the *query point* on the curve. Due to the properties of Hilbert curves, the set of k-NN on the one-dimensional space is an approximation of the neighbors in the two-dimensional space.

Dataset & Experiment Design. The dataset SpitzLoc [4], also used in [59], consists of the

latitude and longitude of the German Green party politician Malte Spitz over a period of six months. A record is stored whenever the phone was connected to a cell tower, received a call or sent a text. We used the two-dimensional data from the date ranges 1-5 Oct., 1-15 Oct., and 1-31 Oct., depicted in Figure 3.7. In all of our experiments we used a Hilbert curve of order 7 and placed the geolocation data in the center of the above Hilbert curve, the size n of each dataset is denoted in Table I. We simulated the k-NN query leakage of this setup and recorded the quality of the reconstruction for different values of $k = \{2, 5, 8\}$ and number of queries m. The quality measures are the Chebyshev distance between the original values and the reconstruction, denoted as AbsoluteError-1D, and the max Euclidean distance computed by inverting the mapping of the curve. Each of the above setup was repeated 20 times for m queries that were generated uniformly at random in $[\alpha, \beta]$. We note that we did not choose m based on the desired ϵ guarantee, but rather chose a value for m that is *orders of magnitude smaller* so as to demonstrate that the attack needs fewer samples than the derived bounds. The attack run on a commercial laptop and the code is written in Matlab. For the vertex enumeration problem, we use routines from the File Exchange of MathWorks [2].

We note here that not only size but also *the distribution of the data* plays a significant role in the success of our attack. This can be seen from the role of the $diam(\mathcal{F}_{[v]})$ in the quality of the reconstruction as well as in the statement of Theorem 7. Thus, even though the number of encrypted values in our experiments appears relatively small we can draw interesting conclusions due to values being highly concentrated.

3.4.1 Evaluation of Unordered Response Attack

Table I gives an overview of our experiments. In the right set of columns of Table I we present the accuracy of the reconstruction across all three datasets for *the same large* number of queries whereas in the left column we attempt to *significantly reduce* the number of observed queries without compromising the quality of the reconstruction. As it is expected, if the exact diameter of the reconstruction class is large then the reconstruction error is large as well. Interestingly, the diameter of the estimated feasible region, denoted as diameter est, is consistently close to the real one. Notice that for smaller number of samples we have smaller success percentage which means that no feasible region was found because the constraints were not approximated in a satisfactory accuracy. This trend does not appear when we increase the number of observed queries, i.e., $m = 800 \cdot 10^6$ on the right column. To visualize the accuracy, Figure 3.7 illustrates the reconstruction output for $m = 800 \cdot 10^6$

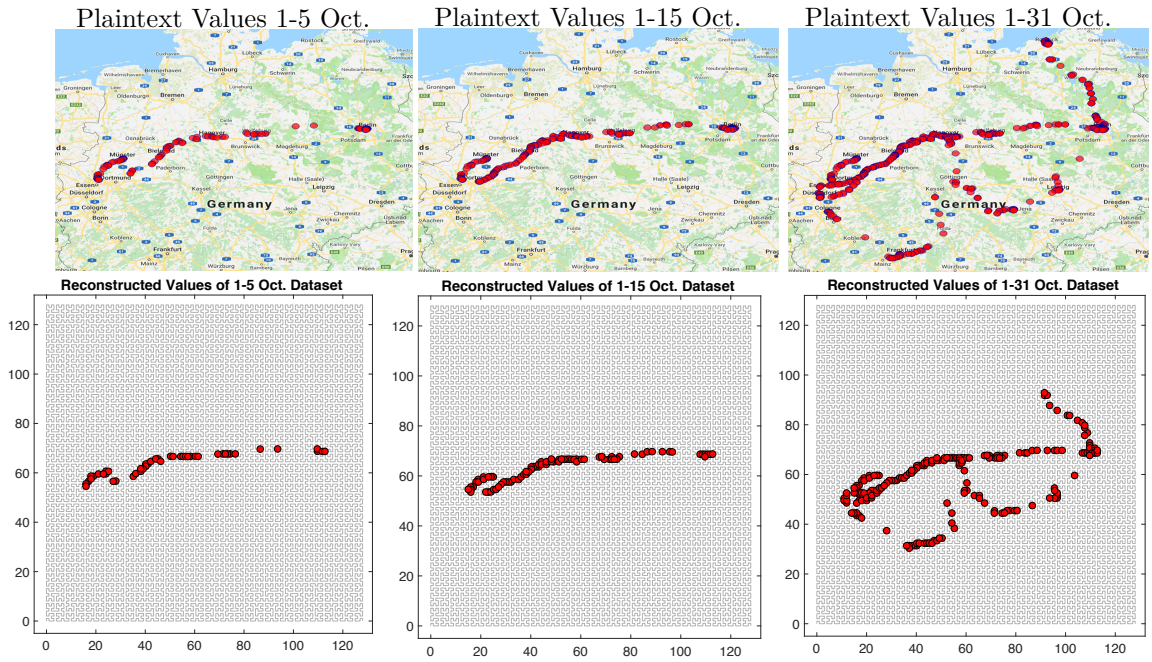


Figure 3.7: Three date ranges for the month October of the publicly available mobile records with the geolocation of the German Green party politician Malte Spitz. on the first row we demonstrate the original dataset and in the bottom the accuracy of the reconstruction that we achieve for unordered responses for $k = 2$.

and $k = 2$ across all three datasets. Overall, the approximate reconstruction is extremely accurate, from 0.003% to 0.1%, not only in one dimension but also in two-dimensions as well.

Table 3.1: Evaluation of AttackUnordered on the SpitzLoc dataset

	1-5 October, $m = 25 \cdot 10^6$, $n = 46$							1-5 October, $m = 800 \cdot 10^6$, $n = 46$						
	diameter		Abs. Error-1D		Rel. Error-1D	Abs. Error-2D	Success	diameter		Abs. Error-1D		Rel. Error-1D	Abs. Error-2D	Success
	exact	est	avg	std	avg	max		exact	est	avg	std	avg	max	
$k = 2$	1.8	1.1	3.6	1.1	0.02%	3.0	40%	1.8	1.7	0.5	0.1	0.003%	0.9	100%
$k = 5$	18.3	17.9	5.7	1.6	0.03%	5.0	80%	18.3	18.3	3.4	0.2	0.02%	2.9	100%
$k = 8$	79.9	78.3	16.9	1.4	0.1%	7.4	100%	79.9	79.5	14.6	0.15	0.09%	6.5	100%
	1-15 October, $m = 70 \cdot 10^6$, $n = 79$							1-15 October, $m = 800 \cdot 10^6$, $n = 79$						
	diameter		Abs. Error-1D		Rel. Error-1D	Abs. Error-2D	Success	diameter		Abs. Error-1D		Rel. Error-1D	Abs. Error-2D	Success
	exact	est	avg	std	avg	max		exact	est	avg	std	avg	max	
$k = 2$	1.9	0.8	1.8	0.7	0.010%	3.0	45%	1.9	1.4	0.6	0.1	0.003%	0.8	100%
$k = 5$	6.6	6.0	1.9	0.6	0.011%	2.5	80%	6.6	6.7	0.6	0.2	0.003%	1.3	100%
$k = 8$	15.4	14.6	2.5	0.6	0.015%	2.9	80%	15.4	15.1	1.0	0.1	0.006%	1.2	100%
	1-31 October, $m = 250 \cdot 10^6$, $n = 183$							1-31 October, $m = 800 \cdot 10^6$, $n = 183$						
	diameter		Abs. Error-1D		Rel. Error-1D	Abs. Error-2D	Success	diameter		Abs. Error-1D		Rel. Error-1D	Abs. Error-2D	Success
	exact	est	avg	std	avg	max		exact	est	avg	std	avg	max	
$k = 2$	1.8	1.0	1.0	0.2	0.006%	1.4	70%	1.8	1.1	0.7	0.1	0.004%	1.0	95%
$k = 5$	6.4	5.0	1.4	0.3	0.008%	2.0	95%	6.4	5.6	0.7	0.1	0.004%	1.1	100%
$k = 8$	12.8	11.6	1.4	0.3	0.008%	2.0	95%	12.8	12.2	0.8	0.2	0.004%	1.0	100%

A Visual Example of a Larger Dataset. In Figure 1.2, we present a visualization of the accuracy of our reconstruction for a larger dataset. On the left there is a picture of the Trojan horse

of 341×385 pixels, where each pixel is either black or white. To create a two-dimensional data set we sampled pixels until we collected $n = 1840$ black points that are uniquely mapped on a Hilbert curve of order 7. The middle subplot of Figure 1.2 shows how the two-dimensional plaintext values are mapped to the corresponding Hilbert curve. The feasible region for $k = 9$ has exact diameter 11.62. After observing $m = 10^9$ queries our attack successfully forms an approximation of the feasible region with diameter 7.03 which results into the reconstruction depicted in the right plot. The visual similarity is confirmed by the absolute error in 1-D which is 2.84, i.e., relative error 1-D of 0.01%. Even in two dimensions the absolute error is 6.15 which is equivalent to relative error in 2-D of 0.01%. For completeness we note that for this amount of queries the attack failed for $2 \leq k \leq 4$ and succeeded for $4 < k \leq 9$. This phenomenon is explained in the next paragraph.

Why Reconstruction for Small k is Harder. According to Table 3.1, the percentage of failures is significantly higher for $k = 2$. To better understand why smaller k values require tighter approximation guarantees, we note that by Lemma 13, each $c_{i,i+1}$ term of the ordering constraint $v_i < v_{i+1}$ consists of the sum of a number of lengths of Voronoi segments. Since our attack uses *estimations* of the above lengths, each term comes with its corresponding error, ϵ. Thus, a sum of 500 length terms introduces $500\,\epsilon$ error since the error *compounds*.

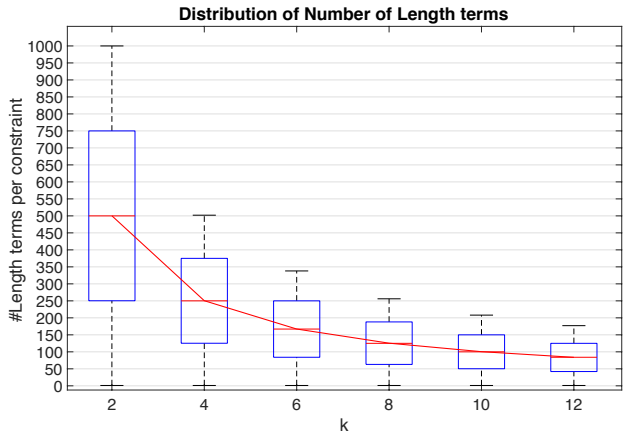


Figure 3.8: Distribution of length terms per ordering constraint for different values of k. The error compounds as the number of length terms per constraint grows.

In Figure 3.8 we fix $n = 2000$ and analyze *the number of length terms* involved in each of the $n - 1$ ordering constraints for varying values of k, independently of the DB. The minimum number of length terms for all k values is one and corresponds to the first constraint that has only a single length term involved. The average and maximum number of length terms are *inversely proportional*

to k. As a result, the combination of small diameter (i.e., concentrated values) of the tested dataset and the above sensitivity to the compound error results in a higher percentage of failures when k is small.

Table 3.2: Evaluation of AttackOrdered on the SpitzLoc dataset

	1-5 Oct., $m = 10^3$				1-5 Oct., $m = 25 \cdot 10^6$			
	Absolute Error-1D		Relative Error-1D	Absolute Error-2D	Absolute Error-1D		Relative Error-1D	Absolute Error-2D
	avg	std	avg	max	avg	std	avg	max
$k = 2$	436.8	200.7	2.6%	60.9	2.9	1.1	0.01%	4.1
$k = 5$	452.7	193.0	2.7%	54.7	2.9	1.3	0.01%	3.7
$k = 8$	480.2	146.1	2.9%	61.7	2.6	1.2	0.01%	4.1
	1-15 Oct., $m = 10^4$				1-15 Oct., $m = 25 \cdot 10^6$			
	Absolute Error-1D		Relative Error-1D	Absolute Error-2D	Absolute Error-1D		Relative Error-1D	Absolute Error-2D
	avg	std	avg	max	avg	std	avg	max
$k = 2$	147.4	49.2	0.8%	25.8	2.6	0.9	0.01%	3.0
$k = 5$	150.6	59.1	0.9%	26.0	2.8	1.2	0.01%	3.1
$k = 8$	150.0	56.9	0.9%	26.0	2.8	1.0	0.01%	3.5
	1-31 Oct., $m = 10^5$				1-31 Oct., $m = 25 \cdot 10^6$			
	Absolute Error-1D		Relative Error-1D	Absolute Error-2D	Absolute Error-1D		Relative Error-1D	Absolute Error-2D
	avg	std	avg	max	avg	std	avg	max
$k = 2$	45.2	15.9	0.2%	16.6	3.2	1.1	0.01%	3.9
$k = 5$	47.6	18.2	0.2%	15.4	3.2	1.2	0.01%	3.8
$k = 8$	50.3	17.7	0.3%	15.5	3.1	1.1	0.01%	3.9

3.4.2 Evaluation of Ordered Response Attack

Table 3.2 shows the accuracy of the approximate reconstruction attack for ordered responses. For this experiment, we simulated the query leakage by ordering the k returned ids. Note that the number of queries is significantly reduced. Since 1) the feasible region and its diameter does not play any role and 2) the estimation of each value is a function of only 3 bisectors, the quality of the reconstruction is *almost unaffected* by the value of k. Similarly to the case of unordered responses, the accuracy of the reconstruction grows significantly with the number of observed queries.

On Efficiency and Number of Queries. We report that having observed enough queries our experiments took a few seconds to reconstruction the plaintext values. We also report that for the accuracy, i.e., ϵ, δ, that we observed from the reconstructed output the theoretical lower bound of our theorems required orders of magnitude more queries. Therefore even though we have rigorous analysis for the required number of queries, our experiments demonstrate that we need *a significantly smaller number of queries in practice.*

Chapter 4

Attacks on Encrypted Databases Beyond the Uniform Query Distribution

In this chapter we propose the first reconstruction attacks on range queries and k-NN queries that reconstruct the underlying values *without any assumptions about the query or the data distribution.* Before our proposed attacks it was unclear whether such a general leakage-based reconstruction is possible given that all the previous attacks relied either on the uniform query distribution assumption or the assumption that the adversary knows both the query and the data distribution. Our generalization is achieved via a synergetic analysis of both access pattern and search pattern leakage and our experimental results demonstrate that an attacker can achieve accurate reconstruction under a wide variety of skewed query distributions and under various database densities without parametrizing the algorithms and without access to any auxiliary information.

4.1 Preliminaries

A *database* is a collection DB of n records $(id_i, val(id_i))$, $i = 0, \ldots, n-1$ where id_i is a unique identifier and $val(id_i)$ is a value from the universe $[\alpha, \beta]$. We assume discrete values so that α, β, and $val(id_i)$ are integers and denote with $N = \beta - \alpha + 1$ the size of the universe. For the sake of

simplicity of the analysis, we assume that the mapping from records to values is injective, that is, there is a single record in the database associated with a value. We call *density* of the database the fraction of values from the universe that are assigned to records. E.g., the *density assumption* studied by Lacharité *et al.* [106] corresponds to density 100%. A range query consists of two values $a \leq b$ from the universe and its response is the set of identifiers of the database records with values within interval $[a, b]$. A k-NN query consists of a value from the universe and its response is a set of k unordered identifiers, where k is fixed and decided at setup-time. We use the term *query* to refer to the plaintext query parameter(s) and the term *search token* to refer to the encrypted query parameter(s) that the client sends to the server. We define *access pattern leakage* the set of encrypted records that are retrieved as part of the response to a token. We define *search pattern leakage* the ability of the server to observe whether two tokens were generated from the same plaintext query. Although there are *response-hiding* STE schemes that minimize the access pattern leakage by imposing a storage overhead, the widely used constructions actually reveal the access pattern for the sake of efficiency. To the best of our knowledge, all structured encryption schemes leak the search pattern.

Assumptions. Our work makes *no assumptions* about the query distribution, data distribution, or access to any auxiliary information about them. Our assumptions are as follows:

- **Static Database.** No updates, i.e., addition, deletions, take place once the database is encrypted.
- **Fixed Query Distribution.** We assume that the adversary issues queries with respect to a fixed query distribution. We emphasize that our adversary does not know *any information about the family or the parameters of the query distribution.*
- **Correctness.** We consider schemes where the response to the issued query is correct. We do not consider schemes that return missing responses or false positive responses, e.g., Logarithmic-SRC [53] and "over-covers" from [63].
- **One-dimensional Data Values.** We do not address encrypted databases for high-dimensional data [42].
- **Known Setup.** We assume that the adversary knows the number of encrypted values n, the size of the universe of values N and the endpoints of the universe α, β.

- **Injective Mapping of Search Tokens.** We assume that distinct queries, can be either a pair of values like the range queries or a single value like the k-NN, map to distinct search tokens. We note that there are cases where search token are comprised by a *set of search tokens*, e.g., [53, 63]. In this case we view the output of the token generation algorithm as a *single token* by first imposing a canonical ordering on the set and then concatenating the tokens. We note here that the injective mapping is a property that is satisfied, to the best of our knowledge, by all known STE encryption schemes.

Order Reconstruction. There is a plethora of techniques [83, 104, 106] in the literature that reconstructs the order of the encrypted values using only the access pattern leakage. For simplicity of the exposition, we assume that the adversary can successfully reconstruct the order by using the appropriate algorithms from the above works and we instead focus on the problem of reconstructing the *plaintext values*. Thus, we treat the ordering as an input to out new value reconstruction algorithms and our techniques are not affected by how this ordering was constructed.

4.2 How to Exploit Search-Pattern Leakage

In this section, we introduce our main tool to reconstruct the plaintext values of an encrypted database *without any assumption about the data or query distribution*. Given a fixed query distribution, the repetition of search tokens, i.e., search-pattern leakage, reveals information about *the total number of search tokens* that return a specific encrypted response. This key observation relates our attack to the extensively studied problem of estimating the support size of a distribution.

We first show how to partition token-response pairs and interpret them as samples from the unknown query distribution. Next, we benchmark two widely-used non-parametric estimators under various query distributions. Finally, we propose a new modular estimator for our attack. Since we obtain a different estimator per encrypted response, the next section shows how to combine the acquired estimations to reconstruct the entire encrypted database.

4.2.1 Conditional Probability Distributions over the Leakage

In this subsection, we show how an adversary that is given a multiset of m token-response pairs $D = \{(t_1, r_1), \ldots, (t_m, r_m)\}$, can partition the tokens and analyze each group as a sample from a

conditional probability distribution. By conditioning on the information observed from the access pattern leakage, we group the information observed by the search pattern leakage.

Remark 2. *Let $D = \{(t_1, r_1), \ldots, (t_m, r_m)\}$ be the multiset of tokens and their corresponding response under an arbitrary token distribution. The mutliset of tokens with the same associated response, i.e., $D_i \coloneqq \{t_j | (t_j, r_i) \in D, r_i \subseteq \{id_0, \ldots, id_{n-1}\}\}$, is a* sample from the conditional probability distribution $p_{T|R}(T = t|R = r_i)$.

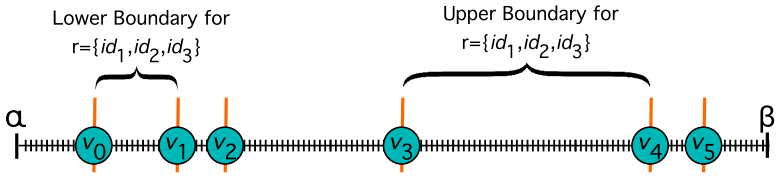


Figure 4.1: To observe response $r = \{id_1, id_2, id_3\}$ the start of the query range must be in-between v_0 and v_1 and the end must be in-between v_3 and v_4. Thus, the total number of queries that return r is $(v_1 - v_0) \cdot (v_4 - v_3)$.

Range Queries. We recall again our assumption that the mapping from range queries to tokens is injective. However, we note that our attack can be applied also to structured encryption schemes that generate *multiple tokens per query* with no false positives. In this scenario the attacker creates a canonical ordering of the collection of tokens, e.g., by lexicographical-ordering, and treats their concatenation as a single token. Schemes with this property include the BRC and URC token generation presented in [53], as well as the cover selection approach presented in [63]. The partition of the token-response pairs is performed with respect to a specific response. Consider a database with values $\{v_0, \cdots, v_{n-1}\}$ from a universe $[\alpha, \beta]$. Since we do not consider schemes with false positives, the number of *distinct* tokens that return a given response $r = \{id_i, \cdots, id_j\}$ is equal the product $(v_i - v_{i-1}) \cdot (v_{j+1} - v_j)$, where v_{-1} and v_n refer to α and β, respectively. An example is depicted in Figure 4.1.

Remark 3. *For the case of range queries on an encrypted database the support size of the conditional distribution $p_{T|R}(T = t|R = \{id_i, \ldots, id_j\})$, where $0 \leq i \leq j \leq n-1$, is the product of (1) the distance between values v_{i-1} and v_i and (2) the distance between values v_j and v_{j+1}, i.e., $(v_i - v_{i-1}) \cdot (v_{j+1} - v_j)$.*

k-NN Queries. A Voronoi diagram gives a natural partition of the query space for k-NN queries. It is known [104] that given a Voronoi diagram, the endpoints of each Voronoi segment correspond to

bisectors between the values.

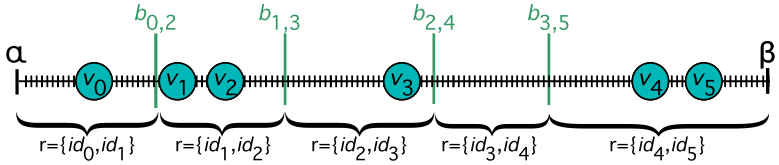


Figure 4.2: Voronoi diagram of a database with 6 values $v_0, \ldots, v_5$ and 2-NN queries. Short vertical black lines indicate distinct queries and tall vertical green lines indicate bisectors $b_{i,i+2}$ for values v_i and v_{i+2}.

Figure 4.2 shows the Voronoi diagram for 2-NN queries on a database DB with values $v_0, \ldots, v_5$ from range $[\alpha, \beta]$. The bisectors of the diagram, $b_{i,i+2}$, partition the query points into intervals where queries yield the same response. E.g., all query points between bisectors $b_{1,3}$ and $b_{2,4}$ yield response $\{v_2, v_3\}$. In our scenario of an encrypted database, the response is a pair of identifiers. Accordingly, we define the following partition of query tokens for k-NN queries: a search token t belongs to group D_i if its response is $\{id_i, \ldots, id_{i+k-1}\}$, for $i \in [0, n-k]$. We recall here our assumption of an injective mapping from queries to tokens, i.e., we never map two distinct queries to the same token. Therefore, the probability distribution on k-NN queries transfers to the probability distribution on tokens.

Let T be a random variable whose possible values are the tokens for k-NN queries generated by the client under an arbitrary token distribution. Let R be a random variable whose possible values are the k-NN responses with respect to DB.

Remark 4. *For the case of k-NN queries on an encrypted database, the support size of the conditional distribution $p_{T|R}\left(T = t | R = \{id_i, \ldots, id_{i+k-1}\}\right)$ is also the length of the corresponding Voronoi segment, i.e., $b_{i,i+k} - b_{i-1,i+k-1}$.*

4.2.2 Estimate Support Size of Each Distribution

In this subsection, we show how to utilize the *frequency* of the observed search tokens so as to estimate the number of possible search tokens that return a specific response r, i.e., estimate the support size of a conditional probability distribution with respect to r. In our approach, each response has a *different non-parametric estimator* that is "fine-tuned" for the specific conditional probability distribution. We focus on a *single response* but in the next section, we describe how an adversary can combine the estimations for different responses to achieve reconstruction. Furthermore, the estimation techniques described here are applied to both range and k-NN queries. To comply with

the notation in the literature [152] on support size estimators, in this subsection N denotes the support size of a single query distribution, whereas in the rest of the chapter N denotes the size of the universe of values, i.e., $N = \beta - \alpha + 1$.

Formulation. We assume a conditional probability distribution $p_{T|R}$ with respect to response r that contains N distinct search tokens observed with probabilities $\pi_i = (\pi_0, \ldots, \pi_{N-1})$. The adversary does not know the support size N or probabilities π_i. The main question we address is:

> *Given a sample D of m search tokens (with multiplicities) from $p_{T|R}$, what is the* ***total*** *number of search tokens in $p_{T|R}$ with non-zero probability?*

Let f_i be the number of search tokens that are observed i times in the sample. We briefly recall the terminology from [152]. The *fingerprint of sample D* is the vector $F = (f_1, f_2, \ldots, f_m)$, where $|D| = m$. Vector F is essentially the frequency of the frequencies. Then we can express the total number of *all distinct* search tokens as $N = f_0 + \sum_{i=1}^{m} f_i$ and the number of observed search tokens as $d = \sum_{i=1}^{m} f_i$. Similarly to [152], we call the *the histogram of the query distribution Q* over the elements of $p_{T|R}$ the mapping $h_Q : (0, 1] \to [0, N]$, where $h_Q(\pi)$ is the number of $p_{T|R}$ elements that occur in probability mass function Q with probability π. Notice that the fingerprint is defined according to a sample while the histogram is defined according to the query distribution.

One Experiment Captures Multiple Distributions. We call a distribution property *symmetric*, or label-invariant, if it only depends on the histogram of the distribution. Thus, a symmetric property does not depend on the *labeling* or the order of the search tokens. The following remark follows from Lemma 17 in [17].

Remark 5. *The* support size *of $p_{T|R}$ is a symmetric property.*

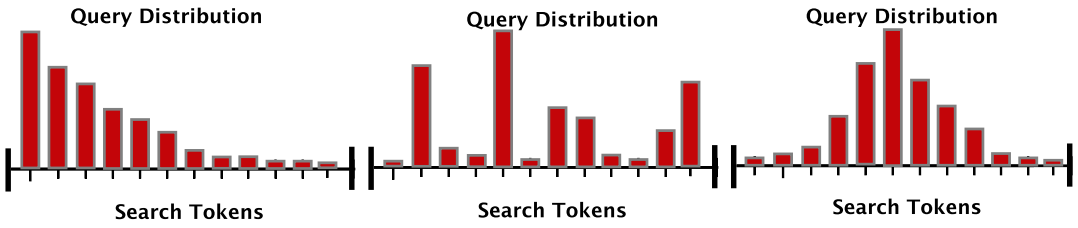


Figure 4.3: An illustration of three query distributions with the same histogram. The result of a support size estimation is the *same* in all three cases.

Jumping ahead, this important property comes into play in our evaluation. When we fix in our experiments the query distribution, we implicitly fix the conditional probability distributions too.

The symmetric property implies that from the point of view of the estimator, it makes no difference which token maps to which fixed probability value. Thus, the result of an experiment would be the same *for every assignment* of the chosen fixed probability values to tokens. As an example, the three pmfs presented in Figure 4.3 have *the same probability values* but different labelings. Since the fingerprint is the same, the support size estimation on the ordered "towers" on the left gives the same estimation as the pmf in the middle or the bell-shaped pmf to the right.

Related Work. The problem of estimating the support size of a distribution has appeared in several fields in different forms. Examples include the estimations of the number of English words Shakespeare knew [60], the number of species in a population of plants or animals [30], and how many dies were used on an ancient coin [148]. As reviewing this large body of work is beyond the scope of this work, we refer the reader to the following surveys [28, 39, 68]. We note that naive application of the estimators for the equiprobable case [86, 86, 111] to settings with varying probabilities has been shown to give an estimation with negative bias [111].

In our work, instead of deploying parametric estimators that assume an underlying family of distributions, we use a more general *non-parametric* approach that is *distribution agnostic*.

The Jackknife Method. *Resampling techniques* are non-parametric methods of statistical inference that draw repeated subsamples from the original sample D. In this work we are interested in the *jackknife method* originally proposed by Quenouille in [140]. In certain scenarios it is not known how to compute an efficient unbiased estimator of a statistic of interest generally denoted as θ. Therefore given a biased estimator $\hat{\theta}$ for a statistic the jackknife approach *estimates the bias* via sampling with replacement from D. An estimate of the bias $\widehat{\text{bias}}_{Jack}$ can be used to correct the estimator as follows:

$$\hat{\theta}_{Jack} = \hat{\theta} - \widehat{\text{bias}}_{Jack}.$$

The resampling approach of the jackknife is the following: to form a new sample we leave one observation out so as to create the subsample $D_{(i)} = (d_1, \ldots, d_{i-1}, d_{i+1}, \ldots, d_m)$. We denote as $\hat{\theta}_{(i)}$ the estimation of θ that is computed based on $D_{(i)}$. The term $\hat{\theta}_{(.)}$ denotes the average of all possible leave-one-out estimations, i.e., $\hat{\theta}_{(.)} = \sum_{i=1}^{m} \hat{\theta}_{(i)}/m$. The jackknife bias is defined as:

$$\widehat{\text{bias}}_{Jack} = (m-1)(\hat{\theta}_{(.)} - \hat{\theta}) = (m-1)(\frac{1}{m}\sum_{i=1}^{m} \hat{\theta}_{(i)} - \hat{\theta}).$$

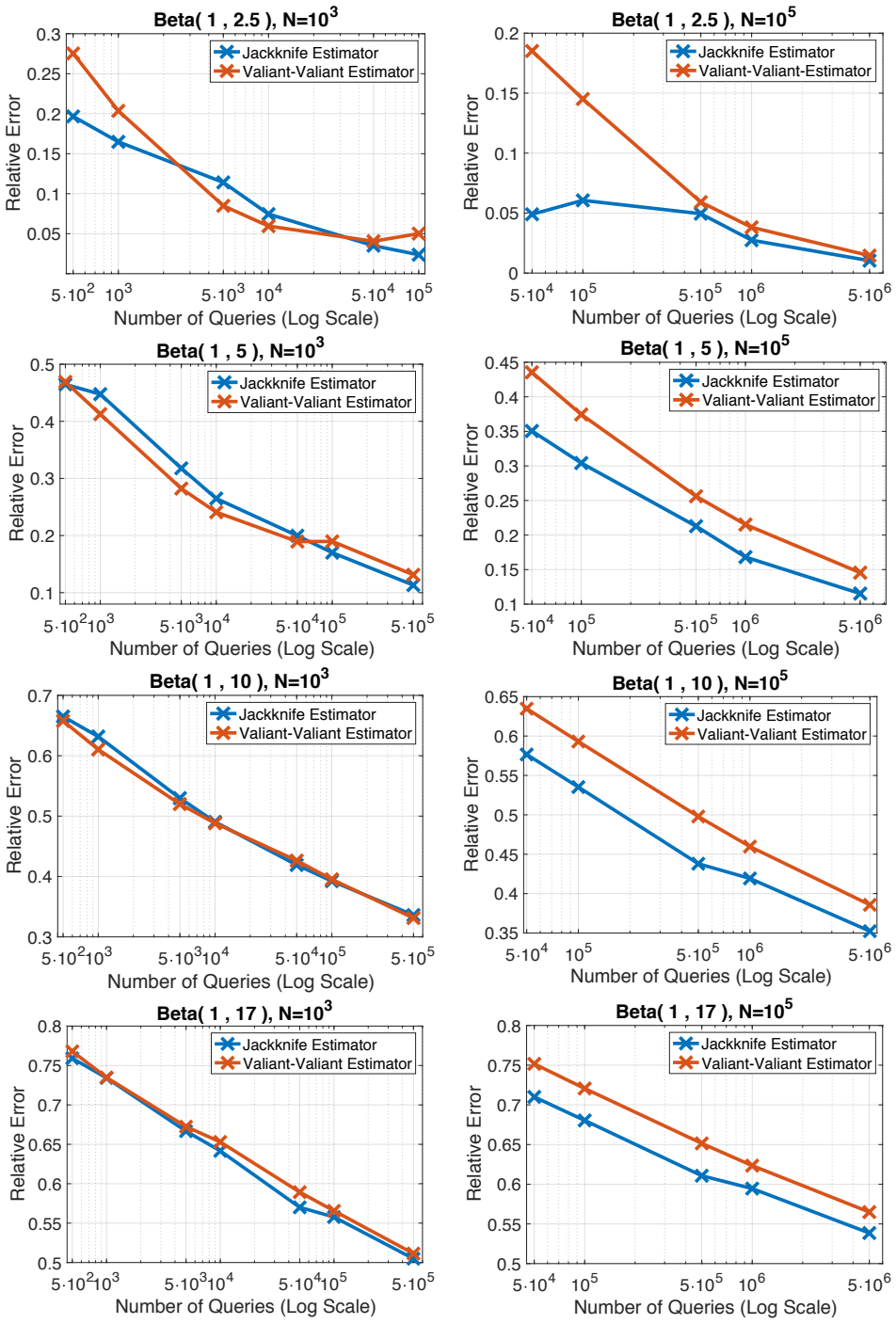


Figure 4.4: Comparison of estimators JACKKNIFE-SELFTUNE and VALIANT-VALIANT with respect to their relative error in support size estimation.

The multiplicative term $(m-1)$ in the above expression is rather counter-intuitive at first sight. One way to interpret this term is to assume that for a fixed m the expected value of the estimator $\hat{\theta}$ is

the estimand plus a bias term of the form bias $= b_1(\theta)/m$. In this case we get:

$$E[\widehat{\text{bias}}_{Jack}] = (m-1)\left(E[\hat{\theta}] - \frac{1}{m}\sum_{i=1}^{m} E[\hat{\theta}_{(i)}]\right)$$
$$= (m-1)\left(\theta + \frac{b_1(\theta)}{m} - \theta - \frac{b_1(\theta)}{m-1}\right) = \frac{b_1(\theta)}{m} = \text{bias}$$

Therefore the *expectation of the bias estimate* is the true formula of the bias. The above exposition concerns the *first order jackknife estimator* since it corrects biases of the order $O(1/m)$. This approach can be generalized to formulate the *k-th order jackknife estimator* that results in a bias of the order $O(m^{-k-1})$. There is an inherit trade-off between the bias and variance, the higher the order of the jackknife estimator the smaller the bias and the larger the variance. Our estimators come directly from the work of Burnham and Overton [29, 30] and where originally proposed for estimating animal populations. The statistic that we are interested in is the total number of distinct classes N. The initial biased estimator $\widehat{N}$ is the number of distinct classes *observed in sample* D, i.e., $\widehat{N} = d = \sum_{i=1}^{m} f_i$. The following expressions present the "bias-corrected" formula of the originally biased estimator $\widehat{N}$, the order of the jackknife describes the level of bias correction applied. For a fixed sample size m the jackknife estimator of order i is a simple linear combination of the fingerprint $F = (f_1, \ldots, f_m)$. That is the i-th order jackknife estimator can be expressed as:

$$\widehat{N}_{J(i)} = \sum_{k=1}^{m} \alpha_k^{(i)} f_k, \tag{4.1}$$

, where $\alpha_k^{(i)}$ coefficients are a function of the sample size m. The jackknife estimators for $\widehat{N}_{J(1)}, \widehat{N}_{J(2)}$, and $\widehat{N}_{J(3)}$ are:

$$\widehat{N}_{J(1)} = d + \tfrac{m-1}{m} f_1, \quad \widehat{N}_{J(2)} = d + \tfrac{2m-3}{m} f_1 - \tfrac{(m-2)^2}{m(m-1)} f_2,$$

$$\widehat{N}_{J(3)} = d + \tfrac{3m-6}{m} f_1 - \tfrac{(3m^2-15m+19)}{(m-1)m} f_2 + \tfrac{(m-3)^3}{(m-2)(m-1)m} f_3.$$

The derivation of the jackknife estimators $\widehat{N}_{J(i)}$ for $i \in [4, 10]$ appear in the next table, these analytical expressions may be of independent interest since they have not appeared before.

Selection of the Jackknife Order. Since we have we have the analytical expression of jackknife estimators $\widehat{N}_{J(i)}$, for $i \in [0, 10]$ an interesting question is how can we choose the appropriate order i given what we observed so far? To *tailor the order of the jackknife estimator* given the data in hand we deploy the order-selection technique originally proposed in [30] based on hypothesis

Jackknife Estimators.

$$\hat{N}_{J(4)} = d + \frac{4m-10}{m} f_1 - \frac{6m^2-36m+55}{(m-1)m} f_2 + \frac{4m^3-42m^2+148m-175}{m(m-1)(m-2)} f_3 - \frac{(m-4)^4}{(m-3)(m-2)(m-1)m} f_4$$

$$\hat{N}_{J(5)} = d + \frac{5m-15}{m} f_1 - \frac{10m^2-70m+125}{m(m-1)} f_2 + \frac{10m^3-120m^2+485m-660}{m(m-1)(m-2)} f_3 - \frac{(m-4)^5-(m-5)^5}{m(m-1)(m-2)(m-3)} f_4 + \frac{(m-5)^5}{(m-4)(m-3)(m-2)(m-1)m} f_5$$

$$\hat{N}_{J(6)} = d + \frac{6m-21}{m} f_1 - \frac{15m^2-120m+245}{m(m-1)} f_2 + \frac{20m^3-270m^2+1230m-1890}{(m-2)(m-1)m} f_3 - \frac{15m^4-300m^3+2265m^2-7650m+9751}{(m-3)(m-2)(m-1)m} f_4 + \frac{(m-5)^6-(m-6)^6}{(m-4)(m-3)(m-2)(m-1)m} f_5 - \frac{(m-6)^6}{(m-5)(m-4)(m-3)(m-2)(m-1)m} f_6$$

$$\hat{N}_{J(7)} = d + \frac{7m-28}{m} f_1 + \frac{-21m^2+189m-434}{m(m-1)} f_2 + \frac{35m^3-525m^2+2660m-4550}{(m-2)(m-1)m} f_3 + \frac{-35m^4+770m^3-6405m^2+23870m-33621}{(m-3)(m-2)(m-1)m} f_4 + \frac{21m^5-630m^4+7595m^3-45990m^2+139867m-170898}{(m-4)(m-3)(m-2)(m-1)m} f_5 + \frac{(m-7)^7-(m-6)^7}{(m-5)(m-4)(m-3)(m-2)(m-1)m} f_6 + \frac{(m-7)^7}{(m-6)(m-5)(m-4)(m-3)(m-2)(m-1)m} f_7$$

$$\hat{N}_{J(8)} = d + \frac{8m-36}{m} f_1 + \frac{-28m^2+280m-714}{(m-1)m} f_2 + \frac{56m^3-924m^2+5152m-9702}{(m-2)(m-1)m} f_3 + \frac{-70m^4+1680m^3-15260m^2+62160m-95781}{(m-3)(m-2)(m-1)m} f_4 + \frac{56m^5-1820m^4+23800m^3-156520m^2+517608m-688506}{(m-4)(m-3)(m-2)(m-1)m} f_5 + \frac{-28m^6+1176m^5-20650m^4+194949m^3-1029028m^2+2920008m-343615}{(m-5)(m-4)(m-3)(m-2)(m-1)m} f_6 + \frac{(m-7)^8-(m-8)^8}{(m-6)(m-5)(m-4)(m-3)(m-2)(m-1)m} f_7 + \frac{(-1)(m-8)^8}{(m-7)(m-6)(m-5)(m-4)(m-3)(m-2)(m-1)m} f_8$$

$$\hat{N}_{J(9)} = d + \frac{9m-45}{m} f_1 + \frac{-36m^2+396m-1110}{(m-1)m} f_2 + \frac{84m^3-1512m^2+9198m-18900}{(m-2)(m-1)m} f_3 + \frac{(1/120)(m-9)^9-(1/24)(m-8)^9+(1/12)(m-7)^9-(1/12)(m-6)^9+(1/24)(m-5)^9-(1/120)(m-4)^9}{(m-3)(m-2)(m-1)m} f_4 + \frac{(1/24)(m-9)^9-(1/6)(m-8)^9+(1/4)(m-7)^9-(1/6)(m-6)^9+(1/24)(m-5)^9}{(m-4)(m-3)(m-2)(m-1)m} f_5 + \frac{(1/6)(m-9)^9-(1/2)(m-8)^9+(1/2)(m-7)^9-(1/6)(m-6)^9}{(m-5)(m-4)(m-3)(m-2)(m-1)m} f_6 + \frac{(1/2)(m-9)^9-(m-8)^9+(1/2)(m-7)^9}{(m-6)(m-5)(m-4)(m-3)(m-2)(m-1)m} f_7 + \frac{-(m-8)^9+(m-9)^9}{(m-7)(m-6)(m-5)(m-4)(m-3)(m-2)(m-1)m} f_8 + \frac{(m-9)^9}{(m-8)(m-7)(m-6)(m-5)(m-4)(m-3)(m-2)(m-1)m} f_9$$

$$\hat{N}_{J(10)} = d + \frac{10m-55}{m} f_1 + \frac{-45m^2+540m-1650}{(m-1)m} f_2 + \frac{120m^3-2340m^2+15420m-34320}{(m-2)(m-1)m} f_3 + \frac{-210m^4+5880m^3-62370m^2+296940m-535227}{(m-3)(m-2)(m-1)m} f_4 + \frac{252m^5-9450m^4+142800m^3-1086750m^2+4164510m-6427575}{(m-4)(m-3)(m-2)(m-1)m} f_5 + \frac{-210m^6+10080m^5-202650m^4+2184000m^3-13306545m^2+43453200m-59411605}{(m-5)(m-4)(m-3)(m-2)(m-1)m} f_6 + \frac{120m^7-7140m^6+182700m^5-2606100m^4+22380120m^3-115700130m^2+333396850m-413066170}{(m-6)(m-5)(m-4)(m-3)(m-2)(m-1)m} f_7 + \frac{-(1/2)(m-10)^{10}+(m-9)^{10}-(1/2)(m-8)^{10}}{m(m-1)(m-2)(m-3)(m-4)(m-5)(m-6)(m-7)} f_8 + \frac{(m-9)^{10}-(m-10)^{10}}{m(m-1)(m-2)(m-3)(m-4)(m-5)(m-6)(m-7)(m-8)} f_9 + \frac{-(m-10)^{10}}{m(m-1)(m-2)(m-3)(m-4)(m-5)(m-6)(m-7)(m-8)(m-9)} f_{10}$$

testing. At a high-level this method tests the null hypothesis $H_i : E[\widehat{N}_{J(i+1)} - \widehat{N}_{J(i)}] = 0$ against $H_i' : E[\widehat{N}_{J(i+1)} - \widehat{N}_{J(i)}] \neq 0$ sequentially for $i \leq 10$ and choose the estimator $\widehat{N}_{J(i')}$ such that $H_{i'}$ is the first null hypothesis not rejected. We denote the above method for order selection as Jackknife-SelfTune.

The Valiant-Valiant Estimator. The work by Valiant and Valiant [152] introduced a framework for rigorously estimating the histogram of a discrete probability distribution from a sample. Since we are using the estimator from [152] as is, we limit our exposition into a high-level description of the estimator and its guarantees and we refer the reader to the original manuscript [152] for the detailed description. The Valiant-Valiant estimator takes as an input a sample from an unknown distribution, creates the fingerprint and then computes a plausible histogram that might have produced the observed fingerprint. Because there are numerous histograms that explain equally well the observed fingerprint the authors propose a method that picks the "simplest" among them.

Theorem 8. *(Corollary 1.12 [152]) There exist absolute positive constants ζ, γ such that for any $0 < \epsilon < 1$, there exists N_ϵ such that for any $N > N_\epsilon$, given a sample of search tokens D of size $m > \frac{\gamma}{\epsilon^2}\frac{N}{\log N}$ sampled from any query distribution π over the domain of $p_{T|R}$ of size $|p_{T|R}| = N$, the* Valiant-Valiant *estimator outputs a $\hat{N}$ such that*

$$\Pr(|N - \hat{N}| \leq N\epsilon) \geq 1 - e^{-N^{\zeta}},$$

provided none of the probabilities in π lie in $(0, \frac{1}{N})$.

It is worth noting that the above guarantees are bounds on the convergence rate and not essential parameters for the Valiant-Valiant estimator. The algorithm itself *does not depend* on any of the above parameters and its only input is a sample D of any size.

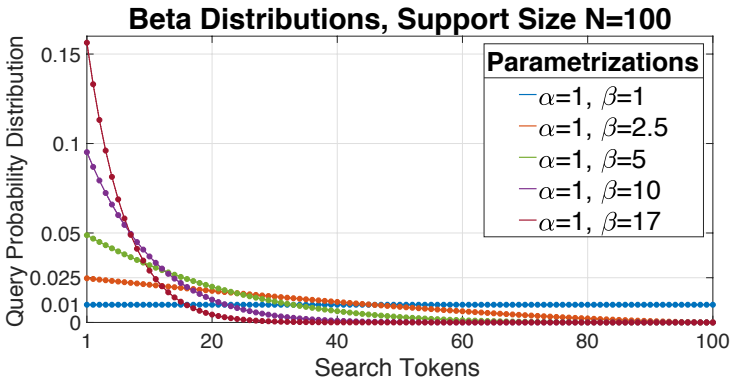


Figure 4.5: Evaluation of the estimators is conducted under various query distributions parameterized as a Beta probability mass function.

Evaluation of the Estimators. We conduct experiments to evaluate the performance of the estimators Valiant-Valiant and Jackknife-SelfTune. The only input that the two non-parametric estimators take is a sample form an unknown query distribution and based on the frequency of the search tokens they estimate the support size. We compute the relative error of the support size estimation under different settings:

- *Query distribution.* We deploy a discretized Beta probability distribution $Beta(\alpha, \beta)$ defined under parameterizations that take values $\alpha = 1$ and $\beta = \{1, 2.5, 5, 10, 17\}$.
- *Scale of support size.* Chosen from $N = 10^5$.
- *Number of observed search tokens.* Varying sample size.

We differentiate in our text between the α, β that denote the boundaries of the universe of values, see Section 4.1, from the α, β used for the *Beta* probability distribution by characterizing the latter as *parameters of the distribution*. Figure 4.5 shows the tested parameterizations of the Beta distribution. *Beta* is defined under continuous interval $[0, 1]$ which we discretized into N segments of equal length. Parameter $\beta = 1$ gives the uniform distribution, parameter $\beta = 2.5$ gives an almost linear decay. For parameter $\beta = 10$, we have roughly a power law, i.e., the Pareto principle, where roughly 80% of the mass is distributed among 20% percent of the search tokens. This behavior has been recorded in a lot of real-world phenomena. To give some more concrete statistics, for parameters $\beta = 2.5, 5, 10, 17$ the percentages of search tokens that: (a) have probability less than $1/N$ are $54\%, 67\%, 77\%, 84\%$ and (b) have probability less than $1/N^2$ are $0.5\%, 12\%, 36\%, 54\%$, respectively. For each parametrization we tested $5 \cdot 10^3$ instances and in Figures 4.4 and 4.6 we report the average absolute relative error. We recall that even though our experiments are conducted over a fixed family of distributions, e.g., the beta distribution, by Remark 5 our observations apply to *any permutation* of beta "towers" of probability mass and thus cover a wide range of query distributions. As it can be seen in Figure 4.4 estimator JACKKNIFE-SELFTUNE is more accurate than VALIANT-VALIANT in the majority of the tested settings. The above measurements experimentally confirm the guarantees of Theorem 8 since a sublinear number of queries is enough to predict the existence of unobserved search tokens except the ones that have probability less than $1/N$. Another observation is that the maximum tested number of observed search tokens, i.e., $500N$, resulted in a relative error that is close to the percentage of search tokens with probability less than $1/N^2$.

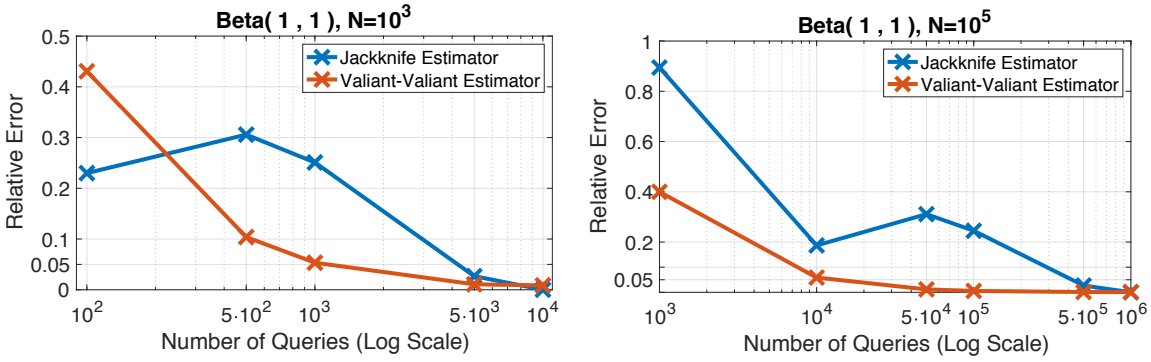


Figure 4.6: Comparison of estimators under uniform query distribution.

Interestingly, for the case of uniform query distribution the VALIANT-VALIANT estimator is *significantly more accurate* when the number of samples is sublinear. Based on this observation

we propose a "modular-estimator" to achieve the best of both worlds, an agnostic non-parametric estimator that deploys (1) the VALIANT-VALIANT when the query distribution is uniform and (2) the JACKKNIFE-SELFTUNE otherwise.

Algorithm 6: MODULAR-ESTIMATOR

Input: Multiset of m search tokens D sampled according to $p_{T|R}$

Output: Estimation of the support size $\hat{N}$

Deploy VALIANT-VALIANT estimator with input D and get $\hat{N}_V$;

Compute number of collisions $c \leftarrow |\{j < k : k \in [2, m], t_j = t_k\}|$;

Set the error parameter for the tester $\epsilon \leftarrow 1/\hat{N}_V$;

if $c/\binom{m}{2} \leq (1 + 2\epsilon^2)/\hat{N}_V$ **then** `// collision prob. tester`

  return $\hat{N}_V$ *since it passed the tester*

end

`// Deploy JACKKNIFE-SELFTUNE;`

Set number of unique tokens based on fingerprint $d \leftarrow \sum_{i=1}^{m} f_i$;

for $i \leftarrow 1$ ***to*** 9 **do**

  Set $b_k \leftarrow \alpha_k^{(i+1)} - \alpha_k^{(i)}$, where $\alpha_k^{(i)}$ is the k-th coefficient of the jackknife estimator of order i, see Equation (4.1);

  $\hat{N}_{J(i+1)} - \hat{N}_{J(i)} \leftarrow \sum_{k=1}^{m} b_k f_k$ `// Eq. (4.1);`

  $\widehat{\text{var}}(\hat{N}_{J(i+1)} - \hat{N}_{J(i)}|d) \leftarrow \frac{d}{d+1}\left(\sum_{k=1}^{m}(b_k)^2 f_k - \frac{\left(\hat{N}_{J(i+1)} - \hat{N}_{J(i)}\right)^2}{d}\right)$;

  Formulate the test statistic $T_i \leftarrow \frac{\hat{N}_{J(i+1)} - \hat{N}_{J(i)}}{\sqrt{\widehat{\text{var}}(\hat{N}_{J(i+1)} - \hat{N}_{J(i)}|d)}}$ for the null hypothesis $H_i : E[\widehat{N}_{J(i+1)} - \widehat{N}_{J(i)}] = 0$;

  Since T_i follows approximately a standard distribution, we can derive its corresponding two-sided significance level, denoted as P_i;

  if $P_i > 0.1$ **then**

    return $\hat{N}_{J(i)}$ *since the null hypothesis* H_i *is not rejected*

  end

end

Modularity via Property Testing. Our estimator is Algorithm 6 (MODULAR-ESTIMATOR). The work of Goldreich and Ron [76] introduced a property testing [74] technique called *collision-probability tester* that given a sample from an unknown distribution it tests whether the sample originated from a distribution that is ϵ-afar from the uniform over $[1, N]$. Recent analysis by Diakonikolas *et al.* [57] showed a tight upper bound on the sample complexity of $O(\sqrt{N}/\epsilon^2)$ which proves sample-optimality. The collision-probability tester takes as parameters the desired error ϵ the sample D and the support size N as an input. Unfortunately in our setup we do not know

N therefore in our algorithm we use the output of VALIANT-VALIANT as an approximation $\hat{N}$ to perform the collision-probability tester. Our approach is modular in the sense that different modules, i.e., estimators, are used for different "shapes" of query distributions. For concreteness we chose 0.1 as the threshold of the significance level of hypothesis testing, per recommendation of [30], and a fixed error ϵ for the collision-probability tester but these quantities can be tuned differently.

4.3 Revisiting Data Reconstruction Attacks

In this section, we use the techniques from Section 4.2 to develop new reconstruction attacks on encrypted databases using *both* the search-pattern leakage and access-pattern leakage. Our reconstruction algorithm for range queries (Section 4.3.1) is significantly different from previous approaches. Our reconstruction algorithm from k-NN queries (Section 4.3.2) builds on previous work [104] but follows a different algorithmic strategy so as to reduce the number of required samples and also scale for larger values of k. We experimentally demonstrate the accuracy of our reconstruction algorithms under various query distributions and densities of the database.

4.3.1 Reconstruction from Range Queries

Illustrative Example. We start by conveying the intuition of our range attack with an application on a simple database with only three values, $\{v_0 = 7, v_1 = 15, v_2 = 20\}$ from universe $[1, 30]$ shown in Figure 4.7. The distances between consecutive pairs, $L_i = v_i - v_{i-1}$, are $L_0 = 7$, $L_1 = 8$, $L_2 = 5$, $L_3 = 11$.

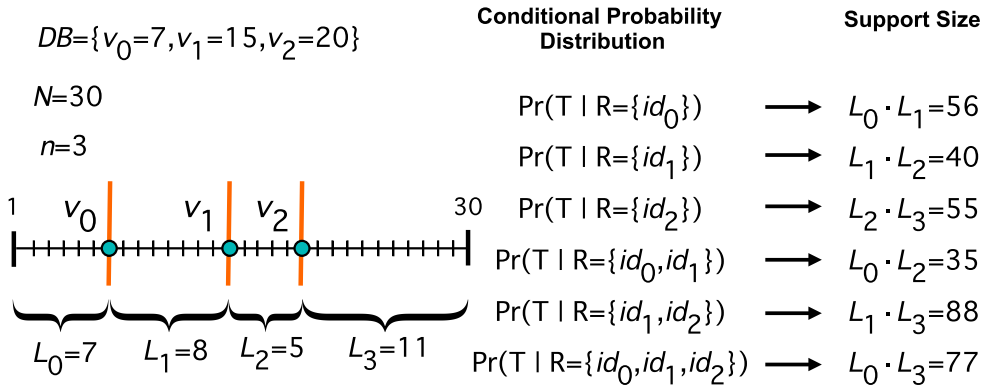


Figure 4.7: Illustrative example of a database along with all the possible conditional probability distributions and their corresponding support size.

For simplicity, we consider first the restrictive scenario where the adversary has observed all possible range queries. In this case, there is *no need to estimate* the number of range-queries that return a specific response r', it is enough to count the number of unique queries that return r'. In other words, the adversary knows the exact support size for every conditional probability distribution $p_{T|R}(T|R = r')$. From Remark 3, the support size can be expressed as the product L_i, L_j for the appropriate pair i, j. The support sizes of all conditional distributions of this example are illustrated in Figure 4.7.

To compute the $n + 1$ unknowns L_0, L_1, L_2, L_3, the adversary solves the following set of $\binom{n}{2}$ equations:

$$\begin{aligned} L_0 \cdot L_1 = 56 \qquad L_1 \cdot L_2 = 40 \qquad L_2 \cdot L_3 = 55 \\ L_0 \cdot L_2 = 35 \qquad L_1 \cdot L_3 = 88 \qquad L_0 \cdot L_3 = 77 \end{aligned} \tag{4.2}$$

One can apply the logarithmic function to transform the products to sums, i.e., $x_0 = \log(L_0), x_1 = \log(L_1), x_2 = \log(L_2), x_3 = \log(L_3)$. Then, using elementary row operations on the system of linear equations one can easily compute the echelon form and show that the rank of the matrix is $n + 1$, thus there is a unique and exact reconstruction for the restrictive scenario where the adversary has seen all possible queries. The above analysis was also presented independently in [117].

We now consider the more realistic, general scenario of an adversary who has observed a *subset* of all possible search tokens, as issued by the client under a fixed query distribution that is *unknown* to the adversary. From Observation 2, a token-response pair, (t', r'), can be seen as a *sample from the conditional probability distribution* $p_{T|R}(T|R = r')$. Thus, the first step of the attack is to partition the observed search tokens with respect to their returned responses, i.e., the conditional distribution they belong to, using the method of Section 4.2.

The result of this partition gives a collection of multisets of search tokens. Each multiset is used to *estimate* the support size of the corresponding distribution. We denote with $\widehat{L}_{i,j}$ the *estimation* of the support size $L_i \cdot L_j$. We note here that some estimations should play a more central role in the overall reconstruction based on the fact that we have *observed more samples*. For example, the support size estimation of $p_{T|R}(T|R = r')$ from a sample of size 2 is less trustworthy than the support size estimation of $p_{T|R}(T|R = r'')$ from a sample of size 10^3. To capture this we model a minimization problem, where the "importance" of an estimate $\widehat{L}_{i,j}$ is expressed by a non-negative weight $w_{i,j}$.

Reconstruction Algorithm. The goal of the proposed optimization is to assign values to the lengths $L_0, \ldots, L_n$ so as to minimize the weighted sum of squared errors. One option for the error function between the estimated products and a pair of lengths is the difference between the two terms, i.e., $e_1(L_i, L_j) = (L_i \cdot L_j - \widehat{L}_{i,j})$. Another option for the error function is the log of the ratio, that is $e_2(L_i, L_j) = \log\left((L_i \cdot L_j)/\widehat{L}_{i,j}\right) = \log(L_i) + \log(L_j) - \log(\widehat{L}_{i,j})$. If there is no sample to feed to the estimator to produce $\widehat{L}_{i,j}$, we assign default value $\widehat{L}_{i,j} = 1$, therefore the ratio in e_2 is well-defined since the denominator takes positive non-zero values. Notice that both e_1 and e_2 output value 0 when the product of the unknowns is equal to the estimated quantity $\widehat{L}_{i,j}$. Our experiments showed that the error function of the log ratio $e_2(L_i, L_j)$ has *superior reconstruction quality* in the majority of the cases compared to the error function $e_1(L_i, L_j)$. For simplicity of the exposition we define new unknowns $x_i = \log(L_i)$ for $i \in [0, n]$ which gives the following final unconstraint optimization problem:

$$\min_{x_0,\ldots,x_n} \sum_{i=0}^{n} \sum_{j=i+1}^{n} w_{i,j}(x_i + x_j - \log(\widehat{L}_{i,j}))^2 \tag{4.3}$$

We set weight $w_{i,j} = \max\{\epsilon, |D_{i,j}|\}$, where ϵ is an arbitrarily small positive value and $|D_{i,j}|$ is the number of tokens used for estimation $\widehat{L}_{i,j}$. The values $x_0, \ldots, x_n$ obtained from the solution of (4.3) are mapped to lengths as $L_i = 2^{x_i}$. As a final step we scale the derived lengths $L_0, \ldots, L_n$ to sum to N.

Theorem 9. *The unconstrained quadratic optimization problem of Equation (4.3) with constant values $w_{i,j}, \widehat{L}_{i,j}$, and unknown values x_i, is a convex function and has a unique solution.*

Proof. We first show that the function in equation (4.3) is convex. Notice that the inner functions, i.e., $(x_i + x_j - \log(\widehat{L_i L_j}))$, can be interpreted as convex functions, i.e., strict equality in the definition of convexity, and their composition with the quadratic function, i.e., $(x_i + x_j - \log(\widehat{L_i L_j}))^2$, output a convex function. Furthermore it is known that the non-negative weighted sum of convex function preserves convexity which means that the function of (4.3) is convex. Due to convexity of (4.3) every local minima is global. The next step is to show that there exists a unique solution. Notice that the following matrix of coefficients M derived from the partial derivatives, also appears in Line 8 of

Algorithm Agnostic-Reconstruction-Range, and is symmetric.

$$M = \begin{bmatrix} \sum_{j\neq 0} 2w_{0,j} & 2w_{0,1} & \dots & 2w_{0,n} \\ 2w_{0,1} & \sum_{j\neq 1} 2w_{1,j} & \dots & 2w_{1,n} \\ \dots & \dots & \dots & \dots \\ 2w_{0,n} & 2w_{1,n} & \dots & \sum_{j\neq n} 2w_{j,n} \end{bmatrix}$$

If we show that $\vec{y}^T M\vec{y} > 0$ for all vectors $\vec{y} \neq 0$, then the matrix is positive definite which implies that there is a unique solution. We want to show $\vec{y}^T My > 0$. Thus:

$$\vec{y}^T My > 0 \Rightarrow 2\vec{y}^T \tfrac{1}{2} My > 0 \Rightarrow \vec{y}^T \tfrac{1}{2} My > 0 \Rightarrow$$

$$\vec{y}^T \cdot \begin{bmatrix} \sum_{j\neq 0} w_{0,j} & w_{0,1} & \dots & w_{0,n} \\ w_{0,1} & \sum_{j\neq 1} w_{1,j} & \dots & w_{1,n} \\ \dots & \dots & \dots & \dots \\ w_{0,n} & w_{1,n} & \dots & \sum_{j\neq n} w_{j,n} \end{bmatrix} \cdot \vec{y} > 0 \Rightarrow$$

$$\begin{bmatrix} y_0 \left(\sum_{j\neq 0} w_{0,j}\right) + \sum_{j\neq 0} y_j w_{0,j} & \dots & y_n \left(\sum_{j\neq n} w_{j,n}\right) + \sum_{j\neq n} y_j w_{j,n} \end{bmatrix} \cdot \vec{y} > 0 \Rightarrow$$

$$\sum_{i=0}^{n} \left(y_i^2 \left(\sum_{j\neq i} w_{i,j}\right) + y_i \sum_{j\neq i} y_j w_{i,j} \right) > 0 \Rightarrow$$

$$\sum_{0\leq i<j\leq n} \left(w_{i,j}(y_i^2 + y_j^2)\right) + \sum_{0\leq i<j\leq n} \left(2y_i y_j w_{i,j}\right) > 0 \Rightarrow$$

$$\sum_{0\leq i<j\leq n} w_{i,j} \left(y_i^2 + y_j^2 + 2y_i y_j\right) > 0 \Rightarrow$$

$$\sum_{0\leq i<j\leq n} w_{i,j} \left(y_i + y_j\right)^2 > 0$$

, which is true since w_i, j is always positive. □

We derive the partial derivative with respect to x_i as:

$$\frac{\partial f}{\partial x_i} = \left(\sum_{j\neq i} 2w_{i,j}\right) x_i + \sum_{j\neq i} \left(2w_{i,j}\right) x_j - \left(\sum_{j\neq i} 2w_{i,j} \log(\widehat{L}_{i,j})\right).$$

We find the global minimum by setting all partial derivatives equal to zero. Our reconstruction method from range queries, RANGE-RECONSTRUCTION, is shown in Algorithm 7.

Comparison with Attack GeneralizedKKNO [83]. We first compare the accuracy of the

Algorithm 7: AGNOSTIC-RECONSTRUCTION-RANGE

Input: Multiset of range search tokens and their responses $D = \{(t_1, r_1), (t_2, r_2) \ldots, (t_m, r_m)\}$; ordering of the database records $I = (id_0, \ldots, id_{n-1})$; endpoints α and β of the database universe; arbitrary positive constant ϵ

Output: Approximate reconstruction $\tilde{v}_0, \ldots, \tilde{v}_{n-1}$

for *every unique response* r *in* D **do**
  Let $id_i \in r$ be the identifier of r with minimum rank in I;
  Let $id_j \in r$ be the identifier of r with maximum rank in I;
  Let $D_{i,j+1}$ be the mulitset of all the pairs in D with response r;
  Let weight $w_{i,j+1} = \max\{\epsilon, |D_{i,j+1}|^2\}$;
  Run Algorithm 6 (MODULAR-ESTIMATOR) on the multiset of search tokens in $D_{i,j+1}$ to output estimated support size $\widehat{L}_{i,j+1}$;
end
Solve the system of linear equations below, obtained by setting the partial derivatives of Eq. (4.3) equal to zero:

$$\begin{bmatrix} \sum_{j\neq 0} 2w_{0,j} & 2w_{0,1} & \ldots & 2w_{0,n} \\ 2w_{0,1} & \sum_{j\neq 1} 2w_{1,j} & \ldots & 2w_{1,n} \\ \ldots & \ldots & \ldots & \ldots \\ 2w_{0,n} & 2w_{1,n} & \ldots & \sum_{j\neq n} 2w_{j,n} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \ldots \\ x_n \end{bmatrix} = \begin{bmatrix} \sum_{j\neq 0} 2w_{0,j} \log(\widehat{L}_{0,j}) \\ \sum_{j\neq 1} 2w_{1,j} \log(\widehat{L}_{1,j}) \\ \ldots \\ \sum_{j\neq n} 2w_{n,j} \log(\widehat{L}_{n,j}) \end{bmatrix}$$

Compute the approximated lengths as $L_0 = 2^{x_0}, \ldots, L_n = 2^{x_n}$;
Scale $L_0, \ldots, L_n$ so as $\sum_{i=0}^{n} L_i = \beta - \alpha + 1$;
Let $v_{-1} = \alpha - 1$;
for $i = 0, \cdots, n-1$ **do**
  Let $\tilde{v}_i = \tilde{v}_{i-1} + L_i$;
end
return $\tilde{v}_0, \ldots, \tilde{v}_{n-1}$;

reconstruction of our attack, AGNOSTIC-RECONSTRUCTION-RANGE, to the accuracy of the state-of-the-art reconstruction attack GENERALIZEDKKNO, which is the the most general (i.e., with fewest assumptions, e.g., only uniform queries) of the three attacks proposed by Grubbs *et al.* [83]. In this experiment, we generate $Q = 10^4$ range queries uniformly at random from the values in universe $[\alpha, \beta] = [1, 10^3]$. We generate the values of the encrypted DB under various database densities. To assess the quality of the reconstruction, we use the *mean square error* (MSE) and the mean of absolute error (MAE) between the original and the reconstructed database. We note here that MSE gives a *higher penalty to reconstructed values with larger error.*

Recall that our algorithm is (1) not tailored to work well on a specific query or data distribution and (2) distribution agnostic, i.e., does not need to know the data/query distribution. Hence, we would expect GENERALIZEDKKNO to have an inherent advantage in this experiment since it is specifically designed for uniform queries. The results of the experiment, shown in Figure 4.8, indicate that this is not the case: in terms of MSE, GENERALIZEDKKNO is 2.5× to 17× worse than our AGNOSTIC-RECONSTRUCTION-RANGE for densities from 20% to 90%, and in terms of MAE GENERALIZEDKKNO is comparable with our AGNOSTIC-RECONSTRUCTION-RANGE for densities from 15% to 75%.

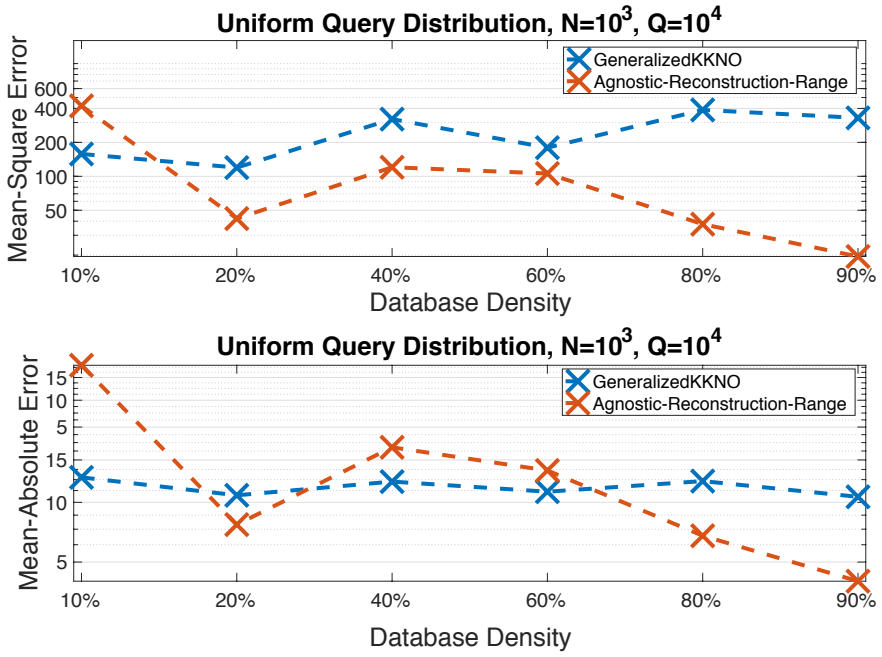


Figure 4.8: Comparison between GeneralizedKKNO and our attack, Agnostic-Reconstruction-Range, under the uniform query distribution.

We explain the experimental results as follows. The MAE quality metric is a first order statistic, therefore the large errors of GeneralizedKKNO are not penalized enough in the bottom plot of Figure 4.8. To explain why the performance of GeneralizedKKNO deteriorates, we note that this algorithm essentially maps the observed frequency to an expected frequency if the record were to have a fixed value. For dense databases, several records will appear together in many responses and as a result, will have very similar frequencies. This implies that multiple records map to the same plaintext value. The experiments confirm this behavior as GeneralizedKKNO tends to map multiple records to the same reconstructed value. To explain the outperformance of GeneralizedKKNO in the sparse regime, recall that the support size of each conditional distribution is *the product* of a pair of distances between database values. When the database is sparse, such distances are larger, hence the support size grows quadratically with the distance. Thus, the adversary needs to see more samples than the tested ones to increase accuracy.

Evaluation on Short Range Queries. In practical data analysis applications, focused short range queries are more likely to occur than exploratory long queries. Also, a client who often issues long range queries would have limited benefits from outsourcing the database. Motivated by this observation, we have conducted experiments on short range queries. First, we explain how we

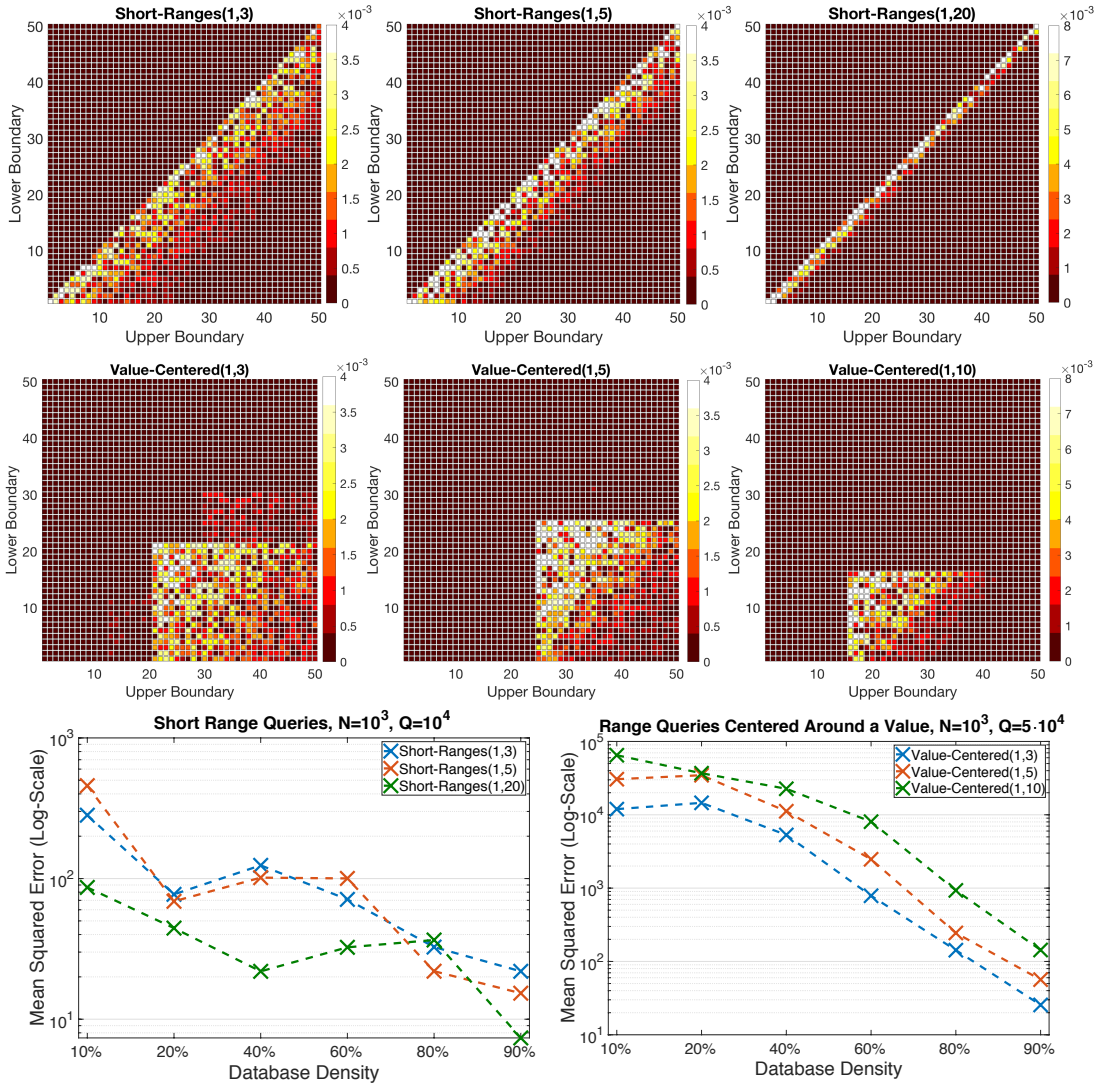


Figure 4.9: Performance of our attack, AGNOSTIC-RECONSTRUCTION-RANGE, under parameterizations of query distributiona Short-Ranges and Value-Centered.

generate short range query distributions and then we report on the experimental results.

Let $|R|$ be the number of all possible range responses. Specifically we generate a query distribution that we call Short-Ranges(α, β) as follows: Generate a Beta(α, β) distribution and discretize into $|R|$ equally spaced intervals. Recall that the cardinality of the universe of values is N. Then process the discretized values from left-to-right and add "noise" by multiplying each probability with a random number from $[0, 1]$ divided by $|R|$. After applying a normalization step, assign *in batches* the "noisy" Beta(α, β) probabilities to queries as follows: assigns the first N probabilities to queries whose range is a single value; assign the next $N-1$ probabilities to queries whose range spans two values; continue up to the range query spanning the entire universe. This process gives higher probability to short

range queries. The higher the value of parameter β, the larger the gap between the probabilities of short and long range queries. To understand how different Short-Ranges is from the uniform we note that for $N = 10^3$, the mean length of a sampled range query under the uniform is 333, which corresponds to 33% of the universe size. The query distributions Short-Ranges(1,3), Short-Ranges(1,5), and Short-Ranges(1,20) have mean length of 142, 90, and 23, which correspond to $14.2\%, 9\%$, and 2% of the universe, respectively.

In this evaluation, we chose parameter $\beta = \{3, 5, 20\}$ and $N = 10^3$. The upper row of Figure 4.9 shows the heatmap of the probability distribution for these three parameterizations but for a smaller universe. The Y-axis, resp. X-axis, corresponds to the lower boundary, resp. upper boundary, and the coloring of each square represents the probability of issuing this range query. As one can see the "bright" high-probability areas are around the diagonal. The MSE plot in Figure 4.9 shows the behavior of Agnostic-Reconstruction-Range under different database densities. The distribution Short-Ranges(1,20) is a case where one would expect the reconstruction algorithm to be challenged due to the fact that only a few records are returned in each response. Interestingly, our reconstruction in Short-Ranges(1,20) is *significantly better* than the other distributions. To explain this, recall that the length of the range queries is really small which implies that the adversary only observes a small number of responses. So even though a lot of the total $\binom{N+1}{2}$ conditional probability distributions will not observe any search token the small number of conditional distributions that are "active" will observe a enough samples to get a very *accurate* estimation of their corresponding support size. The final step of the formulated convex optimization problem in Equation (4.3) combines the accurate estimations efficiently to derive the overall assignment of reconstructed values.

Evaluation on Queries Centered Around a Value. In this experiment we focus on range queries that are centered around a given value. Consider the real-world scenario of an encrypted database with medical data and assume that the client is a researcher who analyzes the medical profile of adolescents with asthma symptoms. We expect the majority of range queries issued by the client on the age attribute to have values within or near range $[13, 19]$ since this is the population of interest.

Inspired by the above scenario, we generate query distributions that we call Value-Centered(α, β), i.e., *tailored to contain a specific value* of the encrypted database. Similar to the generation process of the Short-Ranges query distributions, we discretize a beta pdf and multiply each probability of the pmf with a random number from $[0, 1]$ divided by $|R|$ and as a final step, we normalize. The

difference from the previous experiment is how we assign the resulting probabilities to range queries. For Value-Centered we choose uniformly at random a value v'_1 of the underlying database. Processing again the probabilities-to-be-assigned from left to right, we assign the first batch of probabilities to the *range queries that return the chosen value* v'_1. As the next step, we sample without replacement another value v'_2 from the database and assign the next batch of probabilities to the ranges that return v'_2. This process continues until we have processed all n database values and we finally assign the remaining probabilities to the remaining range queries. The lower row of Figure 4.9 shows the heatmap of these distribution. The "bright rectangles" show that the range queries are "centered" around the value on the upper left corner of the rectangle. The ranges generated with Value-Centered(1,3) better explore the universe of values which allows our reconstruction attacks to achieve the smallest reconstruction error. The case of Value-Centered(1,10) assigns most of the high probabilities to a subset of the ranges that contain a single value therefore the majority of the universe is rarely explored. We report here that 14 out of the 120 runs of the Value-Centered(1,10) were unsuccessful because the queries did not explore the universe sufficiently. In general the query distribution Value-Centered(α, β) is makes the reconstruction more challenging than the previous distribution, a fact that is also supported by the observed MSE which is 100× larger than the previous experiment. This reconstruction error can potentially be reduced by adding a small set of queries of exploratory nature scattered over the universe of values.

4.3.2 Reconstruction from k-NN Queries

In this subsection, we first discuss the limitations of the reconstruction attack AttackUnordered from k-NN queries by Kornaropoulos *et al.* [104]. Next, we introduce our method, which is scalable and supports reconstructions beyond uniform query distributions. Finally, we present experiments about the performance of our attack on synthetic and real-world datasets.

The two new ingredients of our reconstruction algorithm are: (1) use of support size estimators to compute an estimate of the length of each Voronoi segment *without any assumption about the underlying query distribution*; and (2) formulation of an optimization problem that outputs a *minimal distortion* of the estimated lengths so as to transform them to a valid Voronoi diagram to overcome the fact that the estimated lengths of Voronoi segments are not always geometrically consistent.

Overview of AttackUnordered [104]. An insight from [104] is that even when the adversary observes all the possible queries, or else knowns the *exact* length of each Voronoi segment, it is

impossible to achieve exact reconstruction of the encrypted database (see Theorem 2 in [104]). This is because there are arbitrarily many value assignments that have the same Voronoi diagram which implies that the reconstruction error comes from the combination of (1) the length estimation errors and (2) the choice of a reconstruction among the many valid ones. First, AttackUnordered estimates the length of a Voronoi segment $\text{Len}(\{id_i, \ldots, id_{i+k-1}\})$ as the frequency of a response $\{id_i, \ldots, id_{i+k-1}\}$ multiplied by the size of the universe of queries. This simple estimator is accurate under the assumption that the queries are generated uniformly at random. As shown in [104], any set of values that implies the observed Voronoi diagram satisfies three families of constrains:

- ordering constraints, i.e., $v_i < v_{i+1}$,
- bisector constrains, i.e., $(v_i + v_j)/2 = b_{i,j}$, and
- boundary constraints, i.e., $\alpha < v_0$ and $v_{n-1} < \beta$.

All the above constrains form a feasible region $\mathcal{F}$ of potential reconstructions, which is geometrically expressed as a k-dimensional convex polytope.

Limitations of AttackUnordered [104]. We identify here some limitations of the approach in [104] and later show how to overcome them. The length estimation in AttackUnordered can be performed *solely with the access-pattern leakage*, hence even though the adversary observes a wealth of information from the search-pattern, this information is not utilized. Also, algorithm AttackUnordered provides rigorous guarantees about the quality of the reconstruction, but this precision comes with a price. The experiments of [104] show that for a successful reconstruction, it is preferable to have (1) a large number of queries and (2) a small number of neighgbors returned, k. Finally, the number of queries must be large enough so as the *estimated lengths are so accurate that they define a Voronoi diagram without any modification.* As an example, to achieve a reconstruction on the real-world Spitz dataset, the experiments in [104] observed more than 250 million queries. The proposed approach in this chapter achieves a reconstruction on the same dataset with 2.5 million queries, a 100× smaller sample size. Overall, the exact approach of AttackUnordered [104] either succeeds with great accuracy or fails and outputs nothing. Additionally the technique in [104] requires the *explicit computation of the feasible region* by computing the vertices of the feasible region $\mathcal{F}$ which requires time that is linear to the number of vertices of $\mathcal{F}$. We note that a k-dimensional convex polytope has $O(2^k)$ vertices, therefore such an approach does not scale well to larger k values. Our new approach overcomes both of the above limitations and utilizes the search-pattern leakage.

Algorithm 8: AGNOSTIC-RECONSTRUCTION-KNN

Input: A multiset of k-NN search tokens and their responses $D = \{(t_1, r_1), (t_2, r_2) \ldots, (t_m, r_m)\}$, the ordering of the records $I = (id_0, \ldots, id_{n-1})$, as well as α, β, N

Output: Approximate Reconstruction $\tilde{v}_0, \ldots, \tilde{v}_{n-1}$

Compute the left-to-right ordering S of the responses, i.e., the Voronoi segments, using the ordering I of the records.;

for *every* r_i *in* S *from left-to-right* **do**

  Define D_i as the mulitset with tokens from D with response r_i;

  Call MODULAR-ESTIMATOR with input the multiset of tokens D_i and store the estimated support size as $\widehat{L}_i$;

end

Define the vector of estimated lengths $\widehat{l} \leftarrow (\widehat{L}_0, \ldots, \widehat{L}_{n-k})$;

Solve the following convex optimization problem with unknowns the vector of distortions $\vec{\delta}$ and the vector of offsets $\vec{\xi}$:

$$\min_{\vec{\delta}, \vec{\xi}} \quad \sum_{i=0}^{n-k} \delta_i^2$$

$$\text{s.t.} \quad A \cdot \begin{bmatrix} \vec{\delta} \\ \vec{\xi} \end{bmatrix} \leq B \cdot \widehat{l}, \qquad \text{(ordering constraint)}$$

$$a_l^T \cdot \begin{bmatrix} \vec{\delta} \\ \vec{\xi} \end{bmatrix} \leq b_l \cdot \widehat{l}, \qquad \text{(lower boundary constraint)}$$

$$a_u^T \cdot \begin{bmatrix} \vec{\delta} \\ \vec{\xi} \end{bmatrix} \leq b_u \cdot \widehat{l}, \qquad \text{(upper boundary constraint)}$$

$$\vec{\xi} \geq 0, \qquad \text{(positive offsets constraint)}$$

$$\vec{\delta}^T \widehat{l} = N, \qquad \text{(sum of augmented lengths)}$$

where A and B are matrices of constant terms derived from the ordering constraints of Lemma 14, a_l, b_l are vectors of constant terms for the lower boundary constraint derived from Lemma 15, and a_u, b_u are vectors of constant terms for the upper boundary constraint derived from Lemma 15;

From the distortion vector $\vec{\delta}$ returned from the optimization problem and the estimated lengths $\widehat{l}$ we compute the augmented Voronoi diagram;

Given the above Voronoi diagram and the offset vector $\vec{\delta}$ returned from the optimization problem we derive the reconstructed database by substituting on the formulas of Lemma 5 in [104];

return $\tilde{v}_0, \ldots, \tilde{v}_{n-1}$;

Our Reconstruction Algorithm. Algorithm 10 (AGNOSTIC-RECONSTRUCTION-KNN) outlines our reconstruction attack from k-NN queries. A key insight of this algorithm is using the search-pattern leakage to estimate the length of each Voronoi segment under an *any arbitrary query distribution*. We build on the attack in [104] and extend it into two new directions. Instead of expecting a number of queries large enough to accurately estimate a valid Voronoi diagram, we compute the *minimum distortion* for each estimated length so as the *new "augmented" set of lengths comprise a valid Voronoi diagram*. We achieved this by adding *distortion variables* to the *offset variables* of [104] and introducing a convex optimization problem where the feasible region formulation from [104] forms the set of inequality constraints and the objective function expresses the minimization of the distortion. Finally, in order to scale to larger values of k, we don't require the explicit construction of the feasible region and instead output a reconstruction from the feasible region of the augmented

Voronoi diagram.

Observation 2 from Section 4.2 shows that an adversary with a sample D of search tokens and their responses can partition D with respect to each of the $n-k+1$ possible responses and form a collection of *samples from the conditional probability distributions*. From Remark 4 we know that the support size of a conditional probability distribution is the length of the corresponding Voronoi segment. Our algorithm deploys the MODULAR-ESTIMATOR to acquire an *estimation of the length* of each Voronoi segment without any assumptions about the query distribution, see Lines 2-6 in AGNOSTIC-RECONSTRUCTION-KNN. After this step the estimated lengths, i.e., $\widehat{l} = (\widehat{L}_0, \ldots, \widehat{L}_{n-k})$ are treated as constants. We define one *distortion variable* δ_i per estimated length $\widehat{L}_i$, for $i \in [0, n-k]$. We derive the value assignment of these variables by solving a quadratic minimization problem where the objective function is the sum of the squares of δ_i, i.e., $\min \sum_{i=0}^{n-k} \delta_i^2$. This design choice captures our goal to compute the smallest possible distortion. We follow the footsteps of [104] and express the space of valid reconstructions with respect to offsets ξ_i from bisectors. Overall the optimization formulation has $n-k+1$ unknowns for the distortion variables $\vec{\delta} = (\delta_0, \ldots, \delta_{n-k})$ and k unknowns for the offset variables $\vec{\xi} = (\xi_0, \ldots, \xi_{k-1})$, so a total of $n+1$ unknowns. The above objective function can be written as $\vec{x}^T M \vec{x}$, where $\vec{x}$ is the column vector from the concatenation of $\vec{\delta}$ and $\vec{\xi}$, and M is an all-zero matrix except the first $n-k+1$ elements of the main diagonal which have value 1. Since the matrix M is positive semidefinite, the objective function is a convex function.

Additionally the assignment of $\vec{\delta}$ and $\vec{\xi}$ should be such that the collection of augmented lengths, i.e., $\widehat{L}_i + \delta_i$ for $i \in [0, n-k]$, forms a Voronoi diagram. To express this goal we form four type of linear constraints for the optimization problem. The first type of constraints is the ordering constraints. These constraints can be written as $A \cdot [\vec{\delta}, \vec{\xi}]^T \leq B \cdot \widehat{l}$, where A is $(n-1) \times (n+1)$ matrix of constants and B is $(n-1) \times (n-k+1)$ matrix of constants. These matrices can be derived from the analytical formulas of Lemma 14. The second type of constraints is the boundary constraints which guarantee that $\alpha < v_0$ and $v_{n-1} < \beta$, see Lemma 15 for the analytical formula. The third type of constraints guarantees that the offsets are positive, i.e., $\xi \geq 0$. Finally the fourth type of constraints is an equality constraint that guarantees that the augmented lengths sum to N, i.e., $\sum_{i=0}^{n-k} (\widehat{L}_i + \delta_i) = N$.

Lemma 14. *The ordering constraint $v_i < v_{i+1}$ can be expressed as a function of A) the offsets $\xi = (\xi_0, \ldots, \xi_{k-1})$, B) the distortion of each Voronoi segment $\delta = (\delta_0, \ldots, \delta_{n-k})$, and C) the lengths of a subset of Voronoi segments $L_0, \ldots, L_{n-k}$. Specifically by using Lemma 8 from [104] we get the following cases:*

- *if $0 \leq i < k-1$, then $v_i < v_{i+1}$ can be written as:*

$$-\xi_i + \xi_{i+1} - \delta_{i+1} < c_{i,i+1}, \text{ where } c_{i,i+1} = L_{i+1}$$

- *if $i = k-1$, then $v_i < v_{i+1}$ can be written as:*

$$-\xi_{k-1} - \xi_0 + \sum_{1 \leq l \leq k-1} \delta_l < c_{k-1,k}, \text{ where } c_{k-1,k} = -\sum_{1 \leq l \leq k-1} L_l$$

- *if $k \leq i < 2k-1$, then $v_i < v_{i+1}$ can be written as:*

$$\xi_{i \bmod k} - \xi_{i \bmod k+1} - \delta_{i \bmod k+1} < c_{i,i+1}, \text{ where } c_{i,i+1} = L_{i \bmod k+1}$$

- *if $i = 2k-1$, then $v_i < v_{i+1}$ can be written as:*

$$\xi_{k-1} + \xi_0 - \delta_k - \sum_{1 \leq l \leq k} \delta_l < c_{2k-1,2k}, \text{ where } c_{2k-1,2k} = L_k + \sum_{1 \leq l \leq k} L_l$$

- *if $2k \leq i < n-1$ and $(i+1) \bmod k \neq 0$, then $v_i < v_{i+1}$ can be written as:*

$$(-1)^{\lfloor i/k-1 \rfloor}(\xi_{i \bmod k} - \xi_{(i+1) \bmod k}) - (-1)^{\lfloor i/k-1 \rfloor}(\delta_{(i+1) \bmod k}) - \sum_{2 \leq j \leq \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2(\delta_{i \bmod k+(j-1)k+1}) < c_{i,i+1}$$

$$, \text{ where } c_{i,i+1} = (-1)^{\lfloor i/k-1 \rfloor} L_{(i+1) \bmod k} + \sum_{2 \leq j \leq \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2L_{i \bmod k+(j-1)k+1}$$

- *if $2k \leq i < n-1$ and $(i+1) \bmod k = 0$, then $v_i < v_{i+1}$ can be written as:*

$$(-1)^{\lfloor i/k \rfloor+1}(\xi_{i \bmod k} + \xi_{(i+1) \bmod k}) - (-1)^{\lfloor i/k \rfloor+1}(\sum_{1 \leq l \leq k} \delta_l) - (-1)^{\lfloor i/k \rfloor+1}(\delta_k) - \sum_{2 \leq j \leq \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2(\delta_{jk}) < c_{i,i+1}$$

$$, \text{ where } c_{i,i+1} = (-1)^{\lfloor i/k \rfloor+1} \left(\sum_{1 \leq l \leq k} L_l \right) + (-1)^{\lfloor i/k \rfloor+1} L_k + \sum_{2 \leq j \leq \lfloor i/k \rfloor} (-1)^{j+\lfloor i/k \rfloor} 2L_{jk}$$

The first three cases the term $c_{i,i+1}$ consists of the length of a single Voronoi segment. For the fourth case the term $c_{i,i+1}$ is a linear combination of $2k-1$ length terms. For the fifth case the term $c_{i,i+1}$ is a linear combination of at most $\lfloor (n-1)/k \rfloor$ length terms. Finally for the last case $c_{i,i+1}$ is a linear combination of at most $\lfloor (n-1)/k \rfloor + k$ length terms.

Proof of Lemma 14: The proof is derived from Lemma 8 in [104] by substituting L_i with $(L_i + \delta_i)$ for $i \in [0, n-k]$.

Lemma 15. *The boundary constraints $\alpha < v_0$ and $v_{n-1} < \beta$ can be expressed as a function of A) the offsets $\xi = (\xi_0, \ldots, \xi_{k-1})$, B) the distortion of each Voronoi segment $\delta = (\delta_0, \ldots, \delta_{n-k})$, and C) the lengths of a subset of Voronoi segments $L_0, \ldots, L_{n-k}$. Specifically we have the following cases*

- *for the lower boundary*
$$\xi_0 - \delta_0 \leq c_l, \text{ where } c_l = L_0$$
- *for the upper boundary*
-if $k \leq n-1 < 2k$:
$$\xi_{(n-1) \bmod k} + \sum_{j=0}^{n-k-1} \delta_i < c_u, \text{ where } c_u = L_{n-k}$$
-if $2k \leq n-1$:
$$(-1)^{\lfloor (n-1)/k-1 \rfloor} \xi_{(n-1) \bmod k} + (-1)^{\lfloor (n-1)/k-1 \rfloor} \sum_{j=0}^{(n-1) \bmod k} \delta_j + \sum_{j=2}^{\lfloor (n-1)/k \rfloor} 2(-1)^{j+\lfloor (n-1)/k \rfloor} \cdot \left(\sum_{m=0}^{\substack{((n-1) \bmod k) \\ +(j-1)k}} \delta_m \right) < c_u$$
$$, \text{ where } c_u = \beta - (-1)^{\lfloor (n-1)/k-1 \rfloor} \alpha - (-1)^{\lfloor (n-1)/k-1 \rfloor} \sum_{j=0}^{(n-1) \bmod k} L_j - \sum_{j=2}^{\lfloor (n-1)/k \rfloor} 2(-1)^{j+\lfloor (n-1)/k \rfloor} \cdot \left(\sum_{m=0}^{\substack{((n-1) \bmod k) \\ +(j-1)k}} L_m \right)$$

Proof. Lower Boundary Constraint: Using Lemma 5 from [104]:

$$\alpha < v_0 \Rightarrow \alpha \leq b_{0,k} - \xi_0 \Rightarrow \xi_0 \leq b_{0,k} - \alpha$$
$$\Rightarrow \xi_0 \leq \mathsf{Len}(\{id_0, \ldots, id_{k-1}\}) \Rightarrow \xi_0 \leq L_0$$

If we replace every L_i with the term $L_i + \delta_i$ so as to consider the distortion variables $\delta_0, \ldots, \delta_{n-k}$ we get: $\xi_0 - \delta_0 \leq L_0$

Upper Boundary Constraint:

- if $k \leq n-1 < 2k$: Using Lemma 5 from [104] we get,

$$v_{n-1} = b_{(n-1) \bmod k, (n-1) \bmod k + k} + \xi_{(n-1) \bmod k} \Rightarrow$$
$$v_{n-1} = b_{(n-1) \bmod k, n-1} + \xi_{(n-1) \bmod k} \Rightarrow$$
$$v_{n-1} = \left(\alpha + \sum_{j=0}^{n-k-1} \mathsf{Len}(id_j, \ldots, id_{j+k-1}) \right) + \xi_{(n-1) \bmod k} \Rightarrow$$
$$v_{n-1} = \left(\alpha + \sum_{j=0}^{n-k-1} L_j \right) + \xi_{(n-1) \bmod k}$$

The upper boundary constraint is $v_{n-1} < \beta$. If we replace every L_i with the term $L_i + \delta_i$ so as to consider the distortion variables $\delta_0, \ldots, \delta_{n-k}$ we get:

$$
\begin{aligned}
& v_{n-1} < \beta \Rightarrow \\
& \left(\alpha + \sum_{j=0}^{n-k-1} (L_i + \delta_i)\right) + \xi_{(n-1) \bmod k} < \beta \Rightarrow \\
& \xi_{(n-1) \bmod k} + \sum_{j=0}^{n-k-1} \delta_i < \beta - \alpha - \sum_{j=0}^{n-k-1} L_i \Rightarrow \\
& \xi_{(n-1) \bmod k} + \sum_{j=0}^{n-k-1} \delta_i < \sum_{j=0}^{n-k} L_i - \sum_{j=0}^{n-k-1} L_i \Rightarrow \\
& \xi_{(n-1) \bmod k} + \sum_{j=0}^{n-k-1} \delta_i < L_{n-k}
\end{aligned}
$$

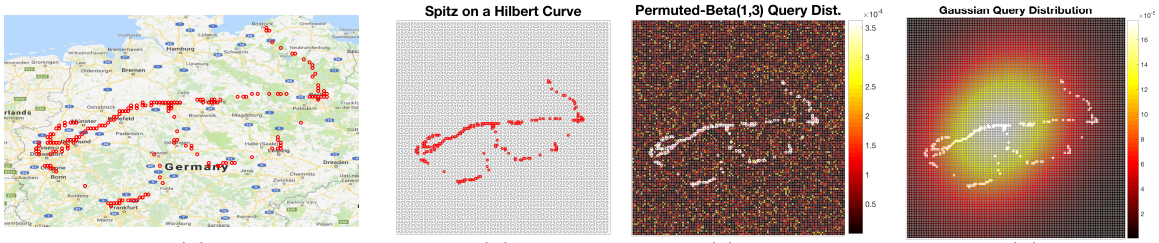
- if $2k \leq n-1$: Using Lemma 5 from [104] we get,

$$
\begin{aligned}
v_{n-1} = {} & (-1)^{\lfloor (n-1)/k-1 \rfloor} (b_{(n-1) \bmod k, ((n-1) \bmod k)+k} + \xi_{(n-1) \bmod k}) + \\
& + \sum_{j=2}^{\lfloor (n-1)/k \rfloor} (-1)^{j+\lfloor (n-1)/k \rfloor} 2 b_{((n-1) \bmod k)+(j-1)k, ((n-1) \bmod k)+jk} \Rightarrow \\
v_{n-1} = {} & (-1)^{\lfloor (n-1)/k-1 \rfloor} \xi_{(n-1) \bmod k} + \\
& + (-1)^{\lfloor (n-1)/k-1 \rfloor} b_{(n-1) \bmod k, ((n-1) \bmod k)+k} + \\
& + \sum_{j=2}^{\lfloor (n-1)/k \rfloor} (-1)^{j+\lfloor (n-1)/k \rfloor} 2 b_{((n-1) \bmod k)+(j-1)k, ((n-1) \bmod k)+jk} \Rightarrow \\
v_{n-1} = {} & (-1)^{\lfloor (n-1)/k-1 \rfloor} \xi_{(n-1) \bmod k} + \\
& + (-1)^{\lfloor (n-1)/k-1 \rfloor} (\alpha + \sum_{j=0}^{(n-1) \bmod k} \mathsf{Len}(id_j, \ldots, id_{j+k-1})) + \\
& + \sum_{j=2}^{\lfloor (n-1)/k \rfloor} 2(-1)^{j+\lfloor (n-1)/k \rfloor} \cdot \left(\sum_{m=0}^{\substack{((n-1) \bmod k) \\ +(j-1)k}} \mathsf{Len}(id_m, \ldots, id_{m+k-1}) \right) \Rightarrow \\
v_{n-1} = {} & (-1)^{\lfloor (n-1)/k-1 \rfloor} \xi_{(n-1) \bmod k} + \\
& + (-1)^{\lfloor (n-1)/k-1 \rfloor} (\alpha + \sum_{j=0}^{(n-1) \bmod k} L_j) + \\
& + \sum_{j=2}^{\lfloor (n-1)/k \rfloor} 2(-1)^{j+\lfloor (n-1)/k \rfloor} \cdot \left(\sum_{m=0}^{\substack{((n-1) \bmod k) \\ +(j-1)k}} L_m \right)
\end{aligned}
$$

The upper boundary constraint is $v_{n-1} < \beta$. If we replace every L_i with the term $L_i + \delta_i$ so as to consider the distortion variables $\delta_0, \ldots, \delta_{n-k}$ we get:

$$
\begin{aligned}
& v_{n-1} < \beta \Rightarrow \\
& (-1)^{\lfloor (n-1)/k-1 \rfloor} \xi_{(n-1) \bmod k} + \\
& + (-1)^{\lfloor (n-1)/k-1 \rfloor} (\alpha + \sum_{j=0}^{(n-1) \bmod k} (L_j + \delta_j) + \\
& + \sum_{j=2}^{\lfloor (n-1)/k \rfloor} 2(-1)^{j+\lfloor (n-1)/k \rfloor} \cdot \left(\sum_{m=0}^{\substack{((n-1) \bmod k) \\ +(j-1)k}} (L_m + \delta_m) \right) < \beta \Rightarrow \\
& (-1)^{\lfloor (n-1)/k-1 \rfloor} \xi_{(n-1) \bmod k} + (-1)^{\lfloor (n-1)/k-1 \rfloor} \sum_{j=0}^{(n-1) \bmod k} \delta_j + \\
& + \sum_{j=2}^{\lfloor (n-1)/k \rfloor} 2(-1)^{j+\lfloor (n-1)/k \rfloor} \cdot \left(\sum_{m=0}^{\substack{((n-1) \bmod k) \\ +(j-1)k}} \delta_m \right) \\
& < \beta - (-1)^{\lfloor (n-1)/k-1 \rfloor} \alpha - (-1)^{\lfloor (n-1)/k-1 \rfloor} \sum_{j=0}^{(n-1) \bmod k} L_j \\
& - \sum_{j=2}^{\lfloor (n-1)/k \rfloor} 2(-1)^{j+\lfloor (n-1)/k \rfloor} \cdot \left(\sum_{m=0}^{\substack{((n-1) \bmod k) \\ +(j-1)k}} L_m \right)
\end{aligned}
$$

□



(a) (b) (c) (d)

Figure 4.10: Real-world dataset Spitz of a privacy-sensitive geolocation trace: (a) data for Otober 1-31, 2009; (b) mapping of the points to a Hilbert curve which reduces the 2D data to 1D; (c) query distribution under attack, which consists of a permutation of a discretized Beta(α, β) distribution; (d) another query distribution under attack, which is a Gaussian centered at the city of Hannover, Germany.

Evaluation on the Spitz Dataset. In this experiment, we evaluate the performance of Agnostic-Reconstruction-KNN on a public real-world data set (also used in [104]) containing the geolocation of politician Malte Spitz.[1] As in [104], we consider the geolocation data for the period October 1 to 31, 2009 and reduce the 2D data to 1D by deploying a Hilbert curve of order 8. The resulting discretized curve has universe of size $N = 65536$ and the dataset has size $n = 258$.

[1] www.zeit.de/datenschutz/malte-spitz-data-retention

The data is shown on a super-imposed map in Figure 4.10(a) and its mapping on a Hilbert curve is in Figure 4.10(b). The deployed query distribution is a discretized beta for varying parameters, similar to the experiments of the previous subsection but without any noise, i.e., *permuted over the universe of queries*. We illustrate this Permuted-Beta(α, β) query distribution with a heatmap on the superimposed data in Figure 4.10(c). Finally we test a Gaussian query distribution with mean centered at the city of Hannover in Germany and it is illustrated in Figure 4.10 (d).

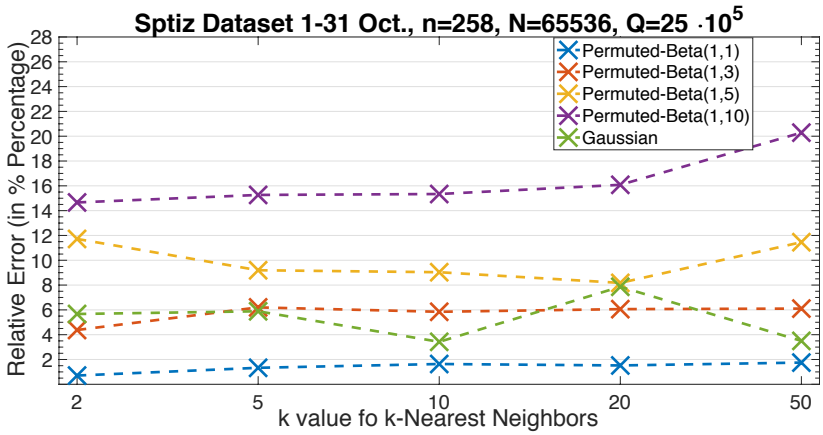


Figure 4.11: Absolute relative error of Agnostic-Reconstruction-KNN for varying query distributions on the Spitz dataset.

The number of queries that the adversary observed is set to $Q = 25 \cdot 10^5$ which is 100× smaller sample size than the experiments conducted in [104]. Each attack was mounted 50 times and Figure 4.11 presents the average absolute relative error. Due to the new design of our reconstruction attack we were able to scale it to $k = 50$ an experiment that is not feasible from the approach followed in [104]. As it is expected the power-law like distribution Permuted-Beta(1,10) is the hardest to reconstruct due to the skewness and the sample size. Nevertheless the relative error ranges from 15% to 20% in this challenging scenario. The reconstruction under the Gaussian query distribution is accurate across all values of k. An illustration this reconstruction is depicted in Figure 1.3.

Evaluation on Synthetic Dataset. We generated synthetic databases under varying densities and query distributions for $N = 10^3$ and $k = \{2, 5, 10, 20, 50\}$. Figure 4.12 shows the average of the mean absolute error of 100 repetitions with $Q = 10^5$. Note that for sparse databases, the distances between the values are larger, hence the offset variables have "more room" to deviate, which increases the size of the feasible region and as a result, the number of possible valid reconstructions. Another factor that increases the size of the feasible region is the increase of the value k, an intuition confirmed by the MAE for $k = 50$ even for the uniform case Permuted-Beta(1,1) which is easier to reconstruct.

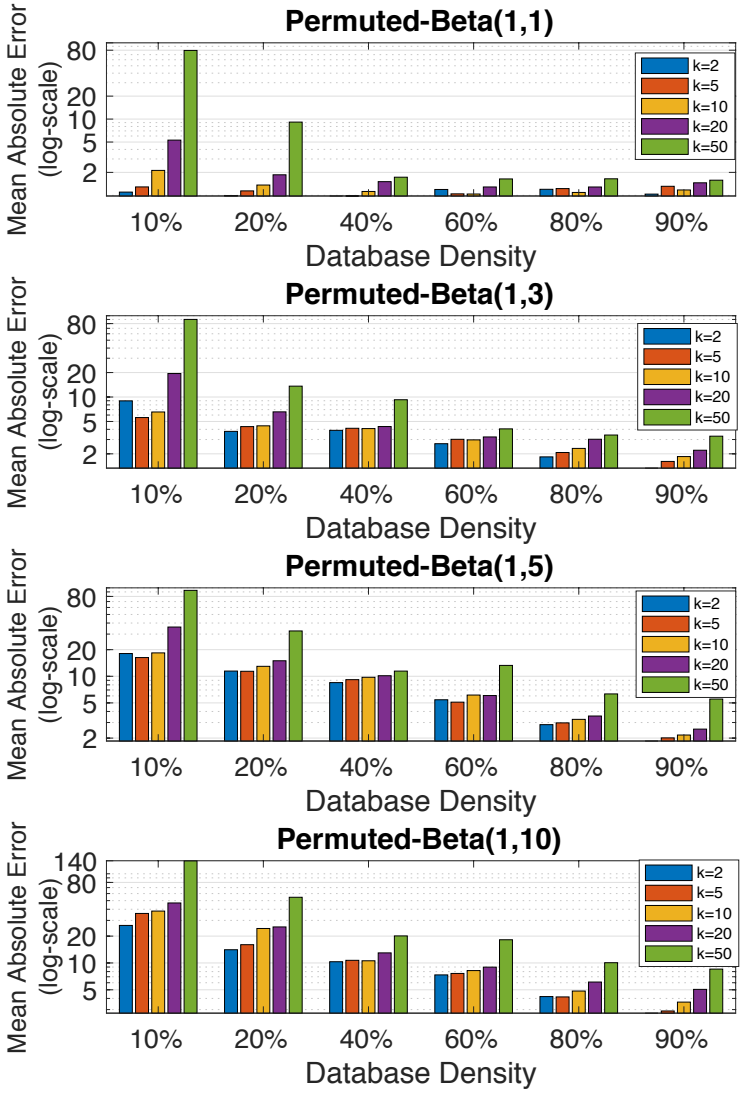


Figure 4.12: Performance of Agnostic-Reconstruction-KNN for varying query distributions on synthetic data.

For densities larger than 20%, the reconstruction is usually within a distance of 20 from the plaintext value for all the tested query distributions.

4.4 Related Work

For encrypted single-keyword search [23, 36, 38, 45, 67, 96, 147, 149] the access pattern leakage of some leakage profiles is vulnerable to *query recovery attacks*, as opposed to the encrypted values. Specifically, Islam *et al.* [93], Cash *et al.* [37], and Zhang *et al.* [163] give *query-recovery* attacks under various assumptions. Encrypted systems with more expressive queries [139] rely on different cryptographic

primitives, e.g., order-preserving encryption, and are vulnerable to data-recovery attacks [59, 82, 128] using only the setup leakage. In terms of efficiency there is a series of works [10, 11, 35, 52, 54] that study how the locality of searchable encryption affects the overall efficiency. Recent work improves the asymptotic complexity of reconstruction from range queries under uniform query distribution or observation of all possible queries [117].

Bibliography

[1] IBM's Cloudant NoSQL DB Geospatial. URL https://console.bluemix.net/docs/services/Cloudant/api/cloudant-geo.html.

[2] Analyze N-dimensional Polyhedra in Terms of Vertices or (In)Equalities, version 1.9.0.0, by Matt J. URL https://www.mathworks.com/matlabcentral/fileexchange/30892-analyze-n-dimensional-polyhedra-in-terms-of-vertices-or-in-equalities.

[3] PostGIS for PostgreSQL. URL https://postgis.net/docs/geometry_distance_knn.html.

[4] Dataset SpitzLoc. URL http://www.zeit.de/datenschutz/malte-spitz-data-retention.

[5] EDRM: Creating Practical Resources to Improve E-Discovery and Information Governance. www.edrm.net/, . Accessed: 2017-11-27.

[6] EDRM: Processing Guide. www.edrm.net/frameworks-and-standards/edrm-model/processing/, . Accessed: 2017-11-27.

[7] Geomesa. URL http://www.geomesa.org/documentation/user/process.html#knearestneighborprocess.

[8] Principles and Strategy for Accelerating Health Information Exchange (HIE). Department of Health & Human Services, USA, 2013.

[9] MATLAB and Optimization Toolbox Release 2016a, The Mathworks, Inc., Natick, Massachusetts, United States., 2016.

[10] G. Asharov, M. Naor, G. Segev, and I. Shahaf. Searchable Symmetric Encryption: Optimal Locality in Linear Space via Two-dimensional Balanced Allocations. In *Proc. of the 48th ACM STOC*, pages 1101–1114, 2016.

[11] G. Asharov, G. Segev, and I. Shahaf. Tight Tradeoffs in Searchable Symmetric Encryption. In *Proc. of the 38th CRYPTO*, pages 407–436, 2018.

[12] A. Athalye, N. Carlini, and D. A. Wagner. Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples. In *in Proc. of the 35th ICML*, pages 274–283, 2018.

[13] D. Avis and K. Fukuda. A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra. *Discrete Comput. Geom.*, 8(3):295–313, 1992.

[14] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik. Countering GATTACA: Efficient and Secure Testing of Fully-sequenced Human Genomes. In *Proc. of the 18th ACM CCS*, pages 691–702, 2011.

[15] F. Baldimtsi and O. Ohrimenko. Sorting and Searching Behind the Curtain. In *Proc. of the 19th FC*, pages 127–146, 2015.

[16] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar. The security of machine learning. *Machine Learning*, 81(2):121–148, 2010.

[17] T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that Distributions are Close. In *Proc. of the 41st IEEE FOCS*, pages 259–269, 2000.

[18] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway. Efficient Garbling from a Fixed-Key Blockcipher. In *IEEE S&P*, pages 478–492, 2013.

[19] J. Bethencourt. Advanced Crypto Software Collection, 2010. URL `http://acsc.cs.utexas.edu/libpaillier/`.

[20] M. Blanton and F. Bayatbabolghani. Efficient Server-Aided Secure Two-Party Function Evaluation with Applications to Genomic Computation. *PoPETs*, 2016(4):144–164, 2016.

[21] M. Blanton and P. Gasti. Secure and Efficient Protocols for Iris and Fingerprint Identification. In *ESORICS*, pages 190–209, 2011.

[22] C. Blundo, E. De Cristofaro, and P. Gasti. EsPRESSO: Efficient Privacy-preserving Evaluation of Sample Set Similarity. *J. Comput. Secur.*, 22(3):355–381, 2014.

[23] R. Bost. $\sum o\varphi o\varsigma$: Forward secure searchable encryption. In *Proc. of the 23rd ACM CCS*, pages 1143–1154, 2016.

[24] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser. Machine Learning Classification over Encrypted Data. In *Proc. of the 22nd NDSS*, 2015.

[25] A. Z. Broder. On the Resemblance and Containment of Documents. In *In Compression and Complexity of Sequences (SEQUENCES)*, pages 21–29, 1997.

[26] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-Wise Independent Permutations. *J. Comput. Syst. Sci.*, 60(3):630–659, 2000.

[27] S.-A. Brown. Patient Similarity: Emerging Concepts in Systems and Precision Medicine. *Frontiers in Physiology*, 7, 11 2016.

[28] J. Bunge and M. Fitzpatrick. Estimating the Number of Species: A Review. *Journal of the American Statistical Association*, 88(421):364–373, 1993.

[29] K. P. Burnham and W. S. Overton. Estimation of the Size of a Closed Population when Capture Probabilities vary Among Animals. *Biometrika*, 65(3):625–633, 1978.

[30] K. P. Burnham and W. S. Overton. Robust Estimation of Population Size When Capture Probabilities Vary Among Animals. *Ecology*, 60(5):927–936, 1979.

[31] R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. *J. Cryptology*, 13(1):143–202, 2000.

[32] R. Canetti. Security and Composition of Cryptographic Protocols: A Tutorial (Part i). *SIGACT News*, 37(3):67–92, Sept. 2006.

[33] N. Carlini and D. A. Wagner. Towards Evaluating the Robustness of Neural Networks. In *Proc. of the 38th IEEE S&P*, pages 39–57, 2017.

[34] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou. Hidden Voice Commands. In *Proc. of the 25th USENIX Security*, pages 513–530, 2016.

[35] D. Cash and S. Tessaro. The Locality of Searchable Symmetric Encryption. In *Proc. of the 33rd EUROCRYPT*, pages 351–368, 2014.

[36] D. Cash, J. Jaeger, S. Jarecki, C. S. Jutla, H. Krawczyk, M. Rosu, and M. Steiner. Dynamic searchable encryption in very-large databases: Data structures and implementation. In *Proc. of the 21st NDSS*, 2014.

[37] D. Cash, P. Grubbs, J. Perry, and T. Ristenpart. Leakage-Abuse Attacks Against Searchable Encryption. In *Proc. of the 22nd ACM CCS*, pages 668–679, 2015.

[38] J. G. Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili. New Constructions for Forward and Backward Private Symmetric Searchable Encryption. In *Proc. of the 25th ACM CCS*, pages 1038–1055, 2018.

[39] A. Chao and C.-H. Chiu. *Species Richness: Estimation and Comparison*, pages 1–26. American Cancer Society, 2016.

[40] M. S. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *Proc. of STOC*, pages 380–388, 2002.

[41] M. Chase and S. Kamara. Structured encryption and controlled disclosure. In *Proc. of the 16th ASIACRYPT*, 2010.

[42] H. Chen, I. Chillotti, Y. Dong, O. Poburinnaya, I. Razenshteyn, and M. S. Riazi. SANNS: Scaling Up Secure Approximate k-Nearest Neighbors Search. Cryptology ePrint Archive, Report 2019/359, 2019. `https://eprint.iacr.org/2019/359`.

[43] Y. Chen, B. Peng, X. Wang, and H. Tang. Large-Scale Privacy-Preserving Mapping of Human Genomic Sequences on Hybrid Clouds. In *Proc. of the 19th NDSS*, 2012.

[44] G. Cormode and S. Muthukrishnan. An Improved Data Stream Summary: The Count-min Sketch and Its Applications. *J. Algorithms*, 55(1):58–75, Apr. 2005.

[45] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In *Proc. of the 13th ACM CCS*, pages 79–88, 2006.

[46] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial Classification. In *Proc. of the 10th ACM KDD*, pages 99–108, 2004.

[47] I. Damgård, M. Geisler, and M. Krøigaard. Efficient and Secure Comparison for On-line Auctions. In *ACISP*, pages 416–430, 2007.

[48] I. Damgård, M. Geisler, and M. Krøigaard. A correction to "Efficient and Secure Comparison for On-line Auctions". *IJACT*, 1(4):323–324, 2009.

[49] G. Danezis and E. De Cristofaro. Fast and Private Genomic Testing for Disease Susceptibility. In *Proc. of the 13th WPES*, pages 31–34, 2014.

[50] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proc. of the 16th WWW*, pages 271–280, 2007.

[51] Deloitte. GCC eDiscovery survey, How ready are you? `https://tinyurl.com/jp2qwey`, 2015.

[52] I. Demertzis and C. Papamanthou. Fast Searchable Encryption With Tunable Locality. In *Proc. of ACM SIGMOD*, pages 1053–1067, 2017.

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

[54] I. Demertzis, D. Papadopoulos, and C. Papamanthou. Searchable encryption with optimal locality: Achieving sublogarithmic read efficiency. In *Proc. of the 38th CRYPTO*, pages 371–406, 2018.

[55] G. development team. GMP: The GNU Multiple Precision Arithmetic Library, 2016. URL `https://gmplib.org/`.

[56] L. Devroye. The Equivalence of Weak, Strong and Complete Convergence in L_1 for Kernel Density Estimates. *The Annals of Statistics*, 11(3):896–904, 1983.

[57] I. Diakonikolas, T. Gouleakis, J. Peebles, and E. Price. Collision-based Testers are Optimal for Uniformity and Closeness. *Electronic Colloquium on Computational Complexity (ECCC)*, 23: 178, 2016.

[58] W. Dong, M. Charikar, and K. Li. Asymmetric Distance Estimation with Sketches for Similarity Search in High-dimensional Spaces. In *Proc. of SIGIR*, pages 123–130, 2008.

[59] F. B. Durak, T. M. DuBuisson, and D. Cash. What Else is Revealed by Order-Revealing Encryption? In *Proc. of the 23rd ACM CCS*, pages 1155–1166, 2016.

[60] B. Efron and R. Thisted. Estimating the Number of Unseen Species: How Many Words Did Shakespeare Know? *Biometrika*, 63(3):435–447, 1976.

[61] Y. Elmehdwi, B. K. Samanthula, and W. Jiang. Secure k-nearest neighbor query over encrypted data in outsourced environments. In *Proc. of the 30th IEEE ICDE*, pages 664–675, 2014.

[62] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft. Privacy-Preserving Face Recognition. In *Proc. of 9th PETS*, pages 235–253, 2009.

[63] S. Faber, S. Jarecki, H. Krawczyk, Q. Nguyen, M. Rosu, and M. Steiner. Rich Queries on Encrypted Data: Beyond Exact Matches. In *Proc. of the 20th ESORICS*, pages 123–145, 2015.

[64] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. J. Strauss, and R. N. Wright. Secure multiparty computation of approximations. *ACM Trans. on Algorithms*, 2(3):435–472, 2006.

[65] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page, and T. Ristenpart. Privacy in Pharmacogenetics: An End-to-End Case Study of Personalized Warfarin Dosing. In *Proc. of the 23rd USENIX Security*, pages 17–32, 2014.

[66] M. Fredrikson, S. Jha, and T. Ristenpart. Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures. In *Proc. of the 22nd ACM CCS*, pages 1322–1333, 2015.

[67] B. Fuller, M. Varia, A. Yerukhimovich, E. Shen, A. Hamlin, V. Gadepally, R. Shay, J. D. Mitchell, and R. K. Cunningham. SoK: Cryptographically Protected Database Search. In *Proc. of the 38th IEEE S&P*, pages 172–191, 2017.

[68] A. Gandolfi and C. C. A. Sastri. Nonparametric estimations about species not observed in a random sample. *Milan Journal of Mathematics*, 72(1):81–105, Oct 2004.

[69] R. Gennaro, C. Gentry, and B. Parno. Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In *Proc. of CRYPTO*, pages 465–482, 2010.

[70] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan. Private queries in location based services: anonymizers are not necessary. In *Proc. of the ACM SIGMOD*, pages 121–132, 2008.

[71] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. pages 518–529, 1999.

[72] O. Goldreich. *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

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

[74] O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

[75] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 43(3):431–473, May 1996.

[76] O. Goldreich and D. Ron. On Testing Expansion in Bounded-Degree Graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(20), 2000.

[77] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984.

[78] Google. Protocol Buffers, 2017. URL `http://code.google.com/apis/protocolbuffers/`.

[79] A. Gottlieb, G. Stein, E. Ruppin, R. Altman, and R. Sharan. A method for inferring medical diagnoses from patient similarities. *BMC medicine*, 11:194, 09 2013.

[80] C. W. J. Granger and P. Newbold. *Forecasting economic time series*. Academic Press, 2014.

[81] M. R. Grossman and G. V. Cormack. Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review. *Rich. JL & Tech.*, 17:1, 2010.

[82] P. Grubbs, K. Sekniqi, V. Bindschaedler, M. Naveed, and T. Ristenpart. Leakage-Abuse Attacks Against Order-Revealing Encryption. In *Proc. of the 38th IEEE S&P*, pages 655–672, 2017.

[83] P. Grubbs, M. Lacharité, B. Minaud, and K. G. Paterson. Learning to Reconstruct: Statistical Learning Theory and Encrypted Database Attacks. In *Proc. of the 40th IEEE S&P*, pages 496–512, 2019.

[84] P. Gupta, A. Goel, J. Lin, A. Sharma, D. Wang, and R. Zadeh. WTF: The who to follow service at Twitter. In *Proc. of the 22nd WWW*, pages 505–514, 2013.

[85] F. Hahn and F. Kerschbaum. Poly-Logarithmic Range Queries on Encrypted Data with Small Leakage. In *Proc. of ACM CCSW*, pages 23–34, 2016.

[86] B. Harris. Statistical Inference in the Classical Occupancy Problem Unbiased Estimation of the Number of Classes. *Journal of the American Statistical Association*, 63(323):837–847, 1968.

[87] J. Hastad and A. Shamir. The Cryptographic Security of Truncated Linearly Related Variables. In *Proc. of STOC*, pages 356–362, 1985.

[88] T. H. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Evaluating Strategies for Similarity Search on the Web. In *Proc. of the 11th WWW*, pages 432–442, 2002.

[89] M. Henzinger. Finding Near-duplicate Web Pages: A Large-scale Evaluation of Algorithms. In *Proc. of the 29th ACM SIGIR*, pages 284–291, 2006.

[90] B. Hore, S. Mehrotra, and G. Tsudik. A Privacy-Preserving Index for Range Queries. In *Proc. of 30th VLDB*, pages 720–731, 2004.

[91] P. Indyk and R. Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proc. of the 13th ACM STOC*, pages 604–613, 1998.

[92] Y. Ishai, J. Katz, E. Kushilevitz, Y. Lindell, and E. Petrank. On Achieving the “Best of Both Worlds" in Secure Multiparty Computation. *SIAM J. Comput.*, 40(1):122–141, 2011.

[93] M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access Pattern Disclosure on Searchable Encryption: Ramification, Attack and Mitigation. In *Proc. of the 19th NDSS*, 2012.

[94] P. Jaccard. Etude de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37, 1901.

[95] S. Kamara and T. Moataz. Boolean Searchable Symmetric Encryption with Worst-Case Sub-linear Complexity. In *Proc. of the 36th EUROCRYPT*, pages 94–124, 2017.

[96] S. Kamara, C. Papamanthou, and T. Roeder. Dynamic Searchable Symmetric Encryption. In *Proc. of the 19th ACM CCS*, pages 965–976, 2012.

[97] S. Kamara, P. Mohassel, M. Raykova, and S. Sadeghian. Scaling Private Set Intersection to Billion-Element Sets. In *Proc. of FC*, pages 195–215, 2014.

[98] S. Keelveedhi, M. Bellare, and T. Ristenpart. DupLESS: Server-Aided Encryption for Deduplicated Storage. In *Proc. of the 22nd USENIX Security*, pages 179–194, 2013.

[99] G. Kellaris, G. Kollios, K. Nissim, and A. O'Neill. Generic Attacks on Secure Outsourced Databases. In *Proc. of the 23rd ACM CCS*, pages 1329–1340, 2016.

[100] A. Khedr, P. G. Gulak, and V. Vaikuntanathan. SHIELD: Scalable Homomorphic Implementation of Encrypted Data-Classifiers. *IEEE Transactions on Computers*, PP:99, 2015.

[101] A. Khoshgozaran, H. Shirani-Mehr, and C. Shahabi. Blind Evaluation of Location Based Queries Using Space Transformation to Preserve Location Privacy. *GeoInformatica*, 17(4): 599–634, 2013.

[102] M. S. Kohn, J. Sun, S. Knoop, A. Shabo, B. Carmeli, D. Sow, T. Syed-Mahmood, and W. Rapp. IBM's Health Analytics and Clinical Decision Support. In *Yearbook of Medical Informatics 9.1*, pages 154–162, 2014.

[103] E. M. Kornaropoulos and P. Efstathopoulos. The Case of Adversarial Inputs for Secure Similarity Approximation Protocols. In *Proc. of the 4th IEEE Euro&SP*, 2019.

[104] E. M. Kornaropoulos, C. Papamanthou, and R. Tamassia. Data Recovery on Encrypted Databases With k-Nearest Neighbor Query Leakage. In *Proc. of the 40th IEEE S&P*, 2019.

[105] M. Lacharité and K. Paterson. Frequency-Smoothing Encryption: Preventing Snapshot Attacks on Deterministically Encrypted Data. *IACR Transactions on Symmetric Cryptology*, 2018(1): 277–313, Mar. 2018.

[106] M. S. Lacharité, B. Minaud, and K. G. Paterson. Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage. In *Proc. of the 39th IEEE S&P*, pages 1–18, 2018.

[107] R. L. Lagendijk, Z. Erkin, and M. Barni. Encrypted Signal Processing for Privacy Protection: Conveying the Utility of Homomorphic Encryption and Multiparty Computation. *IEEE Signal Process. Mag.*, 30(1):82–105, 2013.

[108] J. K. Lawder and P. J. H. King. Using Space-Filling Curves for Multi-Dimensional Indexing. In *Proc. of the 17th BNCOD*, pages 20–35, 2000.

[109] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of Massive Datasets*. Cambridge university press, 2014.

[110] K. Lewi and D. J. Wu. Order-Revealing Encryption: New Constructions, Applications, and Lower Bounds. In *Proc. of the 23rd ACM CCS*, pages 1167–1178, 2016.

[111] R. C. Lewontin and T. Prout. Estimation of the Number of Different Classes in a Population. *Biometrics*, 12(2):211–223, 1956.

[112] F. Li, R. Shin, and V. Paxson. Exploring Privacy Preservation in Outsourced k-Nearest Neighbors with Multiple Data Owners. In *Proc. of ACM CCSW*, pages 53–64, 2015.

[113] P. Li and A. C. König. b-Bit minwise hashing. In *Proc. of WWW*, pages 671–680, 2010.

[114] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the Association for Information Science and Technology*, 58(7):1019–1031, 2007.

[115] G. S. Manku, A. Jain, and A. Das Sarma. Detecting Near-duplicates for Web Crawling. In *WWW*, pages 141–150, 2007.

[116] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

[117] E. A. Markatou and R. Tamassia. Full Database Reconstruction with Access and Search Pattern Leakage. Cryptology ePrint Archive, Report 2019/395, 2019. `https://eprint.iacr.org/2019/395`.

[118] P. McDaniel, N. Papernot, and Z. B. Celik. Machine Learning in Adversarial Settings. *IEEE Security & Privacy*, 14(3):68–72, 2016.

[119] L. Melis, G. Danezis, and E. D. Cristofaro. Efficient Private Statistics with Succinct Sketches. In *Proc. of NDSS*, 2016.

[120] I. Mironov, M. Naor, and G. Segev. Sketching in Adversarial Environments. *SIAM J. Comput.*, 40(6):1845–1870, 2011.

[121] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[122] M. Mitzenmacher, R. Pagh, and N. Pham. Efficient Estimation for High Similarities Using Odd Sketches. In *Proc. of WWW*, pages 109–118, 2014.

[123] P. Mohassel and Y. Zhang. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *Proc. of the 38th IEEE S&P*, pages 19–38, 2017.

[124] M. F. Mokbel, C. Chow, and W. G. Aref. The New Casper: A Privacy-Aware Location-Based Database Server. In *Proc. of the 23rd IEEE ICDE*, pages 1499–1500, 2007.

[125] B. Moon, H. v. Jagadish, C. Faloutsos, and J. H. Saltz. Analysis of the Clustering Properties of the Hilbert Space-Filling Curve. *IEEE Trans. on Knowl. and Data Eng.*, 13(1):124–141, 2001.

[126] M. Mudelsee. *Climate time series analysis*. Springer, 2013.

[127] M. Naor and E. Yogev. Bloom Filters in Adversarial Environments. In *Proc. of CRYPTO*, pages 565–584, 2015.

[128] M. Naveed, S. Kamara, and C. V. Wright. Inference Attacks on Property-Preserving.

[129] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang. Privacy in the Genomic Era. *ACM Comput. Surv.*, 48(1):6:1–6:44, Aug. 2015.

[130] S. Nishimura and H. Yokota. QUILTS: Multidimensional Data Partitioning Framework Based on Query-Aware and Skew-Tolerant Space-Filling Curves. In *Proc. of ACM SIGMOD*, pages 1525–1537, 2017.

[131] D. Notterman, U. Alon, A. Sierk, and A. Levine. Transcriptional Gene Expression Profiles of Colorectal Adenoma, Adenocarcinoma, and Normal Tissue Examined by Oligonucleotide Arrays. *Cancer Research*, 61:3124–3130, 2001.

[132] P. Paillier. Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In *EUROCRYPT*, pages 223–238, 1999.

[133] S. Papadopoulos, S. Bakiras, and D. Papadias. Nearest neighbor search with strong location privacy. *Proc. of VLDB*, 3(1):619–629, 2010.

[134] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The Limitations of Deep Learning in Adversarial Settings. In *IEEE EuroS&P*, 2016.

[135] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, and S. Bellovin. Blind Seer: A Scalable Private DBMS. In *Proc. of the 35th IEEE S&P*, pages 359–374, 2014.

[136] J.-H. Park, M. Kim, B.-N. Noh, and J. B. Joshi. A similarity based technique for detecting malicious executable files for computer forensics. In *Proc. of IEEE International Conference on Information Reuse and Integration*, pages 188–193, 2006.

[137] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly Practical Verifiable Computation. In *Proc. of the 34th IEEE S&P*, pages 238–252, 2013.

[138] R. Poddar, T. Boelter, and R. A. Popa. Arx: A Strongly Encrypted Database System. Cryptology ePrint Archive, Report 2016/591, 2016. `https://eprint.iacr.org/2016/591`.

[139] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. CryptDB: Protecting Confidentiality with Encrypted Query Processing. In *Proc. of the 23rd ACM SOSP*, pages 85–100, 2011.

[140] M. H. Quenouille. Approximate Tests of Correlation in Time-Series. *Journal of the Royal Statistical Society. Series B (Methodological)*, 11(1):68–84, 1949.

[141] A. Rajaraman and J. D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2011.

[142] F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[143] A. Sharafoddini, A. J. Dubin, and J. Lee. Patient Similarity in Prediction Models Based on Health Data: A Scoping Review. *JMIR Med Inform*, 5(1), Mar 2017.

[144] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition. In *Proc. of the 23rd ACM CCS*, pages 1528–1540, 2016.

[145] S. Shekhar, H. Xiong, and X. Zhou, editors. *Encyclopedia of GIS*. Springer, 2017.

[146] A. Shrivastava and P. Li. In Defense of Minhash over Simhash. In *Proc. of AISTATS*, pages 886–894, 2014.

[147] D. X. Song, D. A. Wagner, and A. Perrig. Practical Techniques for Searches on Encrypted Data. In *Proc. of the 21st IEEE S&P*, pages 44–55, 2000.

[148] A. Stam. Statistical Problems in Ancient Numismatics. *Statistica Neerlandica*, 41(3):151–174.

[149] E. Stefanov, C. Papamanthou, and E. Shi. Practical dynamic searchable encryption with small leakage. In *Proc. of the 21st NDSS*, 2014.

[150] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing Properties of Neural Networks. *CoRR*, abs/1312.6199, 2013.

[151] T. Urvoy, E. Chauveau, P. Filoche, and T. Lavergne. Tracking Web Spam with HTML Style Similarities. *ACM Trans. Web*, 2(1):3:1–3:28, 2008.

[152] G. Valiant and P. Valiant. Estimating the Unseen: Improved Estimators for Entropy and Other Properties. *J. ACM*, 64(6):37:1–37:41, Oct. 2017.

[153] T. Veugen. Encrypted Integer Division and Secure Comparison. *Int. Journal of Applied Cryptology*, 3(2):166–180, 2014.

[154] B. Wang, Y. Hou, and M. Li. Practical and secure nearest neighbor search on encrypted large-scale data. In *Proc. of the 35th IEEE INFOCOM*, pages 1–9, 2016.

[155] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu. Efficient Genome-Wide, Privacy-Preserving Similar Patient Query Based on Private Edit Distance. In *Proc. of the 22nd ACM CCS*, pages 492–503, 2015.

[156] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis. Secure kNN computation on encrypted databases. In *Proc. of the ACM SIGMOD*, pages 139–152, 2009.

[157] D. J. Wu, T. Feng, M. Naehrig, and K. E. Lauter. Privately Evaluating Decision Trees and Random Forests. *PETS*, 2016.

[158] R. Xiang, J. Neville, and M. Rogati. Modeling relationship strength in online social networks. In *Proc. of the 19th WWW*, pages 981–990, 2010.

[159] X. Yan, P. S. Yu, and J. Han. Substructure similarity search in graph databases. In *Proc. of SIGMOD*, pages 766–777, 2005.

[160] B. Yao, F. Li, and X. Xiao. Secure nearest neighbor revisited. In *Proc. of the 29th IEEE ICDE*, pages 733–744, 2013.

[161] M. L. Yiu, Y. Tao, and N. Mamoulis. The B^{dual}-Tree: Indexing Moving Objects by Space Filling Curves in the Dual Space. *VLDB J.*, 17(3):379–400, 2008.

[162] P. Zhang, F. Wang, J. Hu, and R. Sorrentino. Towards Personalized Medicine: Leveraging Patient Similarity and Drug Similarity Analytics. 2014:132–6, 04 2014.

[163] Y. Zhang, J. Katz, and C. Papamanthou. All Your Queries Are Belong to Us: The Power of File-Injection Attacks on Searchable Encryption. In *Proc. of the 25th USENIX Security*, pages 707–720, 2016.