Technical University of Crete, Greece
School of Electronic and Computer Engineering

Practical Secure and Efficient Range Search

Demertzis Ioannis

Master Thesis Committee
Professor Minos Garofalakis (ECE)
Associate Professor Antonios Deligiannakis(ECE)
Professor Stavros Christodoulakis (ECE)

Chania, July 2015

Πολυτεχνειο Κρητης

Σχολη Ηλεκτρονικων Μηχανικων και Μηχανικων Υπολογιστων

Πρακτικές Ασφαλείς και Αποδοτικές Ερωτήσεις Εύρους

Ιωάννης Δεμερτζής

Εξεταστική Επιτροπή Μεταπτυχιακής Εργασίας

Καθ. Μίνως Γαροφαλάκης (ΗΜΜΥ)

Αναπλ. Καθ. Αντώνιος Δεληγιαννάκης (ΗΜΜΥ)

Καθ. Σταύρος Χριστοδουλάκης (ΗΜΜΥ)

Χανιά, Ιούλιος 2015

Abstract

Due to their potential for near-infinite scalability, cloud computing platforms are rapidly becoming the defacto standard for large-scale, big data analytics. Still, serious concerns regarding the outsourcing and querying of private company and personal data remain a key roadblock in the adoption of such cloud platforms for numerous big-data applications. In this work, we extend cryptographic Searchable Symmetric Encryption (SSE) schemes to create the first adaptive Range Searchable Symmetric Encryption (RSSE) schemes that allow the execution of range queries in a practical, efficient, and secure manner. We propose a number of new RSSE schemes, that we analytically prove to be adaptively secure according to a novel, cryptographic security definition (RQ-CKA2), and also exhibit interesting security and performance trade-offs. We also tackle the challenge of updates in our RSSE schemes by proposing a general solution that does not introduce any additional leakage over the static case, other than the number of inserts/deletes. The practicality and scalability of our proposed schemes is demonstrated both theoretically and experimentally. More specifically, our techniques outperform state-of-the-art Privacy Preserving Range Querying approaches in terms of both security and efficiency and, at the same time, offer worst-case guarantees on possible leakages and also protect sensitive information regarding the order of encrypted values.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor, Professor Minos Garofalakis, for entrusting me to explore the world of encryption, supervising and motivating me during this thesis. His profound insight and fine suggestions walked me through every step of this work. I would also like to thank Associate Professor Antonios Deligiannakis for substantially contributing to my progress via our impeccable cooperation, his moral and practical support and our fruitful conversations.

I remember the encouragement, the knowledge and the motivation that Professor Stavros Christodoulakis gave me in the area of Databases, and not only, and for that I would like to thank him. He was a true, motivating and genuine mentor for me.

Next, I am sincerely grateful and I acknowledge the prolonged period of time that doctors Odysseas Papapetrou and Stavros Papadopoulos dedicated, in order to enlighten me throughout this whole process, as well as for their thoughtful and expertise commentary.

I would also like to seize the opportunity and thank professors Minos Garofalakis, Antonios Deligiannakis, Stavros Christodoulakis, as well as Stavros Papadopoulos for their immense and genuine support and guidance during the process of applying for a Ph.D. position.

My parents, Michalis and Chrysanthi, as well as my younger brother, Athanasios-Rafail who stood by me and therefore I would like to thank them for their love and compassion.

I am thankful to my precious Sofia Nikolakaki who has been encouraging and supporting me a lot, in the last few years.

Last but not least I would like to sincerely thank my good friends I. Perros and M. Orfanoudakis who stood beside me through challenging times and who offered me some incredible moments, as well as my friends from Chania, M. Alimpertis, S. Bourdakis, K. Douzis, K. Kalaitzis, N. Kofinas, T. Magounaki, D. Makris, K. Makris, P. Malakonakis, A. Markopoulos, D. Paliatsa, A. Soula, and G. Vlachantonis for all the moments, surprises and experiences that we have shared together.

Contents

List of Figures

Chapter 1

Introduction

Cloud computing is regarded as a swiftly emerging computing trend, since its usage has rapidly increased throughout the last few years. This augmented interest has appeared due to benefits that cloud computing offers, such as scalability and elasticity, the cost reduction for small and medium businesses, the easy access on data that has been stored. In general, early adopters are low risk applications that involve less sensitive data and therefore it is not common to move private data to the cloud side yet. It is essential that we first resolve all security issues and that privacy is guaranteed at all times. In order to succeed, no information should be provided to the server about the actual values of the data that has been stored and that is why encryption is implemented on the client side before uploading the data; querying takes place on the encrypted data without requiring decryption (on the cloud side). The main challenge is how to query the encrypted data efficiently and securely, especially in the case of two essential types of queries, that is the point and range queries. We focus on the problem of privacy preserving range querying, as it has so far been an open problem; none of the prior works provides a practical, efficient and provable secure solution.

Various approaches address the problem of querying encrypted data. More specifically, numerous works offer efficient solutions for this purpose, but lack provable secure guarantees [1, 2, 3, 4]. Several other works have adopted the use of Property Preserving Encryption (PPE) schemes. A special case of PPE schemes, called Deterministic Encryption Schemes (DET), have been utilized in order to support equality checks and equi-joins on encrypted data [5, 6, 7, 8]. Another type of PPE schemes is the Order Preserving Encryption (OPE)schemes, which have attempted to tackle the problem of

order comparisons and range queries [9, 10, 11, 12]. In general, both DET and OPE schemes provide the required practicality and efficiency but suffer from weak security guarantees, as well as from frequency analysis attacks; that is the most common attack in cryptography. They also suffer from other severe leakages, such as order relations and statistical information, which renders them impractical for real-world applications. Another suggested approach is the use of Predicate Encryption which either offers strong security guarantees by introducing high space overhead and linear search time [13, 14], or provides more efficient solutions [15] by leaking however essential statistical information i.e. the distribution of the input data. The use of Oblivious RAMs [16, 17] provides optimal security guarantees without leaking any information to the server, not even the access pattern, i.e. the number of times a document contains a specific keyword. The main obstacle in adopting Oblivious RAMs for encrypted query processing is that in practice they have been proved to be less efficient due to a high bandwidth cost overhead. Even the most recent and efficient Oblivious RAM approaches [18, 19, 20] cannot be utilized for answering privacy preserving point and range queries, as they rely on large block sizes, e.g. 4KB.

The approach of Searchable Symmetric Encryption schemes has appeared in the area of privacy preserving keyword search. In particular, such schemes allow the user to store data on an untrusted server or cloud provider with the purpose of securely searching this set of data later on, for records or documents that match a specific keyword, while maintaining privacy. It can be derived that SSE schemes can be used for point querying with a simple transformation. The main advantage of SSE schemes is that these schemes are not only secure encryption schemes, but also they have been proved to be secure searchable encryption schemes. In 2006 Curtmola et al. proved that all preceding security definitions targeted for SSE schemes were not sufficient for the general case of searchable schemes, and therefore they presented the state-of-the-art security definitions. More specifically, they provided the non-adaptive and adaptive security definitions. The former ensures privacy for all queries that are generated at once, whereas the latter also ensures privacy for queries performed at different times. Even though the research area of SSE schemes is very crowded, there is only one work that provided a Range Searchable Symmetric Encryption scheme, that of Li et al. in [21].

Our proposed schemes outperform the work of Li in terms of efficiency, and in addition our main advantage is that our approaches provide stronger security guarantees than

those in the work of Li.

1.1 Thesis Contribution

This work proposes the first adaptive Range Searchable Symmetric Encryption schemes that allows the execution of range queries on encrypted data in a practical, efficient and secure manner. In particular:

1. Our approaches are based on using SSE schemes as black-boxes, thus allowing us to take advantage of the existing state-of-the-art SSE schemes both in terms of security and efficiency.

2. We extend the state-of-the-art security definitions proposed by Curtmola et al. [22] to the range searchable problem which renders us the first to present a formal and unified security definition for RSSE schemes.

3. We present various RSSE schemes, which are adaptively secure and offer trades-offs between security and performance.

4. All of our proposed RSSE schemes require sub-linear search time and reasonable storage demands, except from the Quadratic protocol which is only presented for understanding purposes. In particular, interactive STRC reaches optimal search time, while Linearthmic-URC & MDC and Linear-URC achieve logarithmic search time.

5. We tackle the problem of updates in our RSSE schemes by providing a general solution that does not introduce additional leakages.

6. Our schemes are the first adaptively secure RSSE schemes that outperform the Related Work both in terms of efficiency and privacy. The only work similar to ours is presented by Li et al. in [21], who proposed their own RSSE scheme which is secure for non-adaptive SSE security definitions. Our schemes are more efficient and secure than those of Li et al. [21], and therefore we outperform their work.

7. We present the first privacy preserving range query schemes that hide sensitive information about the order of the encrypted results and that cannot be attacked by [23, 24]. These schemes are the Quadratic, Linearthmic-URC, as well as the non-interactive and interactive STRC RSSE schemes.

8. We conduct a performance evaluation of our schemes that shows their practical performance, thus verifying the theoretical complexities.

We stress out that our work is the first work that can be utilized by practical applications while maintaining strong security guarantees that quantify in advance the worst case leakages.

1.2 Thesis Outline

Chapter 2 presents the concept of Dyadic Intervals 2.1 since it is a basic structure used by the Linearthmic and Linear RSSE schemes in order to organize the data in the cloud. This section also presents the notions of deterministic and randomized encryption schemes 2.3, as well as the notions of PRFs and DPRFs 2.4. At the end of this chapter we present Searchable Symmetric Encryption (SSE) schemes along with their syntax and security analysis, since they are used as building-blocks for our proposed RSSE schemes. Chapter 3 addresses the problem of performing privacy preserving range queries. In particular, the related work that has studied the above problem is outlined and in addition we indicate their limitations. Chapter 4 discusses in detail our approach which focuses on answering encrypted range queries in a guaranteed secure manner by using state-of-the-art SSE schemes as black-boxes. More specifically, we propose an appropriate security game for RSSE schemes in 4.1, the Quadratic RSSE scheme as a warm-up in 4.2 and four novel and adaptive RSSE schemes which are the following Linearthmic URC 4.3 , non-interactive STRC 4.4, Linear-URC 4.5 and interactive-STRC 4.6. In the same chapter, we describe a general solution that allows our novel RSSE schemes to handle updates without leaking any additional information. Chapter 5 presents the experimental evaluation of our proposed algorithms. Finally, in Chapter 6 we conclude the master thesis and outline potential directions for future work.

Chapter 2

Preliminaries

We first present some definitions and methods that are essential both for understanding the algorithms proposed in this thesis, as well as for being able to distinguish how these algorithms compare to alternatives developed in prior work. In particular, we first describe the notion of dyadic intervals in Section 2.1. Section 2.3 briefly overviews the deterministic and randomized encryption schemes that have been widely used in different applications. Section 2.4 presents the notion of PRFs and DPRFs, which are used in our algorithms. Finally, Section 2.5.1 describes the notion of an SSE scheme, which is used as a black-box building block in our algorithms, and compares it to deterministic encryption approaches.

2.1 Dyadic Intervals

Dyadic intervals of a specific domain are intervals arranged in a specific hierarchical structure. According to the definition of dyadic intervals, a dyadic interval always has length equal to an integer power of two, it is included in one and only parent node and it comprises two children nodes which are also dyadic intervals. The above properties apply to all nodes, except from the leafs, since they correspond to unary intervals. The structure of a **dyadic tree** is shown in Figure 2.1. A formal definition of the dyadic interval follows:

Dyadic Interval Definition: A **dyadic interval** over a domain $I = 1, 1, ..., N, |I| = N = 2^n$ is an interval of the form $[q2^j+1, (q+1)2^j]$, where $0 \leq j \leq n$ and $0 \leq q \leq 2^{n-j}-1$.

Minimal dyadic cover Definition ($D([\alpha, \beta])$): The minimal dyadic cover of an interval

$[\alpha, \beta]$, is the set of dyadic intervals $\delta_1, \delta_2, ..., \delta_m$ for the minimum value of m, so that $\delta_1 \cup \delta_2 \cup ... \cup \delta_m = [\alpha, \beta]$.

Dyadic cut point: $d([\alpha, \beta]) = q2^j$ is the **dyadic cut-point** of an interval $[\alpha, \beta]$ if j is the largest integer smaller than n ($|I| = 2^n$) for which there exists q such that $q2^j \in [\alpha, \beta]$.

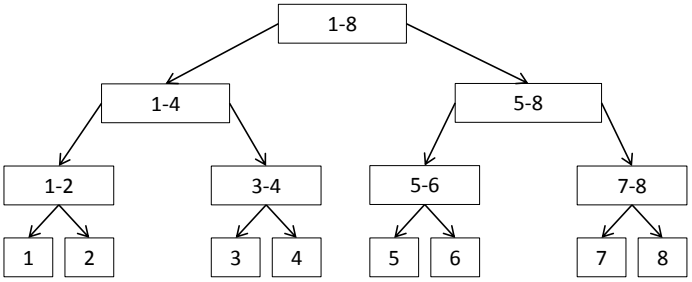


Figure 2.1: Dyadic Interval tree

We introduce some useful properties of the Dyadic Intervals.

Property 1: There are exactly 2^j dyadic intervals at level j, each containing 2^{n-j} points from I.

Property 2: The dyadic intervals in a minimal dyadic cover of an interval, form a partition of that interval.

Property 3: The dyadic cut-point of an interval is unique.

Property 4: The minimal dyadic cover of an interval $[\alpha, \beta]$ has cardinality at most 2(n-1), where $|I| = 2^n$. That is $|D([\alpha, \beta])| \leq 2(n-1)$

Lemma 1: Let $[\alpha, \beta]$ be an interval, $d([\alpha, \beta]) = q2^j$ be its dyadic cut point, and $D([\alpha, \beta])$. Then:

1. None of the dyadic intervals in $D([\alpha, \beta])$ contain the dyadic cut-point, except as a right end-point.

2. The minimal dyadic cover $D([\alpha, \beta])$ is the union of the minimal dyadic covers of $[\alpha, q2^j)$ and $[q2^j, \beta]$, i.e. $D([\alpha, \beta]) = D([\alpha, q2^j)] \cup D([q2^j + 1, \beta])$

3. The minimal dyadic cover $D([q2^j + 1, \beta])$ consists of a sequence of at most j dyadic intervals with strictly decreasing sizes.

4. The minimal dyadic cover $D([\alpha, q2^j]$ consists of a sequence of at most j dyadic intervals with strictly increasing sizes.

5. The minimal dyadic cover $D([\alpha, \beta])$ contains at most $2j$ dyadic intervals, from which

at most two belong to the same level.
More properties, as well as definitions and proofs are presented in [25].

2.2 Adversary Models

We present the description of adversary models, due to the fact that their very existence led to the necessity of exploring privacy preserving approaches. Potential adversaries are the cloud service provider itself, its insiders and any third party attackers who are capable of viewing the target data, monitoring query processing on the data, obtaining or inferring data and queries, as well as gaining access to the database server. Assuming that the adversary is the service provider (untrusted server), which is the worst case scenario, she falls under one of the following categories.

Passive or curious Adversary but not malicious: The attacker in this model is able to obtain and derive data and queries; she has complete access to the database server. She is capable of perceiving the distribution of the ciphertexts if the respective leakage of information occurs, thereby inferring access patterns or query results. Her goal is to gather as much information she can.

Active or malicious Adversary: The difference of this type of adversary, compared to the previous one is that she may misbehave and affect the query processing. More specifically, she has the ability to modify the encrypted data or to alter the answer of the queries that have been submitted by the user.

2.3 Encryption Schemes

At this point we briefly present important notions regarding the operation of a deterministic symmetric encryption scheme (DET), that is an equality property preserving encryption scheme (which leaks equality property), as well as important concepts concerning randomized encryption schemes (RND).

DET: DET consists of 3 parts, a key generator, an encryption algorithm and a decryption algorithm. In order to represent the above parts of DET, we use the notation: DET = (DET.KeyGen,DET.Enc,DET.Dec).
DET.KeyGen: KeyGen receives as input λ, which is a security parameter and outputs

a secret key sk. ($sk \leftarrow KeyGen(1^{\lambda})$)

DET.Enc: Enc is the encryption function that receives as input a secret key sk and a message m and outputs a ciphertext ($c \leftarrow Enc(sk, m)$)

DET.Dec: Dec is the decryption function that receives as input a ciphertext c and a secret key sk and outputs the message ($m \leftarrow Dec(sk, c)$). The following property describes DET:

$$\forall\ m_0, m_1\ \in \mathcal{M}:\ Enc(sk, m_0) = Enc(sk, m_1) \text{ iff } m_0 = m_1$$

DET cannot be secure against chosen-plaintext attacks (CPA), since due to deterministic encryption the equality of the plaintexts is leaked. We refer to any deterministic encryption scheme that is proved to be secure against distinct chosen-plaintext attacks (DCPA) (DCPA that arises by weakening the CPA security definition), as DET. For further details we refer the reader to [26]. More specifically, only information concerning the equality property is leaked from the ciphertexts (i.e., the encrypted messages), as two identical ciphertexts must have been produced by the same input message (plaintext).

RND: Similarly to DET, RND also comprises a key generator, an encryption algorithm and a decryption algorithm. We represent the three parts of RND, with the notation: RND = (RND.KeyGen,RND.Enc,RND.Dec).

The definitions of RND.KeyGen, RND.Enc and RND.Dec are similar to those of the DET encryption scheme, but in this case two equal plaintext messages are mapped to different cipertexts with overwhelming probability, i.e.:

$$\forall\ m_0, m_1\ \in \mathcal{M} : m_0 = m_1,\ Enc(sk, m_0) \neq Enc(sk, m_1)$$

The RND encryption is a randomized encryption scheme shown to be secure against chosen plaintext attacks (CPA), which means that the ciphertexts do not leak any information about the plaintext that was encrypted. Thus, RND encryption schemes achieve stronger security guarantees than DET encryption schemes, since they do not reveal plaintext equalities.

For more details about DET,RND and CPA, as well as DCPA we refer the reader to [27] and [26].

2.4 PRFs and DPRFs

We now provide useful definitions regarding *pseudorandom functions* (PRFs) and *delegatable pseudorandom functions* (DPRFs). PRFs constitute an essential component of the SSE schemes that are presented in Section 2.5.1. DPRFs are an extension of PRFs, proposed by [28].

Intuitively, a PRF is a function in which the input-output functionality cannot be distinguished from a random function using any polynomial time computation. A PRF is a function $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^m$ that is computable in polynomial time and cannot be distinguished from other random functions by polynomial time adversaries.

In [28], Kiayias et al. introduce the notion of delegatable pseudorandom functions (DPRFs). DPRFs are cryptographic primitives that enable the delegation of the evaluation of a PRF to an untrusted server according to a given predicate that defines the inputs on which the server will evaluate the PRF. Intuitively, a new trusted entity, termed as the delegator, is used in order to determine the set of encrpyted inputs (termed as *trapdoors*) that the server receives in each query. In particular, for any secret key k and a predicate $P \in \mathcal{P}$, the delegator computes a trapdoor τ via algorithm T, and τ is transmitted to the server. Let n denote the size in bits of any input string x, i.e., $x = x_{n-1}...x_0$. A dyadic interval of range 2^r can be specified by providing the $n - r$ most significant bits of x. In this case, the trapdoors provided to the server are computed by the delegator based on these $n - r$ most significant bits and the secret key k. Given each trapdoor, [17] demonstrates how the server can compute the PRF value of any value within that dyadic interval.

More specifically, the construction of [28] is based on the GGM pseudo random function family [29] defined by

$$F = f_k : \{0,1\}^n \rightarrow \{0,1\}^\lambda{}_{k \in \{0,1\}^\lambda},$$
$$\text{such that}$$
$$f_k(x_{n-1}...x_0) = G_{x_0}(...(G_{x_{n-1}}(k)))$$

where n is polynomial in λ and $x_{n-1}...x_0$ is the input bit string of size n.

2.5 Searchable Symmetric Encryption (SSE) schemes

2.5.1 Privacy Preserving Point Queries (DET Vs SSE)

The problem of privacy preserving point queries could also be solved with two different approaches. We begin by discussing the problem of secure point querying because we intend to use the secure privacy preserving point querying protocol as a basis for our secure range querying approaches.

Many previous works from the DB community apply the DET encryption scheme, in order to support equality checks and equi-joins [5, 6, 7, 8]. DET is a special case of Property-Preserving encryption (PPE) that leaks information associated with the equality property (two equal ciphertexts imply two equal plaintexts). The data owner encrypts the database using the DET encryption scheme. Once the encryption of the database is completed, the data owner outsources the EDB (Encrypted DataBase) to the cloud and the query processing follows. The query processing comprises several steps. First, the client or data owner implements the DET encryption scheme to encrypt the point query value, under the same secret key (sk_d) as the one used during the encryption phase. Then, she sends the ciphertext to the cloud provider, who is responsible for scanning the entire database and for returning all ciphertext values of the EDB that match the query-ciphertext. The above approach introduces certain limitations, such as the searching time on the cloud that grows linearly as the number of tuples increases. A straightforward solution to address this issue is to organize the EDB using a data structure over the already encrypted value-column, thereby allowing to achieve sub-linear search cost. The main drawback of the DET approach is the vulnerability of the EDB to **frequency analysis attacks**, which is the most common attack in cryptography. The frequency analysis attack problem has not been addressed by any approach proposed by the DB community. Two of these approaches, CryptDB and Monomi claim that they use DET to increase functionality. However, this renders them vulnerable to statistical attacks, as it implies leaking equality messages and achieving weaker security guarantees. They also attempt to support range queries, but with the use of a weaker PPE scheme. Nevertheless, in this case both the equality messages and the order of the plaintexts are revealed, simply by observing the ciphertexts (OPE encryption schemes).

Scheme	Security	Search	Storage	Upd.
Song et.al '00 [30]	CPA	$O(n)$	N/A	no
Goh '03 [31]	non-adaptive	$O(n)$	$O(n)$	yes
Chang et.al '05 [32]	non-adaptive	$O(n)$	$O(n)$	no
Curtmola et.al '06 (SSE-1) [22]	non-adaptive	$O(r)$	$O(nm)$	no
Curtmola et.al '06 (SSE-2) [22]	adaptive	$O(r)$	$O(n+m)$	no
Liesdonk et.al '10 [33]	adaptive	$O(logm)$	$O(nm)$	yes
Chase et.al '10 [34]	adaptive	$O(r)$	$O(nm)$	no
Kamara et.al '12 [35]	adaptive	$O(r)$	$O(n+m)$	yes
Cash et.al '13 [36]	adaptive	$O(r)$	$O(n+m)$	no
Cash et.al '14 [37]	adaptive	$O(r)$	$O(n)$	yes
Stefanov et.al '14 [38]	adaptive	$O(r)$	$O(n+m)$	yes

Table 2.1: Comparison of SSE schemes, where n is the data size (the number of tuples), r the number of documents containing the query word w, m the size of the keyword space

DET encryption schemes allow search-ability, but they cannot provide strong provable secure solutions. We present the **Symmetric Searchable Encryption scheme (SSE)** which tries to solve the keyword search problem, over a collection of encrypted data that can be outsourced to cloud providers; this scenario is very similar to the one of privacy preserving point querying, where words are mapped to searchable values and document-ids to tuple-ids contained in a particular searchable value.

Table 2.1 summarizes the most representative works on SSE schemes, along with their security guarantees and corresponding costs.

The most efficient and secure solutions are the static version of the Kamara et al. approach [35], the approaches of Cash et al. [36] and [37].

2.5.2 Syntax of SSE schemes

In this section, we present the syntax of an SSE scheme. The use of such syntax allows us to consider any SSE scheme as a "black-box".

An indexed-based SSE scheme over a dictionary Δ consists of five polynomial time algorithms (SSE = (KeyGen, Enc, Trpdrm, Search, Dec)) which are the following:

SSE.KeyGen is a probabilistic key generation algorithm that receives as input a security parameter k and outputs a secret key K

SSE.Enc(K,DB) is a probabilistic algorithm that receives a key K and a database so as to output the encrypted database (EDB) which contains a secure index I.

SSE.Trpdr(K,w) is a deterministic algorithm that generates a trapdoor for a given keyword. In particular it receives as inputs a key K and a keyword w, in order to output a trapdoor t.

SSE.Dec(K,c_i) is a deterministic algorithm, which receives as inputs a secret key K and a ciphertext c_i, and outputs the corresponding plaintext.

Note that the client or the data owner executes the above algorithms. The same does not apply for the following algorithm, which is carried out by the server (or the cloud provider).

SSE.Search(EDB,t) is a deterministic algorithm, which receives as inputs the encrypted database, the secure index I, which is contained in EDB and a trapdoor t, produced by a specific word w_*. Furthermore, it outputs a set of all ciphertexts that comprise the word w_*.

2.5.3 Security Analysis of SSE schemes

In [30] Song et al. lay the foundation for many subsequent works in the area of SSE schemes. Their work is secure under Chosen Plaintext Attacks (CPA), a powerful security definition which cannot be achieved with the use of a DET encryption scheme. Later, Goh in [31] explained why CPA security is not adequate for the case of SSE schemes but the work he proposed was not secure either. In [22], Curtmola et al. introduced the state-of-the-art security guarantees for any SSE scheme and provided solutions to achieve these security guarantees. For further discussion about the evolution of SSE schemes, we refer the reader to a recent survey about SSE schemes [39]. The conclusion of this section is that DET approaches present severe leakages, and thereby cannot be used for answering point queries in a secure manner.

In more detail Curtmola et al. proved in [22] that all the previous security definitions for SSE schemes did not provide adequate security guarantees. Curtmola also introduced two new adversarial models for SSE, which are widely used as state-of-the-art definitions for SSE to date. The first is the non-adaptive (IND-CKA1), which guarantees the security

of a scheme if the adversary (or the party who runs the queries) generates all queries at once. The adaptive (IND-CKA2) is considered a strong security definition for SSE, due to the fact that it overcomes the aforementioned limitation of IND-CKA1. An additional contribution of Curtmola et al. in 2011 [40] was that they transformed the above indistinguishability security definitions into proven simpler equivalent ones. These definitions are known as semantic security definitions. For simplicity reasons we use semantic security definitions , which are presented below.

(**Adaptive semantic security**). Let SSE = (Gen, Enc, Trpdr, Search, Dec) be an index-based SSE scheme and consider the following probabilistic experiments, where A is a stateful adversary, S is a stateful simulator and L_1, L_2 are stateful leakage algorithms:

Real*$_{SSE,A}(k)$
$K \leftarrow \text{Gen}(1^k)$
$(\mathbf{D}, st_A) \leftarrow A_0(1^k)$
$(I, \mathbf{c}) \leftarrow \text{Enc}_K(\mathbf{D})$
$(w_1, st_A) \leftarrow A_1(st_A, I, \mathbf{c})$
$\text{t}_1 \leftarrow \text{Trpdr}(w_1)$
for $2 \leq i \leq q$
  $(st_A, wi) \leftarrow A_i(st_A, I, \mathbf{c}, t_1, ..., t_{i-1})$
    $\text{t}_i \leftarrow \text{Trpdr}(w_i)$
end
let $\mathbf{t} = (t_1, ..., t_q)$
$A_q(st_A, I, \mathbf{c}, \mathbf{t})$ returns a bit b,
which is the output of the experiment

Sim*$_{SSE,A,S}(k)$
$(\mathbf{D}, st_A) \leftarrow A_0(1^k)$
$(I, \mathbf{c}) \leftarrow S_0(L_1(D))$
$(w_1, st_A) \leftarrow A_1(st_A, I, \mathbf{c})$
$(\text{t}_1, st_S) \leftarrow S_1(st_S, L_2(\mathbf{D}, w_1))$
for $2 \leq i \leq q$
  $(st_A, w_i) \leftarrow A_i(st_A, I, \mathbf{c}, t_1, ..., t_{i-1})$
  $(\text{t}_i, st_S) \leftarrow S_i(st_S, L_2(\mathbf{D}, w_1, ..., w_{i-1}))$
end
let $\mathbf{t} = (t_1, ..., t_q)$
$A_q(st_A, I, \mathbf{c}, \mathbf{t})$ returns a bit b,
which is the output of the experiment

We say that SSE is adaptively semantically (L_1, L_2)-secure if for all polynomial-size adversaries $A = (A_0, ..., A_q)$, such that $q = poly(k)$, there exists a non-uniform polynomial-size simulator $S = (S_0, ..., S_q)$, such that

$$|Pr[\mathbf{Real}_{SSE,A(k)} = 1] - Pr[Sim_{SSE,A,S(k)} = 1]| \leq negl(k)$$

where the probabilities are over the coins of Gen and Enc.
The L_1 leakage function:

$$L_1(\mathbf{D}) = (|I|, [id(w)]_{w \in \Delta}, [id(D)]_{D \in \mathbf{D}}, [|D|]_{D \in \mathbf{D}})$$

where $|I|$ is the size of the index, $[id(w)]_{w\in\Delta}$ denotes the identifier of each range in the space of the keywords (Δ, $[id(D)]_{D\in\mathbf{D}}$ represents the identifier of each unique document and $[|D|]_{D\in\mathbf{D}}$ is the size of each document.
The L_2 leakage is defined by

$$L_2(\mathbf{D}, w) = (ACCP_t(w), id(w))$$

where $ACCP_t(w)$ is the access pattern defined by the sequence $(id_1, ..., id_{\#D_w})$.

Note that the SSE-1 scheme proposed by Curtmola et al. in [22], the SSE scheme introduced by Chase et al. in [34], as well as the static versions of the SSE scheme suggested by Kamara et al. in [35] take into account these leakages. The approaches of Cash et al. [36, 37] minimize the L_1 leakage, by restricting the leaks to an upper limit for the number of input data.

More intuitively, the above security definition comprises the execution of a real and a simulated protocol. The real protocol executes the exact corresponding SSE scheme directly on the input data (which is outputted by the adversary), whereas the simulated protocol attempts to simulate the real protocol without having any access to the input data. The simulator receives as input **only** the information contained in the leakage functions. Therefore, if there exists a simulator that can correctly simulate the SSE scheme by using functions L_1, L_2, while the adversary is outputting adaptive queries, then the SSE scheme is adaptively secure and the worst case leakage is the information contained in L_1, L_2.

Chapter 3

Related Work

This section presents an overview of research conducted on privacy preserving range queries.

3.1 Range Searchable Symmetric Encryption Scheme

Li et al. in [21] proposed the first range searchable symmetric encryption scheme. This approach extends a solution proposed by Goh in [31] for the case of range queries. The main drawback of the proposed scheme is its inadequate security level, since it does not use the state-of-the-art security definition proposed by Curtmola et al. in [40]. In contrast, they only prove index indistinguishabilty by using the IND-CKA security definition introduced by Goh in [31], which however has several limitations. More specifically, not only does IND-CKA assume a non-adaptive adversary, but also it does not require secure trapdoors. Curtmola et al. proved in [40] that even IND2-CKA proposed by Chang and Mitzenmacher, which is stronger than IND-CKA can be trivially satisfied by any non secure SSE scheme. Specifically in the case of privacy preserving range queries, query privacy is essential as the tokens may disclose the queried range. In such cases the proposed scheme cannot achieve any kind of token privacy, and to prove this we provide the following toy attack. If the trapdoor matrix has $2(logm - 1)$ rows, then the adversary knows that the queried interval is $[1, m - 1]$ where m the size of the domain, e.g. given m=8 the adversary notices that if the number of rows equals 4, then the queried range is [1,6]. In terms of efficiency the proposed algorithm requires time complexity $O(|R|logn)$, where R is the number of data items in the query result, the token size is a matrix with

dimensions w*r, where r is the number of hash functions in the bloom filters and w the number of dyadic intervals ($w \in [1, 2 * logn - 1]$). The same algorithm has space complexity equal to $O(\frac{rn}{ln2} logn)$. This approach further proposes 2 optimization algorithms in order to improve the search time by minimizing the number of nodes that a query needs to traverse. These are the Traversal width and the Traversal Depth optimization algorithms. However, the first has a worst case complexity equal to $O(n^2)$ and also neither of the two optimization algorithms is included in any security analysis.

3.2 Order Preserving Encryption

In the recent years significant research is conducted around the subject of Order Preserving Encryption Schemes (OPE) [41, 42, 26, 9, 43, 10, 44, 45, 11, 46, 47, 48, 49, 50, 12]. OPE schemes are deterministic encryption schemes that preserve the order of the plaintext, allowing the execution of efficient range queries directly on encrypted data on the cloud. An ideal security guarantee for OPE schemes meets the IND-OCPA security definition, ensuring that no other information is revealed besides the order of the plaintext values. Among the family of OPE schemes, only the recent works of Popa et al. in [12] and Kerschbaum et al. [51] achieve IND-OCPA, while the rest achieve weaker security guarantees and appear to have further leakages besides the order. However, all OPE schemes, including [12], [51], have two major drawbacks which are the the disclosure of the order and the distribution leakage of repeated ciphertexts due to determinism. Therefore, an adversary who is either aware of the domain of the encrypted values or gathers statistical data is capable of building a mapping between the actual and the encrypted values.

3.3 Bucketization Approaches

Hacigumus et al. [4] proposed a bucketization based approach that Hore et al. [1] further extended and improved. The bucketization technique presupposes data partitioning into buckets that are eventually stored in the cloud. The client side retains the respective indices, whose number increases linearly with the number of buckets, also affecting the index search process in the same manner. Whenever a range query is to be executed all buckets containing query results are retrieved from the server. However, false positive

values are also included in the retrieved buckets and need to be filtered out from the final answer. This process is carried out on the client side and requires the decryption of all tuples contained in the buckets leading to an often prohibitive cost. Furthermore, updating data is expensive since it requires re-distribution of the tuples. More specifically, it tunes security to the desired level with the cost of degrading efficiency respectively (use of the trade-off between efficiency and privacy).

Wang et al. in [3] create Rp-trees, hierarchical encrypted indices that allow the execution of efficient range queries. They also use the Asymmetric Scalar-product Preserving Encryption approach proposed by [52], in order to perform the bucket encryption (R-tree node) process. Furthermore, they conduct a marginal analysis based on the impact of ordering information leakage on the matter of privacy. They emphasize on the fact that encryption schemes which reveal order information can lead to unacceptable privacy loss and that renders them impractical for real applications. The reduction of ordering information leakage implies the existence of false positives, a trade-off that is tuned by selecting an appropriate tree structure. However, an increased number of false positives burdens the client side in a manner that was previously mentioned.

In terms of security these approaches do not provide any proof.

3.4 Predicate Encryption

Predicate encryption query schemes guarantee strong security definitions, while causing at the same time high computational overhead. In predicate encryption schemes a ciphertext is associated with a set of hidden attributes S. We define a predicate function f() and a token created by the master secret key. Using this token in function f() the client can check whether this secret set for a ciphertext is satisfied (f(s)=1) without decrypting the data.

Boneh and Waters [13] proposed public key based encryption schemes that can support conjunctive subset equality and range queries. Note that, this scheme hides the attributes for messages that match a query. This is defined in [14], as the match concealing property. The complexity of a data record is linear with the range size, which is inefficient in the case of big ranges. This encryption scheme also suffers from big ciphertext sizes.

Shi et al. in [14] provided a relaxation on Boneh's encryption scheme on security guarantees. This scheme cannot hide attributes from messages that are matched by a

token, defined as Matched Revealing. Also, this approach requires linear scan of the data, while revealing the plaintexts of the queried range.

Lu proposed in [15] a predicate encryption scheme with logarithmic time search complexity. As she mentioned, her method suffers from the frequency analysis attack and the query access pattern problem.

The aforementioned approaches guarantee Data Confidentiality, but they do not provide any Access Privacy guarantees. The attack presented in [23] can occur in any of the previous approaches.

3.5 FHE - HE

Homomorphic cryptosystems(HE) cannot support privacy preserving range queries, since an appropriate range query homomorphic cryptosystem does not exist. Fully homomorphic encryption (FHE) scheme, proposed by Gentry [53] is a generalization of homomorphic encryptions, that allows the execution of any arbitrary function, including range queries on encrypted ciphertexts, without requiring any decryption. Yet, it appears to be totally impractical due to the required ciphertext size and computational time that sharply increase as we increase the security level.

3.6 Access Privacy

Querying on data may lead to the reveal of sensitive information about the data and in such cases access privacy is essential. Among the proposed protocols guaranteeing Access Privacy, the most prominent ones are Private Information Retrieval [54] and Oblivious RAM [16] from the Crypto community. Note, that these approaches are potential solutions to the aforementioned problem, but they introduce an additional computational overhead which in practice is prohibitive.

Private Information Retrieval is a memory structure that allows retrieving data from remote database servers securely, without disclosing the selected item [54]. Certain PIR solutions suggest using a single server [55] and others involve multiple servers [54]. Early efforts proposed single server-based PIR solutions that required a considerable computational overhead [56]. However, research conducted on recently developed PIR solutions showed that this protocol can also be efficient [57].

Using Oblivious RAM on the cloud server is one way to yield practical PIR solutions, as proposed in [58]. The main concept of Oblivious RAM is to shuffle and rearrange data items in the RAM, while data accesses occur.

Some alternative methods have been suggested, that aim to provide secure and efficient access to the data. Such methods are Index Shuffling and Covered Search that protect accesses on encrypted indices [59], and hybrid approaches that employ PIR on partial data [60].

Chapter 4

Our approach

In this section we propose four novel algorithms that satisfy the security model described in Section 4.1. All algorithms are adaptive, i.e. queries can be executed in both manners sequentially and continuously as opposed to previous works, e.g. [21], which presume that all queries are executed together in a single batch. The common approach followed by all these works is the utilization of SSE schemes to enable the composition of any possible range query by a small set of chunks that are indexed and accessed efficiently and securely. The algorithms differ on the methods of deriving the chunks and executing the queries, providing different security and performance guarantees.

Wlog., our discussion assumes that the data is initially contained in a relational table. The table contains two attributes, the *tuple-id*, which contains identification data for the record (e.g., name, social security id) and the *tuple-value* that contains the numerical data on which we want to allow range queries (e.g., salaries, dates converted to integers). [1] For example, in an outsourced personnel repository, the tuple-id could map to employee-id, whereas the tuple-value could map to the emplyee's salary, age, or any other sensitive numerical information.

[1] This tuple construction assumes that we want the final – decrypted – answer to contain the ids corresponding to the queried range. However, depending on the setup, the id attribute could be extended to contain additional details, even the full record. Essentially, the id attribute is handled as a binary encrypted object by the algorithms, and therefore it can contain abritrary private information.

4.1 Security Game

In order to prove that our proposed Privacy Preserving Range Querying protocol is secure, we extend the state-of-the-art security definition of SSE schemes to the setting of range querying. In 2006 Curtmola et al. proved in [22] that all the previous security definitions for SSE schemes did not provide adequate security guarantees. Curtmola also introduced two new adversarial models for SSE, which are widely used as state-of-the-art definitions for SSE to date. The first is the non-adaptive (IND-CKA1), which guarantees the security of a scheme if the adversary (or the party who runs the queries) generates all queries at once. The adaptive (IND-CKA2) is considered a strong security definition for SSE, due to the fact that it overcomes the aforementioned limitation of IND-CKA1. More specifically, now the adversary is allowed to take into account the observations of previous queries in order to select the new queries, and thereby this empowers the adversary by allowing her to perform more sophisticated attacks. An additional contribution of Curtmola et al. in 2011 [40] was that they transformed the above indistinguishable security definitions into proven simpler equivalent ones. These definitions are known as semantic security definitions. For simplicity reasons we use semantic security definitions.

The key idea of our approach is that we separate a range into chunks, we associate each chunk with a label, and we use an SSE scheme to retrieve the separate chunks that constitute the queried range. In the case of range queries, we cannot directly use the security definitions targeted for SSE schemes. An intuitive explanation is that these security definitions are applied to word search problems. In such problems the client queries a word, and the server returns all documents that contain the queried-word. Yet, in the range queries problem the client desires to search within a specific range. We decompose this range into a union of multiple chunks (word-searches) according to the method described in Section 3. Note that in our approach the multiple chunk-searches are not independent, whereas the same does not necessarily apply to the word search problem. We extend the CKA2 definition to a range-querying CKA2-security (RQ-CKA2) definition, thus rendering it appropriate for a Range Searchable Symmetric Encryption (RSSE) scheme. Intuitively, the above definition supports that an RSSE scheme is adaptively secure, if the adversary cannot distinguish a real RSSE protocol from a simulated one. Note that the L-functions (Leakage functions) used by the Simulator,

represent the total leakage that can be observed by any polynomial-time adversary during the execution of the proposed scheme.

Definition: RQ-CKA2 security. Let RSSE = (KeyGen, Enc, Trpdr, Dec, RangeSearch) be a index-based range SSE scheme and consider the following probabilistic experiments, where A is a stateful adversary, S is a stateful simulator and L_1, L_2 are stateful leakage functions:

Real*$_{RSSE,A}(k)$
$K \leftarrow \text{Gen}(1^k)$
$(\mathbf{D}, st_A) \leftarrow A_0(1^k)$
$(I, \mathbf{c}) \leftarrow \text{Enc}_K(\mathbf{D})$
$(\mathbf{F}(range_1),\ st_A) \leftarrow A_1(st_A, I, \mathbf{c})$
$\text{t}_1 \leftarrow \text{Trpdr}(\mathbf{F}(range_1))$
for $2 \leq i \leq q$
  $(st_A, (\mathbf{F}(range_i)) \leftarrow A_i(st_A, I, \mathbf{c}, t_1, ..., t_{i-1})$
  if **condition** then
    $\text{t}_i \leftarrow \text{Trpdr}(\mathbf{F}(range_i))$
  else
    skip the query, $\text{t}_i \leftarrow \emptyset$
end
let $\mathbf{t} = (t_1, ..., t_q)$
$A_q(st_A, I, \mathbf{c}, \mathbf{t})$ returns a bit b,
which is the output of the experiment

Sim*$_{RSSE,A,S}(k)$
$(\mathbf{D}, st_A) \leftarrow A_0(1^k)$
$(I, \mathbf{c}) \leftarrow S_0(L_1(D))$
$(\mathbf{F}(range_1),\ st_A) \leftarrow A_1(st_A, I, \mathbf{c})$
$(\text{t}_1, st_S) \leftarrow S_1(st_S, L_2(\mathbf{D}, \mathbf{F}(range_1)))$
for $2 \leq i \leq q$
  $(st_A, \mathbf{F}(range_i)) \leftarrow A_i(st_A, I, \mathbf{c}, t_1, ..., t_{i-1})$
  $(\text{t}_i, st_S) \leftarrow S_i(st_S, L_2(\mathbf{D}, \mathbf{F}(range_1), ..., \mathbf{F}(range_{i-1})))$
end
let $\mathbf{t} = (t_1, ..., t_q)$
$A_q(st_A, I, \mathbf{c}, \mathbf{t})$ returns a bit b,
which is the output of the experiment

RSSE is adaptively semantically secure -$(L_1, L_2, F, condition)$-*RQ-CKA2* - if for all polynomial-size adversaries $A = (A_0, ..., A_q)$, such that $q = poly(k)$, there exists a non-uniform polynomial-size simulator $S = (S_0, ..., S_q)$ such that

$$|Pr[\mathbf{Real}_{RSSE,A}(k) = 1] - Pr[\mathbf{Sim}_{RSSE,A,S}(k) = 1]| \leq negl(k)$$

where the probabilities are over the coins of Gen and Enc.

The leakage functions contain the total amount of information that is leaked. More specifically, L_1 and L_2 leakage functions are directly inherited from the SSE schemes. In

particular, the simulator uses L_1 in order to correctly simulate the encrypted index, while she uses L_2 to answer the range queries posed by the adversary.

Function F is defined as follows:

$$F : X \to Y,$$

X: all possible ranges and Y: all ranges $\in \Delta$ (dictionary)

4.2 Quadratic algorithm

As a warm-up we present *Quadradic*, a simple-naive algorithm, which offers strong security guarantees but requires quadratic space. At the site of the data owner, the algorithm first constructs and populates a new table that maps the tuple-ids to all possible query ranges on which their corresponding tuple-values are contained. For example, in the outsourced personnel repository, an employee with $salary = 1000$ would be contained in all possible ranges $[x, y]$ satisfying $\min \leq x \leq 1000$ and $\max \geq y \geq 1000$, where min and max correspond to the extremes of the domain of salary values. The algorithm will create a record for each such range, mapping tuple-id with a textual representation of the range, i.e., "[x,y]". The owner then generates a secret key K (**SSE.KeyGen**, cf. Section 2.5.2) and encrypts the table with it, using standard SSE encryption (**SSE.Enc**). The encrypted index is then uploaded to the untrusted server, e.g., the cloud machines. Querying of the encrypted index for any query range "[x,y]" is performed via standard SSE functionality, i.e., by producing a trapdoor from the query range and the secret key (**SSE.Trapdoor**), and sending this trapdoor to the SSE index for answering (**SSE.Search**). The results are received in encrypted form, and decrypted at the query initiator site (**SSE.Dec**).

Creating the index incurs a space and computational complexity of $O(nm^2)$, where n is the number of tuples at the initial database, and $m = \max - \min$ is the domain size of the value to be indexed, i.e., the salary. Querying, on the other hand, has a complexity of $O(r)$ for the untrusted server, where r is the size of the answer. (The query initiator only needs to send one query.) In terms of privacy, notice that the prescribed protocol inherits the privacy guarantees of the utilized SSE since it executes a single point query per range. The following theorem summarizes the properties of the approach.

Theorem 4.2.1. *The Quadratic algorithm is* $(L_1, L_2, F, true)$-***RQ-CKA2*** *secure if the utilized SSE scheme is* (L_1, L_2)-***CKA2*** *secure. The complexity of the algorithm is* $O(nm^2)$ *for building the index, where* $m = \max - \min$*, and* $O(r)$ *for query answering, with* r *denoting the size of the answer.*

Proof sketch: To begin with, we define function F as $F(range_x) = range_x$, as well as a simulator that proceeds as follows. Given the L_1 leakage, the simulator is able to simulate the data structures of the RSSE scheme. For each query q, recall that the simulator is given the L_2 leakage function. She uses the query pattern to determine if the query is new and therefore she meets two cases (1) the range has appeared before (2) the range has not appeared before. The simulator uses the L_2 function to determine in which of the aforementioned cases does the specific range query fall into. If it belongs to Case (1) then it returns the previously generated token, whereas if it belongs to Case (2) then the simulator randomly selects the token. It also needs to store the token that corresponds to a specific queried-range, as well as the random choice that was made for each range-token. □

4.3 Linearithmic algorithm

Even though the quadratic algorithm offers the optimal security guarantees, i.e., the ones offered by the underlying SSE scheme, its high storage complexity makes it impractical. To address this constraint we propose the linearithmic algorithm which requires only $O(n \log m)$ cost, i.e., a factor of $m^2/\log m$ less cost than quadratic. The key concept behind Linearithmic's performance is that any possible query range between min and max can be constructed by the union of at most $O(\log m)$ disjoint and sequential dyadic ranges [61], i.e., ranges of the form $[k2^j + 1, (k+1)2^j]$, for integers $j \in [0, n]$ and $k \in [0, 2^{n-j}]$, and $m = \max - \min$. For example, with $\min = 1$, query range $[3, 7]$ can be constructed by combining the dyadic ranges $[3, 4]$, $[5, 6]$ and $[7, 7]$, whereas query range $[2, 4]$ is constructed by combining $[2, 2]$ and $[3, 4]$.

The Linearithmic algorithm builds on this observation, i.e., it constructs an inverted index of dyadic ranges over the SSE scheme, and uses these to efficiently answer range queries. This index is essentially an unencrypted inverted index of the tree of dyadic ranges (cf. Fig. 4.1), yet without storing the links between the tree nodes and without

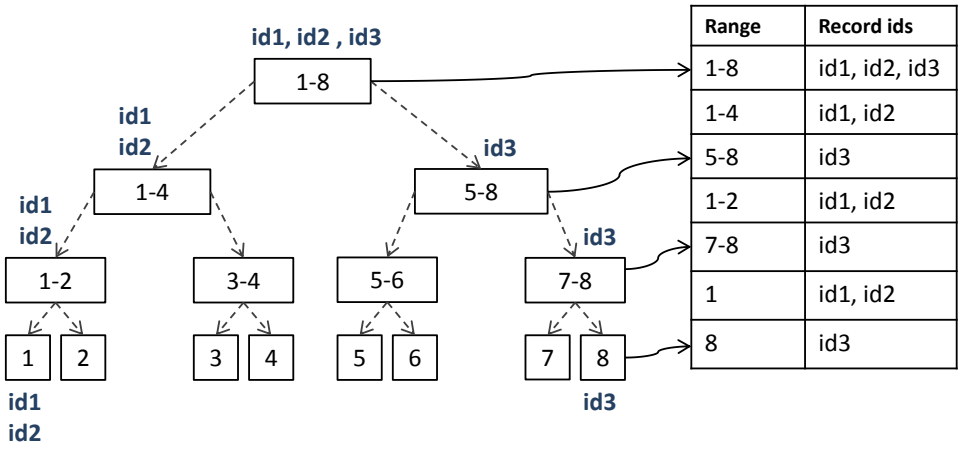


Figure 4.1: The tree of dyadic ranges.

constructing empty nodes. Precisely, at the site of the data owner the algorithm constructs a table that maps the tuple-ids (e.g., the employee ids) to all dyadic ranges that contain the corresponding tuple-values (the salaries). This means that each tuple id ends up in $\log m$ dyadic ranges and the total number of dyadic ranges is only $2m - 1$. The table is subsequently encrypted using the SSE scheme.

Notice that, in contrast to the quadratic algorithm, querying now requires retrieving the tuples for more than one dyadic ranges. Since SSE schemes are only focused on single token search (e.g., a single keyword, or a single dyadic range), retrieving more than one *correlated* dyadic ranges that correspond to a single user query adds an extra leakage. In the following, we present two different query policies over the constructed index and formally quantify the introduced leakage of each one.

Minimal dyadic cover. This is the simplest query policy. The query initiator first breaks the query range to the minimal dyadic cover 2.1. Then, a trapdoor is produced for each dyadic range, and all trapdoors are packed together as a complex trapdoor and sent to the SSE index for getting the answers. All answers are returned back to the query initiator and decrypted using the secret key.

In terms of performance, minimal dyadic cover policy requires executing at most $O(\log m)$ distinct queries in the SSE per query. The cost of producing, retrieving and decrypting the answer is $O(r)$, where r is the size of the answer. In terms of security, the fact that for each user query we now need to execute $O(\log m)$ queries introduces new

leakages, beyond the leakages inherited from the SSE scheme, i.e., L_1 and L_2.

To demonstrate this extra leakage intuitively, consider a toy example where min = 1 and max = 8. Range query $[2, 7]$ is the only valid range query in this domain that will be split to 4 dyadic ranges, $[2, 2]$, $[3, 4]$, $[5, 6]$, $[7, 7]$ (cf. also Fig. 4.1). Therefore, the untrusted server can always map any query containing four trapdoors to the range query $[2, 7]$. A different leakage example includes running query $[4, 6]$ and then query $[4, 4]$. In this case the server will be able to find the inclusion of the second query to the first. In fact, subsequently running more queries with overlap can give sufficient information to the untrusted server to discover the whole structure of the tree of dyadic ranges. Notice that sending the trapdoors one by one, or even introducing delays between them, would still not help – eventually the untrusted server would be able to correlate the dyadic ranges belonging in a single complex query, e.g., when no other query was executed in parallel with this query in the SSE index. As such, the linearithmic algorithm with this querying algorithm is not $(L_1, L_2, true)$-**RQ-CKA2** secure. In particular, it cannot reach the adaptive RQ-CKA2 security definition as it lacks secure trapdoors. The recent work of Li et al. [21] faces the exact same problem, and thereby it cannot be considered adaptively secure.

Uniform Range Cover. We are able to reduce the extra leakage by modifying the querying algorithm such that all queries of the same length generate the same number of trapdoors (i.e., they are all answered by the same number of dyadic ranges). This is achieved by integrating in the query execution algorithm a technique originally proposed in a different context (delegatable PRFs) by Kiayias et al. [28]. For each query q of length $|q|$, the algorithm first computes the maximum number of required dyadic ranges for all queries of length $|q|$. Let this number be denoted with $\ell(|q|)$. For example, for range queries of length 2, $\ell(2) = 2$ (the maximum number of dyadic ranges occurs when the query includes any two nodes that are not siblings), whereas for queries of length 4, $\ell(4) = 3$. Then the algorithm breaks q to the minimal dyadic cover. If the number of resulting dyadic ranges is less than $\ell(|q|)$, some of the dyadic ranges are replaced with their children as follows: starting from the right-most dyadic range in the minimal dyadic cover, the range is replaced with its children, until the total number of ranges equals ℓ. Finally, the trapdoors are produced for the ℓ dyadic ranges and sent to the SSE index for answering.

In terms of cost, Uniform Range Cover policy introduces the same complexity (both network and computation as the Minimal Dyadic Cover policy, i.e., $O(\log m)$ distinct queries per range query, and $O(r)$ cost for sending and decrypting the answer. In terms of security, this policy decreases the extra leakage by making the same-length range queries indistinguishable, under the condition that the queries have an empty overlap. The following theorem summarizes the properties of the Uniform Range Cover algorithm.

Theorem 4.3.1. *The Linearithmic algorithm with Uniform Range Cover is* $(L_1, L_2, F, (\bigcup_{i=1}^{k-1} q_i) \cap q_k = \emptyset)$-***RQ-CKA2*** *secure if the utilized SSE scheme is* (L_1, L_2)-***CKA2*** *secure. The complexity of the algorithm is* $O(n \log m)$ *for building the index, where* $m = \max - \min$*, and* $O(\log m + r)$ *for query answering, where* r *denotes the size of the answer.*

Proof sketch: To begin with, we define the F leakage function, as $F(range_x) = |URC(range_x)|$, and the simulator as follows. Given the L_1 leakage, the simulator is able to simulate the encrypted data structures of the RSSE scheme. For each query q, the simulator is given the leakage function F and the L_2 leakage. By using the query pattern (obtained from the L_2 and F leakages), the simulator checks if the required range has appeared before, or if it intersects with any previously queried range. In the first case the simulator outputs the same tokens that were generated before. Otherwise, in the second case the simulator rejects the requested query because the condition is not met. Now if the queried range does not belong to neither of the aforementioned cases, the simulator uses the L_2, F leakages to simulate the token. In particular, it uses the F leakage to correctly simulate the token size, and then it randomly selects $F(range_x)$ tokens. Subsequently, the simulator uses the access pattern found in the L_2 leakage to correctly simulate the result of the requested range query. The final step of the simulator is to store the tokens that corresponded to the specific queried-range, as well as the random selection that was made for each range-token. □

4.4 Single Trapdoor Range Cover algorithm

The Single Trapdoor Range Cover algorithm (STRC for short) further reduces leakage by introducing false positives. The goal of STRC is to enable answering any abritrary range query by retrieving exactly one dyadic range from the SSE index, thereby addressing the L_3 privacy leakage of the linearithmic algorithm. Precisely, STRC augments the SSE

index to include also the ranges of the form $[k2^j + 2^{j-1} + 1, (k+1)2^j + 2^{j-1}]$, for j and k integers and $j \geq 1$. Intuitively, this additional data corresponds to adding in the dyadic tree all nodes that cover the range corresponding to any two cousin nodes with distance 1 (i.e., the nodes that are at the same level and have different parents that are siblings – the red-dotted nodes in Figure 4.2). Similar to the linearithmic algorithm, empty dyadic ranges are not included in the index.

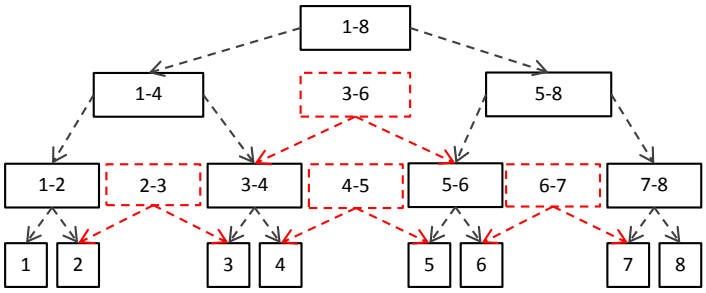


Figure 4.2: The dyadic ranges tree augmented by the STRC algorithm.

This augmented index enables efficient execution of any possible query by retrieving a single query range. In particular, the query initiator always queries for the smallest node from the augmented index that fully covers the query range. For example, if the query is $[2, 4]$, the query initiator will ask for $[1, 4]$, whereas if the query is $[4, 5]$, the query initiator will ask directly for $[4, 5]$. Clearly, this strategy may lead to false positives, i.e., tuples that should normally not be included in the answer will be returned to the user and filtered locally. However, the algorithm guarantees that the number of tuple values that will be wrongly included in the query will be less than or equal to $3|q|$, where $|q|$ denotes the query length.

Notice that, even though the new index enables executing any query with a single trapdoor, which makes the queries indistinguishable, the number of replications of each tuple is not identical. For example, a tuple with tuple-value 1 will be replicated 4 times in the SSE index whereas a tuple with tuple-value 4 will be replicated 6 times. To avoid introducing new leakage during updates (we will discuss more for updates in Section 4.7), we resort to padding. In particular, after constructing the inverted index, the data owner adds random tuples to the index such that the total index size becomes exactly $n(2 \log m - 2)$. These random tuples have a tuple-value that is outside the domain range

of the real data, in order not to pollute possible query results, e.g., tuple-value= max +1. Subsequently, the padded index is encrypted using the SSE scheme, and then outsourced.

The following theorem summarizes the properties of the algorithm.

Theorem 4.4.1. *The STRC algorithm is* $(L_1, L_2, F, true)$-***RQ-CKA2*** *secure if the utilized SSE scheme is* (L_1, L_2)-***CKA2*** *secure. The complexity of the algorithm is* $O(n \log m)$ *for building the index, where* $m = \max - \min$.

Proof sketch: To begin with, we define function F such that $F(range_x) = range_y$, where $range_y \in \Delta^{Linearthmic-STRC}$ (Function $F()$ maps the input range to a range in the dictionary, thus minimizing the false positive values). In addition, we define a simulator that proceeds as follows. Given the L_1 leakage, the simulator is able to simulate the data structures of the RSSE scheme. For each query q, recall that the simulator is given the L_2 and F leakage functions. Firstly the simulator evaluates the requested range using the F leakage function to acquire the mapping of the specific range to $\Delta^{Linearthmic-STRC}$. Then, she uses the query pattern to determine if the query is new, and therefore meets one of the two cases (1) the range has appeared before (2) the range has not appeared before. The simulator uses the L_2 function to determine in which of the aforementioned cases does the specific range query fall into. If it belongs to the first case then it returns the previously generated token, whereas if it belongs to the second case the simulator randomly selects the token. It also needs to store the token that corresponds to a specific queried-range, as well as the random choice that was made for each range-token. □

Notice that, even though the number of wrongly-included keys in the query is at most $O(r)$, the number of false positives in the final answer is bounded only by n. Such a high number of false positives can be due to a pathological scenario caused by extremely skewed distributions of the tuple values (e.g., the salaries). For example, consider a data set where $n-1$ tuples have a tuple value equal to 1, and one tuple has a value of 3. If the range query is $[2, 3]$, then the returned tuples would be the ones within $[1, 4]$, i.e., $n - 1$ false positives. It is straightforward to get stricter (probabilistic) bounds for particular distributions, e.g., uniform, but this is not a generic solution. The interactive algorithm proposed in the next section addresses this limitation, in a way orthogonal to the values distribution.

4.5 Linear - URC

The Linear approach is based on the utilization of DPRFs. The idea is to deploy the DPRF functionality across the dyadic interval nodes. The storage requirements of the Linear approach are $O(n)$, since the actual data are stored once in the leaves of a GGM tree. The search processing and communication costs at the client side are logarithmic in the size of the domain as the URC algorithm is being used. Regarding the proposal of the Linear-URC RSSE scheme, our contribution lies in the fact that we are the first to present how to integrate DPRF with SSE schemes, and we are also the first to prove that the Linear-URC approach is an adaptively secure RSSE scheme.

In general, SSE schemes employ a PRF function in **SSE.Enc()** and the same PRF function with the same key is additionally used in **SSE.Trpdr()**. Our Linear-URC RSSE approach replaces these PRF values with DPRF ones. The construction algorithm receives as inputs the min and max domain values, and then shifts all the tuples so that the minimum domain value becomes 0. Note that it has to store the minimum value in order to correctly respond to range queries. If the max - min value does not equal $2^x - 1$, for some integer $x > 1$, the construction algorithm computes the next power of two, denoted as j. Subsequently, it randomly selects and stores a secret key k. The construction algorithm computes 2^j PRF values by using the DPRF functionality and the secret key k. Furthermore, it associates each of the 2^j values with their corresponding PRF value. For example value 3 is mapped to $f_k(0011) = G_0(G_0(G_1(G_1(k))))$ assuming that the minimum value is 0 and the maximum value is 15.

The SSE.Trpdr() algorithm uses the same secret key as the one used by SSE.Enc(). More specifically, SSE.Trpdr() initially computes the corresponding PRF values of the URC nodes. For each of these nodes, SSE.Trpdr() outputs the respective PRF value, as well as the respective height location in the tree. Then, SSE.Search() utilizes the DPRF functionality to reach the PRF-leaf values of each URC node and returns all the encrypted tuples associated with the corresponding PRF values; nodes at level 0 are PRF-leaf values. For example, let us assume that the client posed the range query 2-5. Then, function SSE.Trpdr(2-5) returns the following three $<$ PRF-value, level $>$ pairs, $< F_k(001), 1 >, < F_k(0100), 0 >, < F_k(0101), 0 >$. The SSE.Search() algorithm only computes the PRF values of the first pair $F_k(0010), F_k(0011)$, as the remaining pairs are

located in level 0 of the tree, i.e. they are PRF-leaf values. Finally, SSE.Search() outputs the tuples associated with the PRF-leaf values.

The idea of integrating the DPRF functionality with the URC query policy and the SSE schemes, in order to configure an RSSE scheme, leads to additional leakages compared to the Linearthmic-URC approach. In particular, the L_1 leakage function increases with the size of the value domain, and thereby renders it possible for the simulator to correctly simulate the dictionary. Additionally, the simulator is enhanced with the L_3 leakage. The extra leakage is quantified as follows:

$$L_3(D, node) = (ACCP_w(node), level(node), id(node))$$

where *node* corresponds to non-DPRF values, $level(node)$ indicates the height of the the DPRF-node, $ACCP_w(range)$ is the ordered sequence of DPRF leafs corresponding to the input-node $(id(w_1), ..., id(w_{|range|}))$.

On the analysis of the L_3 leakage it is worth noting that for each observed node in Linear-URC, the adversary derives the full order of the noticed node's leaves, but not the order of the remaining nodes. This is a substantial downgrading of security, but quantifying the leakage enables us to propose a more efficient RSSE scheme. Note that even though the OPE schemes provide a slightly more efficient solution than Linear-URC, still Linear-URC outperforms the security guarantees of OPE schemes.

The following theorem summarizes the properties of the algorithm.

Theorem 4.5.1. *The Linear (DPRF) algorithm with Uniform Range Cover is* $(L_1, L_2 \cup L_3, F, (\bigcup_{i=1}^{k-1} q_i) \cap q_k = \emptyset)$-***RQ-CKA2*** *secure if the utilized SSE scheme is* (L_1, L_2)-***CKA2*** *secure. The complexity of the algorithm is* $O(n)$ *for building the index, and* $O(\log m + r)$ *for query answering, where* r *denotes the size of the answer.*

Proof sketch: To begin with, we define function F as $F(range_x) = |URC(range_x)|$ and the simulator as follows. Given the L_1 leakage the simulator is able to simulate the encrypted data structures of the RSSE scheme. In particular knowing the domain of values (assuming that it is a power of 2, if not then the simulator computes the next power of the maximum size of the domain), allows the simulator to compute all the nodes of the DPRF tree and associate each input tuple with the corresponding DPRF leaf. Then she deploys the SSE scheme, which is (L_1, L_2)-**CKA2** secure, with a trapdoor computation produced by the DPRF functionality instead of the; Kiayias et al. have proved that a

delegatable PRF is indistinguishable from a PRF. For each query q, the simulator is given the leakage functions F, L_2 and the DPRF oracle. With the use of the query pattern, the simulator checks if the required range has appeared before, or if the required range intersects with any previously queried range. In the first case the simulator outputs the same tokens which were generated before, while in the second case the simulator rejects the requested query because the necessary condition is not met. Now if the queried range does not belong to either of the aforementioned cases, the simulator uses the L_2, L_3, F leakages to simulate the token. In particular, it uses the F leakage to correctly simulate the token size, and then it randomly selects $F(range_x)$ tokens. With the use of the L_3 leakage, the simulator knows the level of each node, which reveals the number of computations required to obtain the DPRF values. Additionally, the L_2 leakage function contains for each node the respective DPRF leafs in a sorted manner, thus allowing the simulator to correctly simulate the result of the requested range query. Finally, the simulator needs to store the tokens that correspond to a specific queried-range, as well as the random choice that was made for each range-token.

4.6 Interactive algorithm

To enable bounding the number of false positives, even in the presense of high skew in the tuple values, we consider an interactive, two-rounds, algorithm. Briefly, the algorithm constructs two indexes using STRC, the *data* index and the *pointers* index. The data index is used for storing the tuples ordered by their tuple values, e.g., the salary, whereas the pointers index contains pointers on the starting and ending positions for each tuple value in the first index.

The data index is constructed as follows. First, the data owner sorts all tuples based on the tuple value, with ties broken at random. All tuples are then indexed by STRC, but now using their position in the ordered list as the value for indexing. Conceptually, the constructed index corresponds to a tree that has a single tuple per leaf, (i.e., a total of n leaves), and the order of leaves (from left to right) is the same as the order of tuples in the ordered list (cf. Fig. 4.3). Notice that, since the placement of the tuples in the data index solely depends on the order of each tuple in the sorted list, the formed dyadic ranges do not correspond to the real tuple values. For example, the dyadic range $[1, 4]$ will contain the four tuples with the lowest tuple values. Their corresponding tuple values

may, or may not be contained in the range $[1, 4]$. As such, the data index alone is not sufficient for executing range queries on the tuple values efficiently.

The pointers index covers this requirement by providing an efficient and secure way to retrieve the starting and ending positions for each of the observed tuple values. Conceptually, the pointers index is an inverted index that maps each non-empty dyadic range to a list of starting and ending positions – one starting and ending position for each distinct tuple value contained in the dyadic range. For instance, since the data index depicted in Fig. 4.3 contains three tuples with tuple value 1 at positions 1 to 3, the corresponding pointers for dyadic range $[1, 1]$ will be 1 and 3 (cf. Fig. 4.4). Similarly, dyadic range $[1, 2]$ will contain the same pointers, since there is no tuple with value 2 in the data index, whereas range $[1, 4]$ will contain both the starting and ending pointers for 1 and for 3, i.e., only for the values that occur in the data set.

Query execution starts from the pointers index, to locate the starting and ending positions of the query range. For example, query $[1, 2]$ will return starting position 1 and ending position 3. Then, the user locates the smallest dyadic range from the data index that fully contains the starting and ending position (i.e., $[1, 4]$), and executes the query. Note that querying each of the indexes requires exactly one query message, thereby making the queries indistinguishable. The query at the data index may return some false positives in addition to the true answers, but the number of false positives is now strictly bounded by $3r$, where r is the size of the correct answer. It is important to note that this worst-case bound is unaffected by the distribution of the tuple values, in contrast to the case of STRC (Section 4.4).

The following theorem summarizes the properties of the Interactive algorithm.

Theorem 4.6.1. *The Interactive algorithm is* $(L_1, L_2, F, true)$-***RQ-CKA2*** *secure if the utilized SSE scheme is* (L_1, L_2)-***CKA2*** *secure. The complexity of the algorithm is* $O(n \log n)$ *for building the index, where* $m = \max - \min$*, and* $O(\log n + r)$ *for query answering, where* r *denotes the size of the answer.*

Proof sketch: To begin with, we define the F function to be identical to the respective one in the non-interactive STRC proof. We prove the Interactive STRC protocol to be secure by using the same security game as the one used in the non-interactive protocol, yet with certain modifications. Firstly, in this case we have two instances of the STRC $(L_1, L_2, true)$-**RQ-CKA2** secure scheme. The first STRC scheme is deployed

for the pointer index to which we refer as index 1, i.e. $\text{RSSE}^1 = (\text{Gen}^1, \text{Enc}^1, \text{Trpdr}^1, \text{Search}^1, \text{Dec}^1)$, while the second STRC scheme is deployed for the data index to which we refer as index 2, i.e. $\text{RSSE}^2 = (\text{Gen}^2, \text{Enc}^2, \text{Trpdr}^2, \text{Search}^2, \text{Dec}^2)$. Furthermore, the interactive security game corresponds to an **RSSE** scheme which includes the following algorithms (**Gen**, **Enc**, **Trpdr**, **Search**, **Dec**). We denote each of these functions with bold writing in order to emphasize on the new functionality introduced by this security scheme. Every time, we call the **Gen** function which subsequently calls both, the function for the pointers index (Gen^1) and the function for the data index (Gen^2); the same action is also performed for the rest of the functions marked with bold. Also D_1 denotes the database which is retrieved after the preprocessing of the input data for the pointers' index and similarly D_2 is the databases associated with the data index.

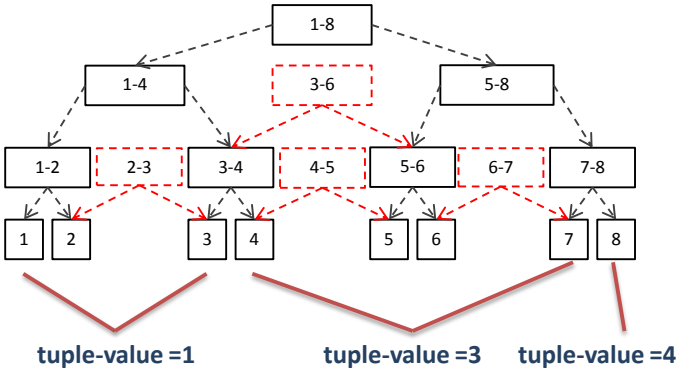
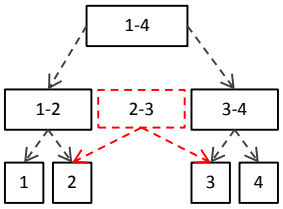


Figure 4.3: Interactive Protocol - Data index

We define the Simulator as follows. Given the $L_1(D_1)$ and $L_1(D_2)$ leakages the simulator is able to simulate the encrypted data structures of the interactive RSSE scheme. For each query q, recall that the simulator is provided with the $L_2(D_1, w)$, $L_2(D_2, w)$ and F leakage functions. We stress out that the adversary outputs the requested range, denoted as $range_q^{point}$, but the simulator also needs the correct range of a specific data index, depicted as $range_q^{data}$. Let us assume that the simulator obtains the additional information of $range_q^{data}$, without introducing extra leakage, by receiving this information as input from the data owner herself i.e. the data owner undertakes the intermediate step i.e. she receives as input the token for $range_q^{point}$ and by having access to the decryption oracle, she decrypts the results of this range and returns $range_q^{data}$ to the simulator. Thus, the simulator possesses for each requested range q, the corresponding range of the



Range	List of <tuple-value : start, end>
1	<1:1,3>
3	<3:4,7>
4	<4:8,8>
1-2	<1:1,3>
2-3	<3:4,7>
3-4	<3:4,7>, <4:8,8>,
1-4	<1:1,3>, <3:4,7>, <4:8,8>

Figure 4.4: Interactive Protocol - Pointers Index

data index. Then, it encounters three cases. The first case (1) is that $range_q^{point}$ has appeared before, the second case (2) is that $range_q^{data}$ has appeared before, and finally case (3) represents the situation where neither $range_q^{point}$ nor $range_q^{data}$ (the case where $range_q^{point}$ has appeared before but $range_q^{data}$ has not, cannot take place in this protocol). In case (1) the simulator uses $L_2(D_1, w)$, $L_2(D_2, w)$ and outputs the same tokens for both $range_q^{point}$ and $range_q^{data}$ that were generated before. In case (2) the simulator randomly selects and stores the token that corresponds to $range_q^{point}$ but returns the same token that was previously generated for $range_q^{data}$. Finally, in case (3) she randomly selects the tokens for $range_q^{pointers}$ and $range_q^{data}$, and in addition she needs to store the tokens that correspond to the specific queried-range, as well as the random choice that was made for each range-token. □

4.7 Handling updates

Even though the discussion up to now focused on index construction for static data sets, the proposed algorithms can be extended to handle updates as well. The challenge is twofold: (a) to enable efficient inserts and deletes and b) to avoid leaking extra information.

Dynamic SSE schemes (DSSE, for short) enable updates, but only by a substantial compromise on both security and performance. In fact, the state-of-the-art DSSE schemes with sublinear update cost (e.g. [35, 38]) cannot achieve the leakages achieved

by static SSE schemes. Furthermore, the most efficient DSSE scheme still suffers from high computational and storage complexity, and is impractical for big data applications. As such, in this work we choose to integrate our algorithms with static SSE schemes, and to enable updates using a technique for dynamization of static data structures of Bentley and Saxe [62]. Even though the following discussion considers the linearithmic algorithm for simplicity, it also applies to STRC and the interactive algorithm.

The main idea is that we organize n sequential updates (both insertions and deletions) to a collection of at most $\log n$ independent encrypted indexes. In detail, for each new tuple, the data owner runs the linearithmic algorithm, with a fresh secret key, to construct a new SSE index that contains only this tuple. The single-tuple index is subsequently uploaded to the untrusted server. Whenever two indexes of the same size s are detected (where s is a power of 2), these are downloaded by the data owner, decrypted, and merged to form a new SSE index of size $2s$, again with a fresh secret key. The new index is then used to replace the two indexes of size s. Clearly, a merging may have a cascading effect, i.e., subsequent merges. In this case, all merges are executed at the same time for avoiding redundant work, i.e., construction and uploading of the intermediary indexes. In terms of querying, all queries are executed to all SSE indexes available at the time of the query, and the results are sent back encrypted to the query initiator. Since the indexes are created using different secret keys (generated on the fly using SSE.KeyGen), the data owner keeps track of these $\log n$ keys locally.

As typical in big data management systems, e.g., hbase, tuples are not explicitly deleted. Instead, we modify the tuple structure to include a *deletion flag*. Deletions are simulated by inserting cancellation tuples, i.e., new tuples with the same tuple-ids and tuple-values, but with the deletion flag set to true. Cancellation tuples are also included in the query results, enabling the query initiator to filter out the deleted tuples. A normal and a cancellation tuple are be fully removed during merging of two indexes, if they end up in the same index.

In terms of cost, it is straightforward to show that the maximum number of concurrent indexes is at most $\log n$, where n is the number of updates. The total space complexity is the sum of the space required by the individual indexes I_k, i.e., $\sum_{k=1}^{\log n} |I_k|$. A standard result guarantees that the total space complexity of all indexes is the same as the space complexity of the corresponding employed algorithm of [62]. The amortized insertion time per tuple increases by a factor of $\log n$ compared to the time of the static algorithm.

Query execution cost is increased by a factor of $O(\log n)$ of the corresponding algorithm, since we now need to query at most $\log n$ different indexes. The number of false positives for algorithms STRC and Interactive stays the same.

Chapter 5

Experimental Evaluation

This section presents the experimental evaluation of our proposed algorithms. Our primary goal is to clearly illustrate the performance of the presented range symmetric encryption schemes.

5.1 Data Sets

The experimental evaluations were performed on two real datasets. The first real dataset was derived from the US Postal Service (USPS) and contains the annual incomes [63] of the company's employees. This dataset comprises 389,032 salary tuples. The specific attribute is a real number which has been transformed to an integer. The second real dataset is the Gowalla [64] dataset, which consists of 6,442,890 users' check-in records from February 2009 to October 2010. We performed encrypted range queries on the time stamp attribute of the second data set. The values corresponding to the time stamp attribute were also transformed to integers with the use of the same encoding as in [21]. Furthermore 5 million records were selected in a random uniform manner from the Gowalla dataset that were subsequently partitioned into 10 fixed sized datasets whose size varies from 0.5 to 5 million records with a scaling factor of 0.5 million records.

5.2 Implementation details

Our experiments were carried out using a single core of a DELL latitude E6520 computer with 8GB RAM and Linux operating system. As we discussed in **??** the proposed schemes

utilize Searchable Encryption schemes as black-boxes. For this reason, we implemented the T-Set algorithm proposed by Cash et al. in [36] in Java. In addition, for both the T-Set implementation and the encryption we used AES-CBC, HMAC-SHA1 and HMAC-SHA-512 (provided by the javax.crypto package). Cash et al. described T-Set as a hash table with B buckets each of size S. The configuration of values B and S depends on the number of the input data N and the space overhead parameter k, such that the overflow probability of any bucket after storing N tuples in this hash table is sufficiently small and constant. Also the total size $B * S$ of the hash table should be $O(N)$ (for further details we refer the reader to the original paper [36]). We conducted all the experiments with space overhead parameter k equal to 1.1, and the S value equal to 6000. Te number of buckets B was determined based on k, S and N so that negligible overflow probability was achieved.

5.3 RSSE schemes

We performed experimental evaluation on the Linearthmic - MDC, the Linearthmic - URC, the non-interactive STRC, the Linear - URC and the interactive STRC approaches, in terms of construction time, as well as index size. We excluded from our experiments the Quadratic RSSE scheme, for reasons that were included in section **??**. In brief, it has prohibitive storage demands and construction time. Note that the Linearthmic-MDC and Linearthmic-URC schemes have the same construction time and storage demands since they only differ in the range query policy. Additionally, as we have discussed in **??** the non-interactive and interactive STRC approaches introduce false positives, so we evaluate the respective performance.

5.4 Construction Time & Index Size

The experimental results showed that constructing an RSSE schemes is linear with the number of data items, both in terms of time and space. Figures 5.1, 5.2 and Table 5.1 show the construction time as well as the respective required storage for each proposed RSSE scheme. We observed that the interactive STRC approach yields the highest cost, both in terms of construction time and space. The interpretation of this remark lies in the fact that interactive STRC builds two non-interactive STRC instances, one for the

RSSE scheme	Constr. Time (sec)	Index size (MB)
Linear - URC	7.276	10.30
Linearthmic-MDC & URC	140.410	195.7
Non-Interact. STRC	283.442	391.4
Interactive STRC	330.030	419.14

Table 5.1: Construction time & Storage (USPS dataset)

data index and one for pointer index. Furthermore, in the case of the Gowalla dataset Figures 5.1, 5.2, we noticed that interactive STRC requires less than 2x time and storage demands compared to the non-interactive STRC, while in the USPS dataset interactive STRC requires almost 1.2x more time and space compared to the non-interactive. This is due to the fact that the salaries in the Gowalla dataset follow an almost uniform distribution, whereas in the USPS dataset the same values are skewed. In addition, the mean ratio of the distinct values to the total records in the Gowalla dataset is 93.81%, whereas in the USPS dataset it is 4.66% . The worst case scenario for interactive STRC is to require double time and storage compared to non-interactive STRC. Note that this happens when the distribution of the range attributes is uniform and the ratio of the unique range values to the total number equals 100%. In order to compare the construction performance of our work with the [21], we have to stress out that the only meaningful comparison that can be performed is between the Linearthmic-MDC RSSE scheme with the approach in [21], as the rest of the schemes are incomparable with [21] since they provide much stronger security guarantees. Hence, we noticed that in the Gowalla dataset case (studied in both works) the constructions of Linearthmic-MDC RSSE outperform those of [21] in terms of time and space, and also Linearthmic-MDC does not introduce false positives as the work of Li. Finally, note that all the proposed RSSE schemes, except from the Linear one, can be fully parallelized in each level of the index tree without affecting the query processing policy, while the same is not applicable in [21].

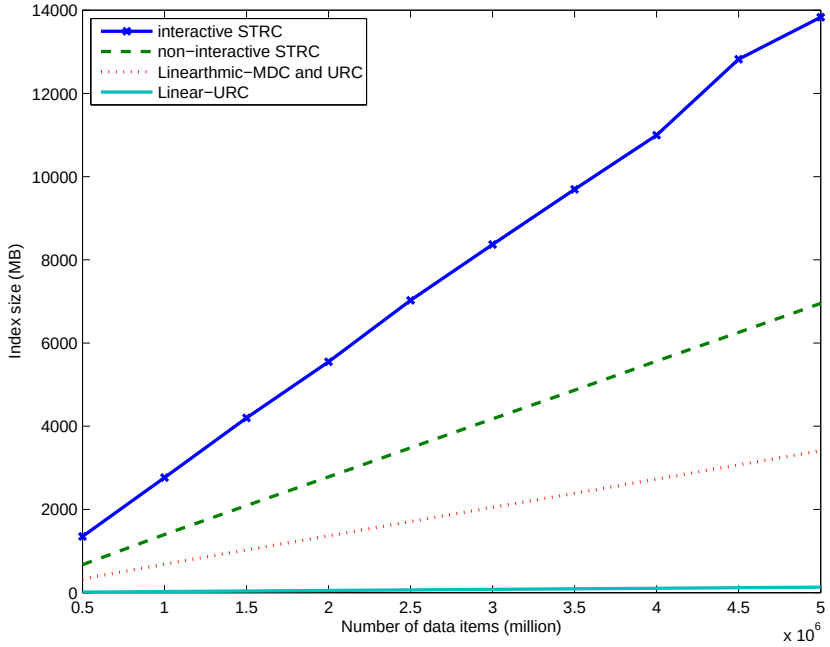


Figure 5.1: Index size - Gowalla Dataset

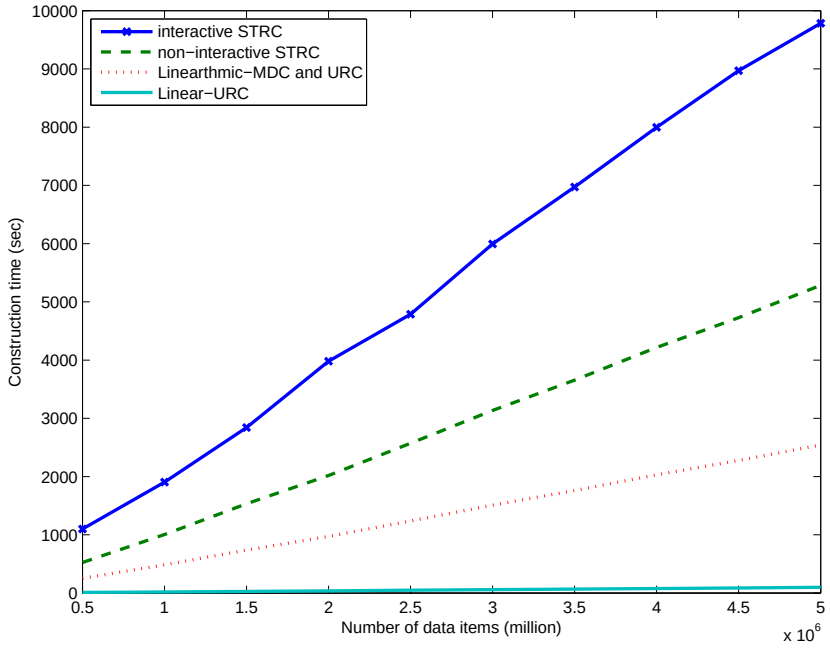


Figure 5.2: Construction time - Gowalla Dataset

5.5 Evaluation of False Positives

We experimentally evaluated the average false positives rate for the non-interactive STRC RSSE scheme and for the interactive STRC RSSE scheme; the rest of the proposed algorithms do no require false positives, and therefore are excluded from the following experiments. In Section **??** we discussed that the number of false positives in the interactive STRC RSSE scheme is bounded for each query by at most 3r, where r is the actual answer. The aforementioned point indicates that applications that need a constant number (linear in the answer size) of false positives for each query, should use STRC. Figures 5.3, 5.4 shows that the average false positives rate for the non-interactive and interactive STRC schemes in both evaluation datasets, changes according to the size of the range query. In particular, the x-axis represents the range size which is determined by the percentage of the domain it includes. Note that in our experiments the queried values follow uniform distribution. We observed that the difference in the average false positive rates produced by the non-interactive and interactive STRC is bigger for the USPS dataset, which contains skewed data, and smaller in the Gowalla dataset. Figures 5.5, 5.6 shows the average false positives ratio for the 10 fixed datasets from Gowalla, while we modify in Figure 5.5 the range size and in Figure 5.6 the query result size. Furthermore, the amortized false positives number for each query in both the non-interactive and interactive schemes is less than 75%, which corresponds to the theoretical worst case false positives bound in interactive-STRC. Finally, note that the average false positives rate for both the evaluated STRC schemes is incomparable with the corresponding false positive rates of the works presented in [21], since the security guarantees in [21] are much weaker and unrealistic for any real application.

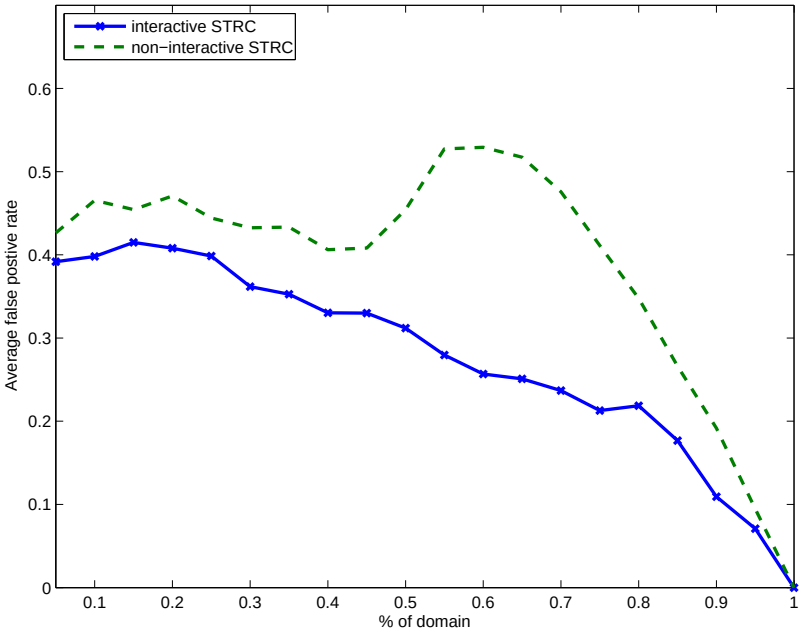


Figure 5.3: Average false positive rate evaluation - Gowalla Dataset

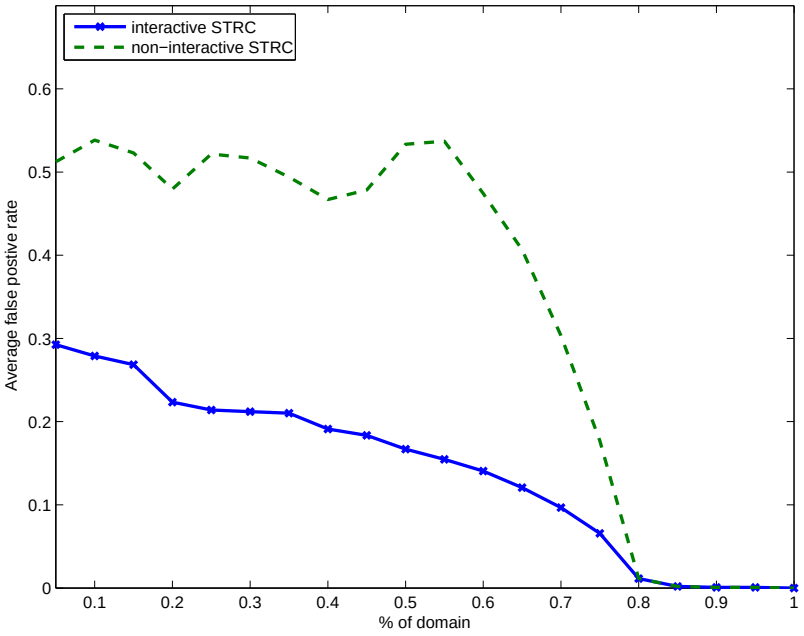


Figure 5.4: Average false positive rate evaluation - Salaries Dataset

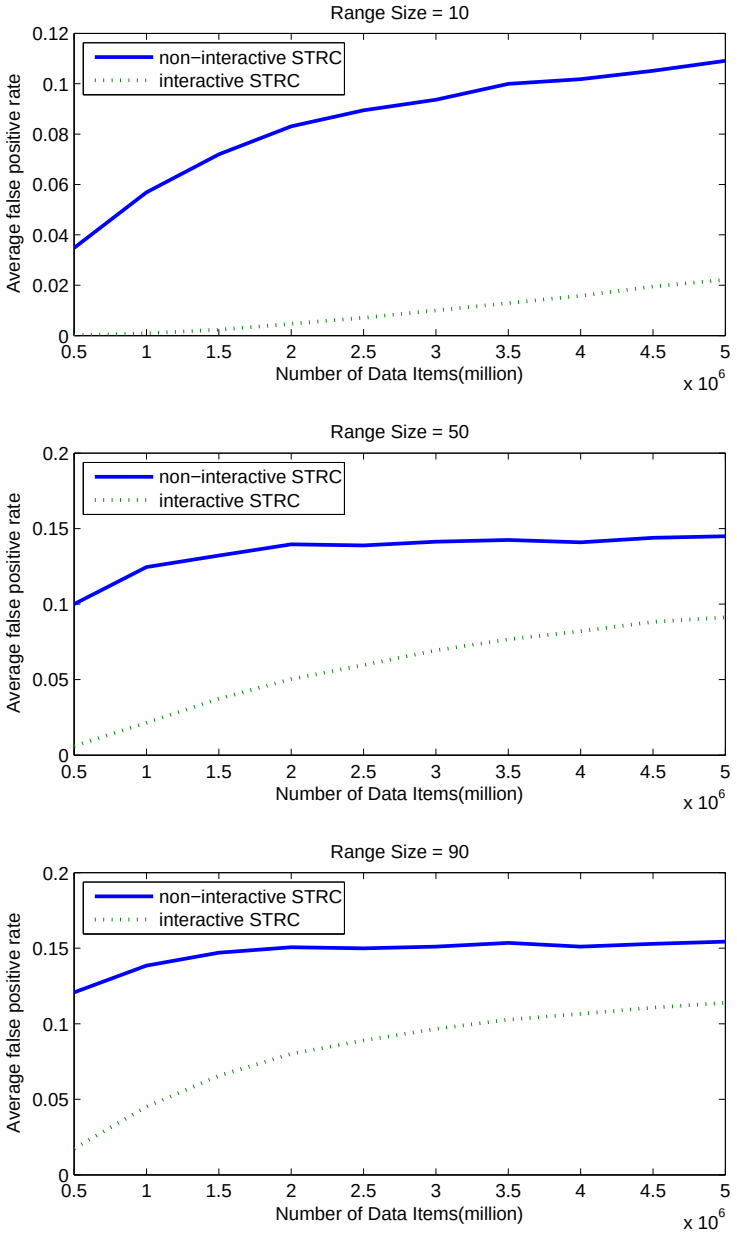


Figure 5.5: Average false positive rate evaluation - Gowalla Dataset

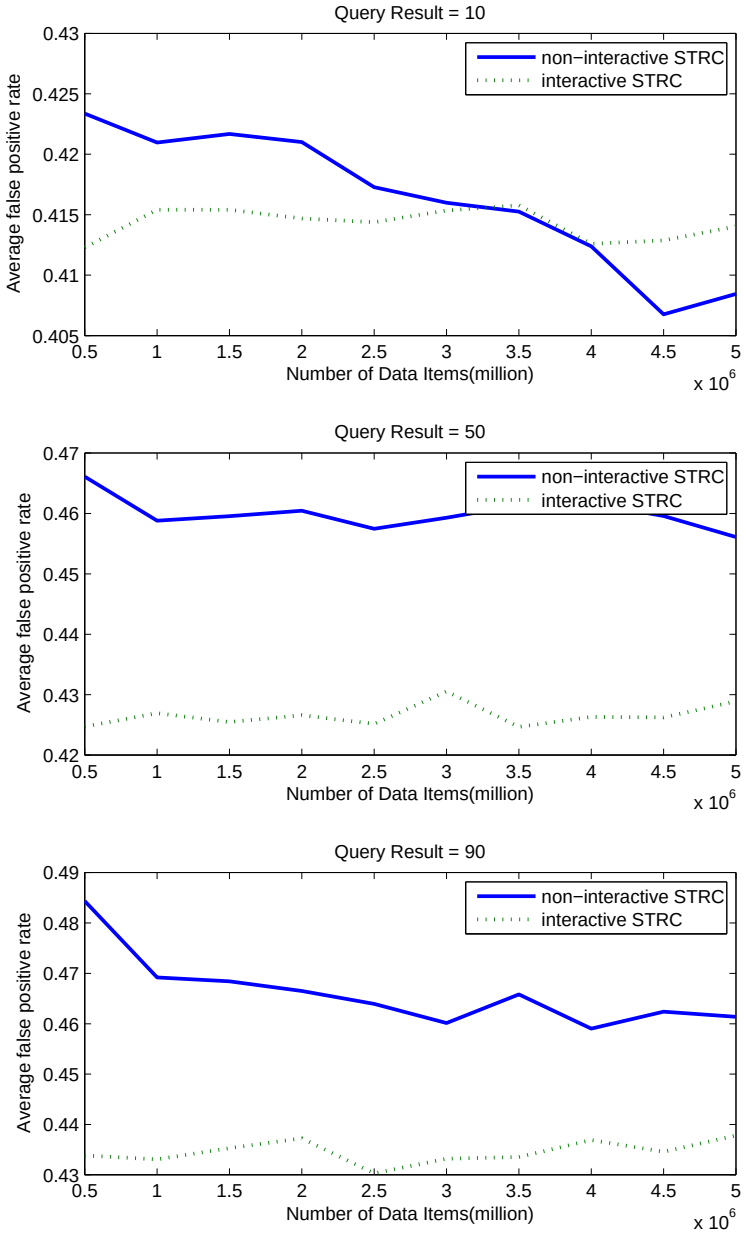


Figure 5.6: Average false positive rate evaluation - Gowalla Dataset

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Conclusively, this work presents the first adaptive Range Searchable Symmetric Encryption schemes by extending the very mature research area of SSE schemes. We provided the first adaptive RSSE security definition, which is used in order to guarantee the security of our own proposed RSSE schemes. Moreover, we showed how to reach this security definition with the use of the Quadratic RSSE scheme which introduced however prohibitive computational and storage demands. Thus, we proposed several other practical RSSE schemes, both in terms of efficiency and security that present security-performance tradesoffs. We described the Interactive STRC scheme which achieves optimal search time and security guarantees by introducing a linear in the answer number of false positives. In addition, we studied a general solution to address updates in our RSSE schemes without introducing additional leakages. We outperform the contemporary related work both in terms of efficiency and security, by proposing the first privacy preserving range query schemes that hide sensitive information regarding the order of the encrypted results and that cannot be attacked by [23, 24] (Quadratic, Linearthmic-URC, the non-interactive and interactive STRC RSSE schemes). Finally, we verified the practical performance of our schemes with the theoretical complexities and showed that our RSSE schemes outperform the closest to our work proposed approaches in the area of RSSE [21].

6.2 Future Work

A significant matter not discussed in this thesis is the case of privacy preserving multi-dimensional range queries. Therefore, developing a generalization of the proposed RSSE approaches can be achieved by extending the Linearthmic, non-interactive and interactive STRC schemes with an additional logn factor for each dimension.

Moreover, the proposed approaches could be integrated with techniques provided by Oblivious RAMs in order to further improve their security guarantees and to completely hide leakages such as query and access patterns which are introduced by the used SSE schemes. Note that Linearthmic-URC, as well as non-interactive and interactive STRC reveal query and access patterns, as they are an extension of SSE schemes. Yet these leakage functions do not include information about the order of the encrypted query result as happens in our other proposed schemes and in similar related works.

Additionally, we have to examine how to further improve the efficiency of our proposed RSSE schemes by assuming the existence of non-colluding servers, which is a relaxation of our proposed security definition.

Finally, throughout the thesis the adversary is designated with specific properties i.e. she is curious but honest. It would be interesting to extend the proposed approach in such a way that malicious adversaries and hostile environments are handled and dealt with, simulating therefore real world conditions. This can be easily achieved by adding message integration techniques (MAC protocols).

References

[1] Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment (2004) 720–731 1, 16

[2] Hore, B., Mehrotra, S., Canim, M., Kantarcioglu, M.: Secure multidimensional range queries over outsourced data. The VLDB Journalb•"The International Journal on Very Large Data Bases **21**(3) (2012) 333–358 1

[3] Wang, P., Ravishankar, C.V.: Secure and efficient range queries on outsourced databases using rp-trees. In: Data Engineering (ICDE), 2013 IEEE 29th International Conference on, IEEE (2013) 314–325 1, 17

[4] Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing sql over encrypted data in the database-service-provider model. In: Proceedings of the 2002 ACM SIGMOD international conference on Management of data, ACM (2002) 216–227 1, 16

[5] Popa, R.A., Redfield, C., Zeldovich, N., Balakrishnan, H.: Cryptdb: protecting confidentiality with encrypted query processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, ACM (2011) 85–100 1, 10

[6] Tu, S., Kaashoek, M.F., Madden, S., Zeldovich, N.: Processing analytical queries over encrypted data. In: Proceedings of the 39th international conference on Very Large Data Bases, VLDB Endowment (2013) 289–300 1, 10

[7] Bajaj, S., Sion, R.: Trusteddb: a trusted hardware based database with privacy and data confidentiality. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, ACM (2011) 205–216 1, 10

[8] Arasu, A., Blanas, S., Eguro, K., Kaushik, R., Kossmann, D., Ramamurthy, R., Venkatesan, R.: Orthogonal security with cipherbase. In: CIDR. (2013) 1, 10

[9] Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In: Advances in Cryptology–CRYPTO 2011. Springer (2011) 578–595 2, 16

[10] Seungmin, L., Donghyeok, L., Taekyong, N., Sehun, K.: Chaotic order preserving encryption for efficient and secure queries on databases. IEICE transactions on information and systems **92**(11) (2009) 2207–2217 2, 16

[11] Xiao, L., Yen, I.L., Huynh, D.T.: Extending order preserving encryption for multi-user systems. IACR Cryptology ePrint Archive **2012** (2012) 192 2, 16

[12] Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. IACR Cryptology ePrint Archive **2013** (2013) 129 2, 16

[13] Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Theory of cryptography. Springer (2007) 535–554 2, 17

[14] Shi, E., Bethencourt, J., Chan, T.H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: Security and Privacy, 2007. SP'07. IEEE Symposium on, IEEE (2007) 350–364 2, 17

[15] Lu, Y.: Privacy-preserving logarithmic-time search on encrypted data in cloud. In: Proc. of NDSS. (2012) 2, 18

[16] Ostrovsky, R.: Efficient computation on oblivious rams. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing, ACM (1990) 514–523 2, 18

[17] Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. Journal of the ACM (JACM) **43**(3) (1996) 431–473 2

[18] Stefanov, E., Shi, E.: Oblivistore: High performance oblivious cloud storage. In: Security and Privacy (SP), 2013 IEEE Symposium on, IEEE (2013) 253–267 2

[19] Stefanov, E., Shi, E., Song, D.: Towards practical oblivious ram. arXiv preprint arXiv:1106.3652 (2011) 2

[20] Stefanov, E., Van Dijk, M., Shi, E., Fletcher, C., Ren, L., Yu, X., Devadas, S.: Path oram: An extremely simple oblivious ram protocol. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, ACM (2013) 299–310 2

[21] Li, R., Liu, A.X., Wang, A.L., Bruhadeshwar, B.: Fast range query processing with strong privacy protection for cloud computing. Proceedings of the VLDB Endowment **7**(14) (2014) 2, 3, 15, 21, 27, 39, 41, 43, 47

[22] Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of the 13th ACM conference on Computer and communications security, ACM (2006) 79–88 3, 11, 12, 14, 22

[23] Dautrich Jr, J.L., Ravishankar, C.V.: Compromising privacy in precise query protocols. In: Proceedings of the 16th International Conference on Extending Database Technology, ACM (2013) 155–166 4, 18, 47

[24] Islam, M.S., Kuzu, M., Kantarcioglu, M.: Inference attack against encrypted range queries on outsourced databases. In: Proceedings of the 4th ACM conference on Data and application security and privacy, ACM (2014) 235–246 4, 47

[25] Rusu, F.I.: Sketches for aggregate estimations over data streams. PhD thesis, University of Florida (2009) 7

[26] Boldyreva, A., Chenette, N., Lee, Y., O Neill, A.: Order-preserving symmetric encryption. In: Advances in Cryptology-EUROCRYPT 2009. Springer (2009) 224–241 8, 16

[27] Katz, J., Lindell, Y.: Introduction to modern cryptography. 2007 8

[28] Kiayias, A., Papadopoulos, S., Triandopoulos, N., Zacharias, T.: Delegatable pseudorandom functions and applications. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, ACM (2013) 669–684 9, 27

[29] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM (JACM) **33**(4) (1986) 792–807 9

[30] Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on, IEEE (2000) 44–55 11, 12

[31] Goh, E.J., et al.: Secure indexes. IACR Cryptology ePrint Archive **2003** (2003) 216 11, 12, 15

[32] Chang, Y.C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Applied Cryptography and Network Security, Springer (2005) 442–455 11

[33] Van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P., Jonker, W.: Computationally efficient searchable symmetric encryption. In: Secure Data Management. Springer (2010) 87–100 11

[34] Chase, M., Kamara, S.: Structured encryption and controlled disclosure. In: Advances in Cryptology-ASIACRYPT 2010. Springer (2010) 577–594 11, 14

[35] Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In: Proceedings of the 2012 ACM conference on Computer and communications security, ACM (2012) 965–976 11, 14, 36

[36] Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.C., Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries. In: Advances in Cryptology–CRYPTO 2013. Springer (2013) 353–373 11, 14, 40

[37] Cash, D., Jaeger, J., Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M., Steiner, M.: Dynamic searchable encryption in very-large databases: Data structures and implementation. (2014) 11, 14

[38] Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. (2014) 11, 36

[39] Bösch, C., Hartel, P., Jonker, W., Peter, A.: A survey of provably secure searchable encryption. ACM Computing Surveys (CSUR) **47**(2) (2014) 18 12

[40] Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions. Journal of Computer Security **19**(5) (2011) 895–934 13, 15, 22

[41] Özsoyoglu, G., Singer, D.A., Chung, S.S.: Anti-tamper databases: Querying encrypted databases. In: DBSec. (2003) 133–146 16

[42] Agrawal, R., Kiernan, J., Srikant, R., Xu, Y.: Order preserving encryption for numeric data. In: Proceedings of the 2004 ACM SIGMOD international conference on Management of data, ACM (2004) 563–574 16

[43] Agrawal, D., El Abbadi, A., Emekci, F., Metwally, A.: Database management as a service: Challenges and opportunities. In: Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on, IEEE (2009) 1709–1716 16

[44] Kadhem, H., Amagasa, T., Kitagawa, H.: Mv-opes: Multivalued-order preserving encryption scheme: A novel scheme for encrypting integer value to many different values. IEICE TRANSACTIONS on Information and Systems **93**(9) (2010) 2520–2533 16

[45] Kadhem, H., Amagasa, T., Kitagawa, H.: A secure and efficient order preserving encryption scheme for relational databases. In: KMIS. (2010) 25–35 16

[46] Xiao, L., Yen, I.L., Huynh, D.: A note for the ideal order-preserving encryption object and generalized order-preserving encryption. IACR Cryptology ePrint Archive **2012** (2012) 350 16

[47] Yum, D.H., Kim, D.S., Kim, J.S., Lee, P.J., Hong, S.J.: Order-preserving encryption for non-uniformly distributed plaintexts. In: Information Security Applications. Springer (2012) 84–97 16

[48] Liu, D., Wang, S.: Programmable order-preserving secure index for encrypted database query. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, IEEE (2012) 502–509 16

[49] Ang, G.W., Woelfel, J.H., Woloszyn, T.P.: System and method of sort-order preserving tokenization (2012) US Patent 20,120,278,897. 16

[50] Liu, D., Wang, S.: Nonlinear order preserving index for encrypted database query in service cloud environments. Concurrency and Computation: Practice and Experience (2013) 16

[51] Kerschbaum, F., Schroepfer, A.: Optimal average-complexity ideal-security order-preserving encryption. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM (2014) 275–286 16

[52] Wong, W.K., Cheung, D.W.l., Kao, B., Mamoulis, N.: Secure knn computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, ACM (2009) 139–152 17

[53] Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009) 18

[54] Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. Journal of the ACM (JACM) **45**(6) (1998) 965–981 18

[55] Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on, IEEE (1997) 364–373 18

[56] Sion, R., Carbunar, B.: On the computational practicality of private information retrieval. In: Proceedings of the Network and Distributed Systems Security Symposium. (2007) 2006–06 18

[57] Olumofin, F., Goldberg, I.: Revisiting the computational practicality of private information retrieval. In: Financial Cryptography and Data Security. Springer (2012) 158–172 18

[58] Williams, P., Sion, R.: Usable private information retrieval. In: Network and Distributed System Security Symposium. (2008) 19

[59] De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Pelosi, G., Samarati, P.: Efficient and private access to outsourced data. In: Distributed Computing Systems (ICDCS), 2011 31st International Conference on, IEEE (2011) 710–719 19

[60] Wang, S., Agrawal, D., El Abbadi, A.: Generalizing pir for practical private retrieval of public data. In: Data and Applications Security and Privacy XXIV. Springer (2010) 1–16 19

[61] Gilbert, A.C., Kotidis, Y., Muthukrishnan, S., Strauss, M.: How to summarize the universe: Dynamic maintenance of quantiles. In: VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China. (2002) 454–465 25

[62] Bentley, J.L., Saxe, J.B.: Decomposable searching problems. (1979) 37

[63] http://www.app.com/ 39

[64] Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM (2011) 1082–1090 39