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

Private Data Analytics in Cloud Computing Environments

Demertzis Ioannis

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

Chania, October 2013

Πολυτεχνειο Κρητης

Σχολη Ηλεκτρονικων Μηχανικων και Μηχανικων Υπολογιστων

Ανάλυση Προσωπικών Δεδομένων σε συστήματα νέφων υπολογιστών

Ιωάννης Δεμερτζής

Εξεταστική Επιτροπή

Καθ. Μίνως Γαροφαλάκης (ΗΜΜΥ)

Επικ. Καθ. Αντώνιος Δεληγιαννάκης (ΗΜΜΥ)

Καθ. Σταύρος Χριστοδουλάκης (ΗΜΜΥ)

Χανιά, Οκτώβριος 2013

Abstract

The term of Cloud Computing is frequently encountered, since it offers a variety of computing features, such as services and resources, that are delivered over the Internet to the service provider's infrastructure. What makes cloud computing appealing is the reduced cost, as well as benefits, such as scalability and elasticity. However, it is noted that cloud computing is preferred by applications that use less sensitive data, because security issues have not been yet completely resolved, with respect to efficiency. Approaches proposed by different scientific fields have tried to eliminate this problem and provide the desirable efficiency and security guarantees. Although improvements have been made to privacy preserving queries, few concern privacy preserving range queries.

This thesis suggests an encryption scheme that allows the execution of range queries on encrypted data in an efficient and secure manner. We assume that the adversary has statistical information about the distribution and the domain of the values, as well as that he is honest but curious. Regarding the efficiency of our approach, we answer range queries at logarithmic computation cost and without revealing the order of the ciphertexts, by using the dyadic intervals technique. Furthermore, our encryption scheme satisfies strong security definitions provided by the Crypto community and resolves complex and practical issues, such as the Query Access Pattern problem and the protection against statistical attacks. To the best of our knowledge this work is the first that simultaneously satisfies the desirable efficiency and privacy guarantees, while dealing with severe and complex issues in the case of privacy preserving range queries.

Περίληψη

Το Cloud Computing είναι ένας όρος που συναντάμε συχνά, καθώς παρέχει μία ποικιλία από υπηρεσίες και πόρους, τα οποία προσφέρονται μέσω του διαδικτύου στις υποδομές των παρόχων υπηρεσιών. Εκείνα που καθιστούν το Cloud Computing δελεαστικό είναι το μειωμένο κόστος, καθώς και οφέλη, όπως η ελαστικότητα και η επεκτασιμότητα. Παρ΄ όλα αυτά, σημειώνεται ότι η συγκεκριμένη τεχνολογία προτιμάται από εφαρμογές που δεν χρησιμοποιούν ευαίσθητα δεδομένα, καθώς δεν έχουν επιλυθεί όλα τα ζητήματα που σχετίζονται με την προστασία των δεδομένων αυτών, λαμβάνοντας υπόψη και την απόδοση. Τεχνικές που έχουν προταθεί από διαφορετικά επιστημονικά πεδία έχουν προσπαθήσει να εξαλείψουν το παραπάνω πρόβλημα και να παρέχουν την επιθυμητή αποτελεσματικότητα και τις απαραίτητες εγγυήσεις ασφάλειας. Αν και έχουν σημειωθεί βελτιώσεις σχετικά με επερωτήσεις σε αποθηκευμένα ιδιωτικά δεδομένα, λίγες από αυτές αφορούν επερωτήσεις εύρους.

Η συγκεκριμένη διπλωματική προτείνει ένα σχέδιο κρυπτογράφησης που επιτρέπει την εκτέλεση επερωτήσεων εύρους σε κρυπτογραφημένα δεδομένα, αποτελεσματικά αλλά ταυτοχρόνως και με ασφάλεια. Υποθέτουμε ότι ο αντίπαλος διαθέτει στατιστικές πληροφορίες που αφορούν την κατανομή και το πεδίο ορισμού των δεδομένων, ενώ ο ίδιος είναι ειλικρινής, αλλά περίεργος. Όσον αφορά την αποτελεσματικότητα της προσέγγισης μας, η απάντηση σε επερωτήσεις εύρους γίνεται σε λογαριθμικό κόστος υπολογισμού και χωρίς την αποκάλυψη της διάταξης των κρυπτογραφημένων δεδομένων, με τη χρήση της τεχνικής των δυαδικών δέντρων. Επιπλέον, το προτεινόμενο σχέδιο κρυπτογράφησης εγγυάται ισχυρούς ορισμούς ασφάλειας που παρέχονται από την κοινότητα της κρυπτολογίας και επιλύει περίπλοκα και ρεαλιστικά ζητήματα, όπως το πρότυπο διάσχισης ερωτημάτων και την προστασία απέναντι σε επιθέσεις που έχουν ως στόχο την διαρροή της κατανομής των δεδομένων. Σύμφωνα με τα όσα γνωρίζουμε έως τώρα, η συγκεκριμένη δουλειά είναι η πρώτη που ταυτόχρονα πετυχαίνει την επιθυμητή αποτελεσματικότητα και εγγυάται την απαιτούμενη ασφάλεια, ενώ αντιμετωπίζει σοβαρά ζητήματα, στην περίπτωση των επερωτήσεων εύρους επάνω σε αποθηκευμένα ιδιωτικά δεδομένα.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor, professor Minos Garofalakis, for initiating me to the world of encryption, supervising and motivating me. I would also like to thank assistant professor Antonios Deligiannakis for his cooperation, support and our fruitful discussions.

I remember the encouragement and the knowledge that professor Stavros Christodoulakis gave me in the area of Databases, and not only, and for that I would like to thank him.

Next, I am deeply grateful and I acknowledge the time that doctor Odysseas Papapetrou spent, in order to guide me throughout this whole process, as well as for his thoughtful and detailed comments.

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 thank my good friend I. Perros, for the hours in the office we spent together that passed so pleasantly, as well as all my friends, M. Alimpertis, F. Abatzi, K. Douzis, N. Kofinas, T. Magounaki, D. Makris,K. Makris, A. Markopoulos, M. Orfanoudakis, D. Paliatsa, N. Pavlakis, A. Soula, E. Soulas and G. Vlachantonis, for all the moments, surprises and experiences that we have shared together.

Contents

List of Figures

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's 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). Recent work on fully homomorphic encryption has allowed the execution of arbitrary computations on encrypted data but with a prohibitively high cost on performance, that makes it impractical. Another more practical approach to the problem suggests the use of Partial Homomorphic Encryption (PHE), in order to be able to perform certain functions. More specifically PHE allows us to sum and multiply the values of plaintexts, while in encrypted form, using Paillier ($Dec(Enc(m1) * Enc(m2) \ mod \ n^2) = m1 + m2 \ mod \ n$) and ElGamal ($Dec(Enc(m1)^{m2} \ mod \ n^2 = m1 * m2 \ mod \ n$) cryptosytems, respectively. However, the above cryptosystems are quite expensive, indicating that we need to start taking into account more practical solutions, such as Nondeterministic Encryption, Deterministic Encryption and finally Order-Preserving-Encryption (OPE). The first provides no functionality over the encrypted data since it produces different ciphertexts for the same plaintext ($a = b \Leftrightarrow Enc(a) <> Enc(b)$) but at the

same time it provides strong security guarantees. The second allows us to check whether 2 plaintexts are equal by checking whether their corresponding ciphertexts are equal ($a = b \Leftrightarrow Enc(a) = Enc(b)$). Finally, the latter allows the execution of order operations on plaintexts, such as range queries and can perform the same operations on ciphertexts in a similar manner ($a <= b \Leftrightarrow Enc(a) <= Enc(b)$). Consequently little change on the existing software is needed, i.e. easier adaptation and better performance is achieved but at the same time the security level is being degraded. Any order-preserving encryption is not secure against tight estimation exposure, if the adversary can guess the domain, knows the distribution of values in that domain, is able to construct a mapping between ordered ciphertexts and plaintexts. The above indicates a weakness of all OPE schemes, a weakness that may reveal order information to an untrusted server. Also, besides the recent work of MIT, mOPE, all previous OPE schemes allowed further information leakage, in addition to order. Hence, it is necessary to construct a cryptographic scheme that will let us execute range queries efficiently without revealing the order of the ciphertexts and without allowing any leakage.

1.1 Thesis Contribution

In this work, we propose an encryption scheme that allows the execution of range queries on encrypted data in an efficient and secure manner. Our approach can answer range queries in logarithmic cost in the dataset size, by using the dyadic intervals technique. Furthermore, our approach satisfies strong security definitions provided by the crypto community, the IND-CPA and IND-DCPA definitions, depending on what is being encrypted. Our encryption scheme also deals with and resolves complex and practical problems, such as the Query Access Patterns issue and the protection against statistical attacks that aim to Distribution leakages. Therefore, it is targeted for real life applications and conditions. Regarding the efficiency of our technique compared to other approaches, such as in Order Preserving Encryption Schemes, it is worth noting that in our case the required efficiency level is reached without revealing the order of the ciphertexts, while in other cases security is relative, depending on how much we want to improve efficiency. Moreover our technique does not assume the existence of trusted hardware in the Cloud Infrastructure, which also strengthens our claim that our encryption scheme can be easily adapted to the real world requirements. Finally, our encryption structure allows the

execution of updates under some assumptions and could form the base for more complex queries at logarithmic cost.

1.2 Thesis Outline

Chapter 2 presents basic notions of encryption, followed by more detailed descriptions about two block cipher schemes, the Cipher Block Chaining Mode and the Cipher Block Counter Mode, that are used in our approach. At the end of this chapter the concept of Dyadic Intervals is analyzed, since it is the structure that we use in order to organize the data in the cloud. Chapter 3 deals with the problem of how to execute privacy preserving range queries. In particular, the related work that studies the above problem is outlined, with the most commonly practiced approaches being the Order Preserving Encryption Approaches, introduced by the crypto community. In Chapter 4 we discuss in detail our approach that focuses on how to answer encrypted range queries in a guaranteed secure manner, without revealing the order of ciphertexts. More specifically, our architecture handles data, that is transformed in a key-value pair form, where both the key and the value are encrypted prior to the storage on the server side. Additionally, the dyadic intervals technique is applied, in order to answer range queries in logarithmic time and a justification for the use of, the DET and RAND encryption schemes in order to guarantee that the encryption of the key and the value respectively are semantically secure. In the same chapter we also face and deal with the Query Access Pattern issue, as well as the Distribution of the values problem, that could threat on overall system security. The first is resolved with the use of $2FN$ servers, while the latter is prevented with the implementation of a first and second level dyadic tree index. Furthermore, we describe how updates are performed in our architecture, given that the order of the encrypted buckets is concealed from the attacker, while less information is revealed compared to other OPE schemes. Finally, in Chapter 5 we conclude the thesis and outline potential directions for future work.

Chapter 2

Background

2.1 Encryption

Cryptography (or cryptology) is the practice and study of techniques for secure communication, in the presence of adversaries. Secure communication leads to both confidentiality (in order to prevent the exposure of transmitted data) and integrity or authenticity (in order to avoid their modification). In this Section, we deal with data confidentiality.

In cryptography, encryption is the process of encoding messages (or information) in such a way, that no one but the authorized parties can understand them (protection against eavesdroppers). In an **encryption scheme**, the message or information is referred as the **plaintext**. By using an encryption algorithm and an encryption key, the initial message is encoded in order to produce a plaintext in an unreadable (by a human or computer) form. This message is referred as the **ciphertext**. Any adversary that obtains the ciphertext should not be able to extract any information about the original message. An authorized party, however, shall be able to decode the ciphertext using a decryption algorithm, that usually requires a secret decryption key, not available to adversaries. The above process is shown in Figure 2.1. For technical reasons, an encryption scheme usually needs a key-generation algorithm, to randomly produce key.

There are two fundamental types of encryption schemes: **symmetric-key** and **public-key** encryption. In symmetric-key encryption, the encryption and decryption keys are **identical**, forcing the contracting parties to agree on a secret key prior to original message exchange. In public-key schemes, the encryption key is publicly available for encryption

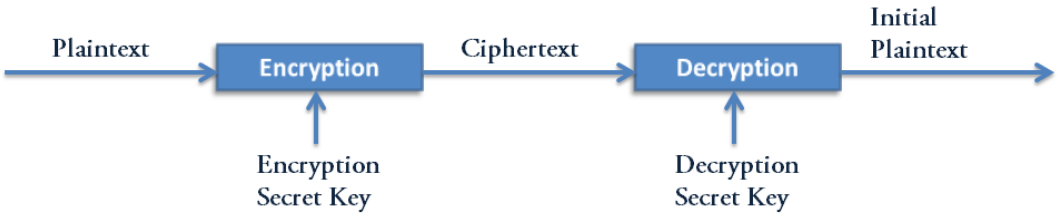


Figure 2.1: Encryption and Decryption

purposes, but only the receiving party has access to the decryption key, which means that only authorized parties can decrypt the encrypted messages.

An encryption scheme is charaterized by $(\mathcal{M}, \mathcal{C}, \mathcal{K}, Enc, Dec)$, where $\mathcal{M}$ are all the possible messages (plaintexts), $\mathcal{C}$ is a set of all possible ciphertexts and $\mathcal{K}$ are all the possible keys, that belong to a keyspace. Enc and Dec are "efficient" functions (algorithms), parametrized by keys where Enc: $\mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$, Dec: $\mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$. Additionally, the algorithms Enc and Dec need to be "efficient", in the sense of running in polynomial time (theoretically), or having concrete time constraints (practical approach).For all possible messages, the following **consistency** equation should be satisfied :

$$\forall m \in \mathcal{M}, \forall k \in \mathcal{K} : Dec(k, Enc(k, m)) = m$$

Usually algorithm Enc is randomized, but Dec is always deterministic.

In order to highlight the differences among the aforementioned encryption schemes, note that the advantages of symmetric algorithms are the concrete security and high speed they provide. Public-key algorithms, on the other hand, need keys of length at least 3,000 bit, in order to achieve the same level of security with a 128-bit symmetric algorithm. Thus, public-key algorithms are incredibly slow and this is impractical for encryption of large amounts of data. Typically, symmetric algorithms are about 1,000 times faster than asymmetric ones.

2.1.1 Nondeterministic and Deterministic Encryption Schemes

Nondeterministic encryption schemes use randomness in the encryption algorithm. More specifically, encrypting the same message several times and under the same key, extracts in general different ciphertexts. From the perspective of databases, we support that nondeterministic encryption schemes cannot provide **any** functionality, since they

preserve strong security levels but do not support any search over the ciphertexts. Still, we can apply them on attributes, but only on those that will not execute selection predicates. The property of Non-Deterministic Encryption Schemes is shown in Figure 2.2. We can observe the different ciphertexts, for the same pair (key,message).

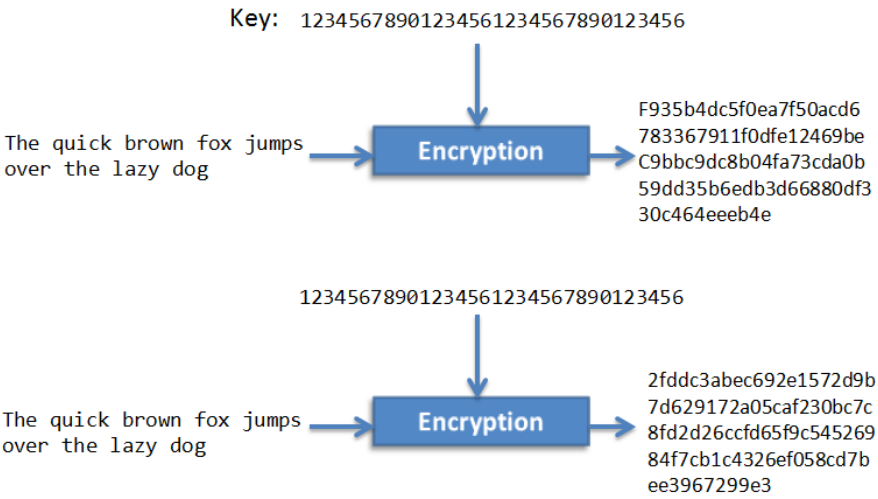


Figure 2.2: Nondeterministic Encryption Scheme(AES + CBC + random IV)

A **deterministic encryption scheme** is an encryption scheme, that always produces the same ciphertext for a given pair of plaintext and key, even over separate executions of the encryption algorithm. We can conclude that in any Deterministic Encryption Scheme the following formula exists:

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

The property of Deterministic Encryption Schemes is shown in Figure 2.3 . It is clear that the same ciphertext is created for the same key, message pair. Consequently, the existence of same ciphertexts implies same plaintexts and therefore the encryption of the same message **leaks information** and leads to severe attacks, especially when the message space $\mathcal{M}$ is small. A solution, in order to resolve the above issue is to enforce the encryptor to never encrypt the same message twice, by choosing messages in a random manner and from a message structure that ensures uniqueness.

A special category of deterministic encryption schemes are the **Order Preserving Encryption (OPE)** schemes, that preserve the order of the plaintexts. More specifically:

$$Enc(sk, m_0) < Enc(sk, m_1), \text{ iff } m_0 < m_1 \text{ for any key } sk$$

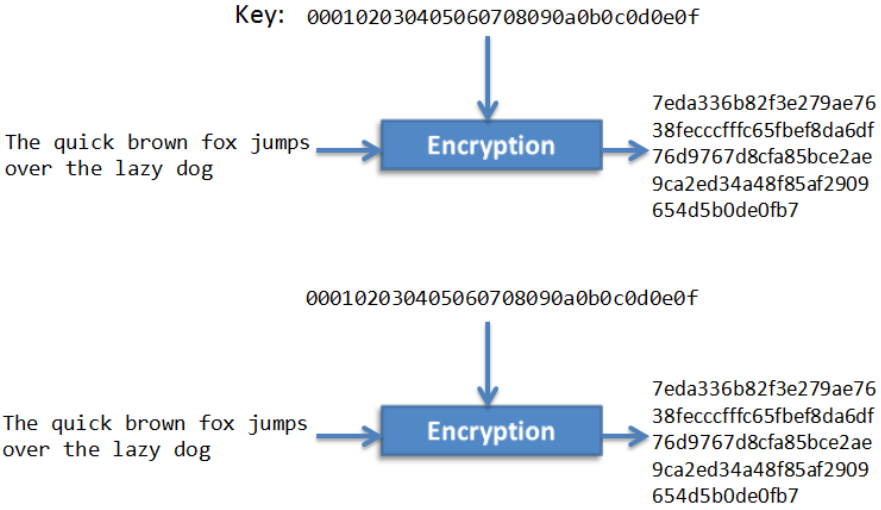


Figure 2.3: Deterministic Encryption Scheme(AES + CBC + constant IV)

2.1.2 Security of Encryption Schemes

In this Section we define the notion of security from the aspect of confidentiality.

The ability to observe ciphertexts determines the attacker's power, while the her goal is to break the encryption scheme and therefore in order to start understanding the term security, it is essential to comprehend when an encryption scheme is insecure or when it is most likely that the attacker will break it. Then, we immediately have to face two security issues:

The attacker should not be able to recover the secret key

or

The attacker should not be able to recover the entire plaintext

Even though the above sentences help us get an idea about the notion of security, still its definition remains inadequate, while the main question that should be answered is: **"What is a secure ciphertext?"**. In order to answer, we present Shannon's perception of security in 2.1.2.1 that will help the understanding of security definitions of Nondeterministic Encryptions schemes in 2.1.2.2, for Deterministic Encryption schemes in 2.1.2.3 and for Order Preserving Encryption schemes 2.1.2.4, proposed by Crypto community.

2.1.2.1 Shannon's definition of Security and Semantic Security

Shannon was the first to believe that *"Ciphertext should not reveal any information about plaintext"* and he introduced the notion of **perfect secrecy**, which is defined below.

Definition 1: A cipher (*Enc*,*Dec*) over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ has **perfect secrecy** if

$$Pr[Enc(k, m_0) = c] = Pr[Enc(k, m_1) = c]$$
$$\forall\ m_0, m_1 \in \mathcal{M}\ ,\ length(m_0) = lentgh(m_1)\ and\ \forall c \in \mathcal{C},$$
where k is uniformly distributed in $\mathcal{K}$

More specifically, even if a ciphertext is given to an attacker, she still cannot distinguish whether the specific ciphertext corresponds to m_0 or m_1 and therefore cannot extract any information about the plaintext corresponding to the ciphertext that is being studied. Consequently, the ciphertext remains secure (only from ciphertext attacks). Unfortunately Shannon also proved the following theorem:

Shannon's Theorem: Perfect secrecy implies that $|\mathcal{K}| \geq |\mathcal{M}|$

Therefore, it is difficult to achieve perfect secrecy in practice, because it assumes that the key space is greater or equal to the message space. In order to give a more practical approach to what a secure cipher is, we restate the first definition to the following:

Let a cipher (Enc,Dec) over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. Then, according to Shannon's perfect secrecy:
Definition 2: (Enc,Dec) has perfect secrecy, if $\forall m_0, m_1\ \in \mathcal{M}\ (|m_0| = |m_1|)$

$$\text{Distribution of } Enc(k, m_0) = \text{Distribution of } Enc(k, m_1)\ \text{,where } k \leftarrow \mathcal{K}$$

Compared to Shannon's initial definition, this definition randomly picks a key k. Then the distribution of ciphertexts, when we encrypt m_0 is exactly the same distribution as if we encrypted m_1. Consequently, even if the adversary observes the ciphertext she still does not know whether it came from the distribution as the result of encrypting m_0 or if it came from the distribution as the result of encrypting m_1 and therefore she cannot tell whether we encrypted m_0 or m_1. The same thing applies for all messages of the

same length and as a final result the attacker does not really know what message was encrypted. The above definition is too **strong**, in the sense that it requires **long keys**. Lets try to weaken the above distribution in the following:

Definition 3: (Enc,Dec) has perfect secrecy, if $\forall m_0, m_1 \in \mathcal{M}$ $(|m_0| = |m_1|)$

Distribution of $Enc(k, m_0) \approx_p$ Distribution of $Enc(k, m_1)$, where $k \leftarrow \mathcal{K}$

The difference between definitions 2 and 3, is that definition 3 supports two computationally indistinguishable distributions, meaning that the attacker cannot distinguish the two distributions, even if they are very different. More specifically, assuming we provide to the attacker a sample from each distribution, she remains incapable of determining the source of each sample. The problem with this definition, is that it remains too strong and therefore we need to add another constraint so that instead of $\forall\ m_0, m_1$, **only** for pairs m_0, m_1, the attacker actually exhibits. That leads us to the following definition of **semantic security**, that considers only one time keys, meaning that we do not use the same key to encrypt multiple messages.

Definition: Semantic Security(one-time key)
$E = (Enc, Dec)$ a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$
For b=0,1 define experiments EXP(0) and EXP(1),
1. Chal. C takes a random key
2. Adv. A output to C m_0,$m_1 \in \mathcal{M}$ so that $|m_0| = |m_1|$
3. Chal. C output to A, $c \leftarrow Enc(k, m_b)$

---

for b=0,1: $W_b := [event\ that\ EXP(b) = 1]$
$Adv_{SS}[A, E] := |Pr[W_0] - Pr[W_1] \in [0, 1]$

E is **semantically secure** if for all "efficient" adversaries A, $Adv_{SS}[A, E]$ is **negligible**

We define for b=0,1 two experiments EXP(0) and EXP(1) as follows: Assume we have an **adversary A**, determined to break the system and a **challenger C**, who receives as input variable b. First, the challenger C gets a random key, while the adversary outputs

two messages, m_0,$m_1 \in \mathcal{M}$ and $|m_0| = |m_1|$, i.e. an explicit pair of messages, on which the attacker wishes to be challenged. Note that the attacker is aware of the length of the messages. Afterwards, the challenger outputs either the encryption of m_0, or the encryption of m_1. So, in the case of b=0, the challenger outputs the encryption of m_0, while in the case of b=1, the challenger outputs the encryption of m_1.

Then, the adversary tries to guess the value of **b**. We define as events W_b, all events that for b=0,1: $W_b :=$ [event that $EXP(b) = 1$]. At this point, we present the advantage of the adversary that is used in the above experiment, which is defined as the semantic security advantage of adversary A against scheme $\mathcal{E}$, to be the difference of the probability of the events W_0 and W_1. More formally:

$$Adv_{SS}[A, E] := |Pr[W_0] - Pr[W_1] \in [0, 1]$$

In order for the encryption scheme E to be semantically secure, the $Adv_{SS}[A, E]$ should be negligible. The notions of non-negligible and negligible follow.

Non-negligible and negligible:

In practice ϵ is scalar and

- ϵ non-negligible : $\epsilon \geq 1/2^{30}$ (likely to happen over 1GB of data)
- ϵ negligible : $\epsilon \leq 1/2^{80}$ (will not happen over life of key)

In theory ϵ is function ϵ:$Z^{\geq 0} \rightarrow R^{\geq 0}$ and

- ϵ non-negligible : $\exists d : \epsilon \geq 1/\lambda^d$ inf. often $(\epsilon \geq 1/poly, for\ many\ \lambda)$
- ϵ negligible : $\forall\ d, \ , \lambda \geq \lambda_d : \epsilon(\lambda) \leq 1/\lambda^d$ $(\epsilon \leq 1/poly,\ for\ large\ \lambda)$

2.1.2.2 Indistinguishable under chosen plaintext attacks (IND-CPA)

We generalize the above definition of semantic security, which considers only one-time keys, in a security definition, which considers many-time keys. With the term *many-time key*, we refer to encryption schemes that use the same key in order to encrypt multiple messages. The fact that a key is used more than once, means that the adversary will observe many ciphertexts, encrypted under the same key. We note that the term **indistinguishable under chosen plaintext attack (IND-CPA)** is equivalent to the term semantically secure under chosen plaintext attacks and they are both used in the bibliography.

The adversary's goal is to break the semantic security. The following definition is a standard semantic security game, similar to the previous one, only now we perform iteration over many queries (over q queries), so that the attacker can issue queries adaptively one after the other in order to simulate the attacker's ability to distinguish multiple ciphertexts encrypted under the same key. Note that in the following security game we select b only once in the beginning of the game and we use the same b for all q queries.

Definition: Semantic Security many-time key (IND-CPA security)
$E = (Enc, Dec)$ a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$
For b=0,1 define experiments EXP(0) and EXP(1),
1. Chal. C takes a random key
for i=1,...,q
2. Adv. A output to C $m_{i,0}$,$m_{i,1} \in \mathcal{M}$ so that $|m_{i,0}| = |m_{i,1}|$
3. Chal. C output to A $c_i \leftarrow Enc(k, m_{i,b})$

---

for b=0,1: $W_b := [event\ that\ EXP(b) = 1]$
$Adv_{CPA}[A, E] := |Pr[W_0] - Pr[W_1] \in [0, 1]$

E is **semantically secure** if for all "efficient" adversaries A, $Adv_{CPA}[A, E]$ is **negligible**

2.1.2.3 Indistinguishable under distinct chosen plaintext attacks (IND-DCPA)

Using the above definition of IND-CPA security and given the same plaintext in an encryption scheme E, which always produces the same ciphertext under a certain key (Deterministic Encryption Scheme), then E **is not CPA secure** with $Adv_{CPA}[A, E] = 1$. This can be easily shown as follows:

The adversary
1. In round q=1 ask for $m_{1,0}$,$m_{1,1}$ so that $m_{1,0} = m_{1,1} = m_*$
Takes back the $c_* = Enc(k, m_*)$
2. In round q=2 ask for $m_{2,0} = m_*$,$m_{2,1} <> m_*$
3. Adversary outputs 0 if $c = c_*$ with $Adv_{CPA}[A, E] = 1$

We can translate the above problem, to terms of the database world. More specifically, the database has duplicates and therefore, under deterministic encryption, these "equal" tuples will be encrypted to the same ciphertext. However, the adversary can observe when two ciphertexts encrypt the same plaintext, which leads to **information leakage**. In order to provide a security definition for Deterministic encryption schemes, transform the IND-CPA security game to the **IND-DCPA (indistinguishable under distinct chosen plaintext attacks)** security game, where the adversary is bound to ask for distinct right and left messages, $m_{i,0}$,$m_{i,1}$ respectively in order to prohibit the attacker from encrypting the same key, message pair twice. Conclusively, the definition of IND-DCPA is weaker compared to IND-CPA, but in the case of encrypting a domain with unique messages the two definitions become equivalent.

Definition: Semantic Security many-time key (IND-DCPA security)
$E = (Enc, Dec)$ a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$
For b=0,1 define experiments EXP(0) and EXP(1),
1. Chal. C takes a random key
for i=1,...,q
2. Adv. A output to C $m_{i,0}$,$m_{i,1} \in \mathcal{M}$ so that $|m_{i,0}| = |m_{i,1}|$
3. Chal. C output to A $c_i \leftarrow Enc(k, m_{i,b})$
where $m_{1,0}, ..., m_{q,0}$ **are distinct and** $m_{1,1}, ..., m_{q,1}$ **are distinct**

---

for b=0,1: $W_b := [event\ that\ EXP(b) = 1]$
$Adv_{DCPA}[A, E] := |Pr[W_0] - Pr[W_1] \in [0, 1]$

E is **sem. secure**, if for all "efficient" adversaries A, $Adv_{DCPA}[A, E]$ is **negligible**

2.1.2.4 Indistinguishable under ordered chosen plaintext attacks (IND-OCPA)

First of all OPE schemes, as all deterministic encryption schemes, cannot be IND-CPA. Secondly, an OPE scheme cannot be IND-DCPA, since by definition IND-DCPA solves the problem of determinism, when in our case an OPE also preserves the order of the

ciphertexts. In order to propose a security definition for all OPE schemes, we have to considerably **weaken** IND-DCPA. The **strongest** definition for OPE schemes is called **IND-OCPA (indistinguishable under ordered chosen plaintext attacks)**, as mentioned in [1]. IND-OCPA aims to hide all information about the plaintext values, except from the ciphertext order, which is the minimum requirement that should be satisfied in order to guarantee the order-preserving property. The IND-OCPA definition, is similar to the previous ones, except that now we enforce additional constraints on the adversary's queries (relaxing IND-DCPA). In [1] and in [2] a proof for the following Theorem is provided.

Theorem: Any OPE scheme, that is IND-OCPA secure has ciphertext size exponential in the plaintext size.

The above indicates that OPE schemes, satisfying IND-OCPA under the **strongest** definition, are impractical. IND-OCPA's definition is shown below.

Definition: Semantic Security many-time key for OPE scheme (IND-OCPA security)

$E = (Enc, Dec)$ a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$

For b=0,1 define experiments EXP(0) and EXP(1),

1. Chal. C takes a random key

for i=1,...,q

2. Adv. A output to C $m_{i,0}$,$m_{i,1} \in \mathcal{M}$ so that $|m_{i,0}| = |m_{i,1}|$
3. Chal. C output to A $c_i \leftarrow Enc(k, m_{i,b})$

where $m_{1,0}, ..., m_{q,0}$ are distinct and $m_{1,1}, ..., m_{q,1}$ are distinct

and $m_{l,0} < m_{j,0} \ \leftrightarrow \ m_{l,1} < m_{j,1}$ for $1 \leq l, j \leq, q$

---

for b=0,1: $W_b := [event\ that\ EXP(b) = 1]$

$Adv_{DCPA}[A, E] := |Pr[W_0] - Pr[W_1] \in [0, 1]$

E is **semantically secure**, if for all "efficient" adversaries A, $Adv_{OCPA}[A, E]$ is **negligible**

Thus, IND-OCPA is too strong a definition to be achieved, with respect to efficiency. In order to create practical encryption schemes, a weaker security definition is described in

[3]. By the term weaker, we imply that we try to weaken the definition of IND-OCPA and as a result we get the definition of **IND-OLCPA (indistinguishable under ordered and local chosen plaintext attack)**. The IND-OLCPA security definition suggests that the adversary learns the encryption only for nearby values, but still it is difficult to enforce such a property in a practical system. An alternative definition of security is proposed by Boldyreva in [1] [4], where the notion of **random order-preserving function (ROPF)** is defined. It has been shown that, apart from the order of the ciphertexts, the security definition of ROMF reveals other information too. More specifically, it leaks at least half of the plaintext bits. The above is mentioned in [4] [5]. There are other security definitions too, only they are significantly weaker and inapplicable in practice, compared to what we have seen above. Consequently, we assume that the adversary attacks will be more restricted and therefore there will be no further reference to them.

2.1.3 Block ciphers

A block cipher is a method for encrypting plaintexts. More specifically, an algorithm and a cryptographic key are applied on a block of data. Block ciphers may either be deterministic or non-deterministic, depending on whether the Initialization Vector (IV) is constant or random respectively. IV corresponds to a unique nonce, meaning that the (key,nonce) pair is used for a unique message. The method of Block ciphers implies, that we first partition a message into blocks of equal size and then carry out an encryption on each block. A ciphertext is formed, after concatenating the IV, with the encoded blocks. At this point, we will present two methods for creating block ciphers, **Cipher Block Chaining mode(CBC-mode)** and **Counter mode (CTR-mode)**. Afterwards, we will mention theorems derived from their IND-CPA analysis.

Cipher Block Chaining (CBC)

In CBC, each block of the plainetext is XORed with the previous ciphertext block before being encrypted. More specifically, assume that the above procedure has just started. Then, the first block is XORed with a key and the **Initialization Vector**. In Figure 2.4 and 2.5 we present the encryption and the decryption circuits respectively. At this point we are going to describe the encryption procedure, which uses the same **encryption algorithm** on each block. At first, a XOR function is executed between the first block and the IV. The result is encrypted under a key, producing the first ciphertext. Then,

the ciphertext that was derived from the first block, is XORed with the second block and again the result is encrypted under a key, producing the second ciphertext. The above procedure is repeated, until the final block is XORed with its previous ciphertext and encrypted under a key.

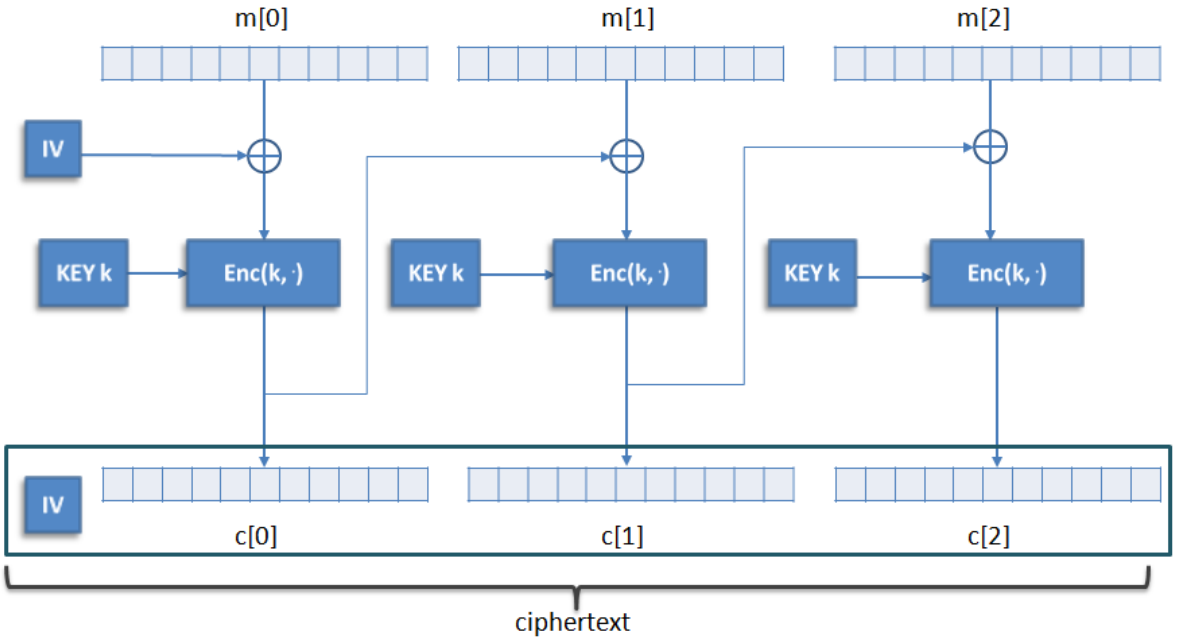


Figure 2.4: Cipher Block Chaining (CBC) mode encryption

The reverse procedure is implemented, in order to achieve decryption and the same **decryption algorithm** is implemented on all encrypted blocks. First, we perform decryption on the first encrypted block (ciphertext), which is followed by a XOR function between the product of the decryption and the IV. The result constitutes the plaintext of the first block. Then, we decrypt the second block under the same key and we execute the XOR function between the decrypted second block and the plaintext of the first block, thereby getting the plaintext of the second block. Similarly to the encoding procedure, the above process is repeated until we get the plaintext that corresponds to the last ciphertext.

Pseudo Random Permutation(PRP) is a function that receives as input a key and a message and produces a permutation of the message. Both the output of the pseudo random permutation function and the message have the same domain. Furthermore, both

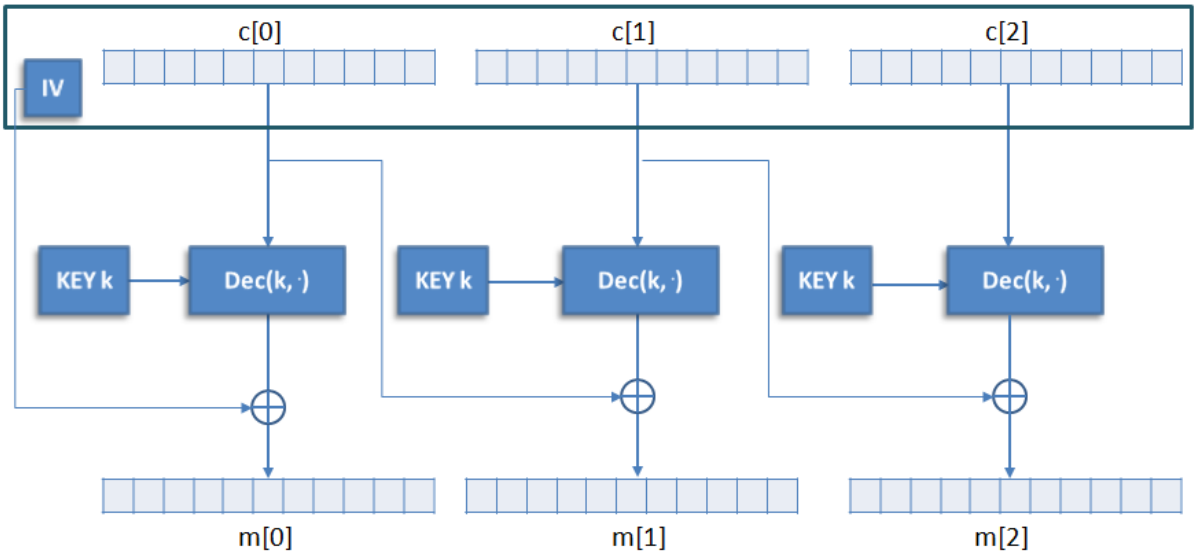


Figure 2.5: Cipher Block Chaining (CBC) mode decryption

PRP and its inverse function should be efficient. Also, PRP is secure if all efficient adversaries cannot distinguish PRP from the Truly Random Permutation with non-negligible advantage, as mentioned in [6].

Note that *Enc* and *Dec* are secure PRPs. The encryption algorithm, as any PRP algorithm, consists of: Enc: $\mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$. Decryption is also PRP since it is the inverse algorithm of encoding.

In order to present the CPA analysis of CBC, we describe the following theorem:
CBC Theorem: For any $L > 0$, if E is a secure PRP over $(\mathcal{K}, \mathcal{X})$, then
The CBC encryption scheme is semantically secure under CPA over $(\mathcal{K}, \mathcal{X}^L, \mathcal{X}^{L+1})$.
In particular, assume that we have a q-query adversary A attacking a CBC encryption scheme E_{CBC}. Then, there exists a PRP adversary B s.t.

$$Adv_{CPA}[A, E_{CBC}] \leq 2Adv_{PRP}[B, E] + 2q^2L^2/|X|$$

which implies that CBC **is only** secure as long as $q^2L^2 << |X|$, where q is the number of encrypted messages with a specific secret key, and L is the length of the biggest message.
Counter mode (CTR-mode) CTR-mode is a block cipher method, that uses secure Pseudo Random Functions (PRFs). PRFs differs from PRPs, because they receive as input a key and a message and produce an output that may not have the same domain

with the message. A PRF is secure if all efficient adversaries cannot distinguish PRF from the Truly Random Function as mentioned in [6]. Unlike CBC, CTR is parallelizable in both, the encryption and the decryption phases. The use of PRFs does not require a different Decryption function, since a PRF with same k and IV values is applied. In this method the IV is split in half. More specifically, the first half corresponds to a nonce, while the latter corresponds to a counter. In Figure 2.6 and Figure 2.7 we note the encryption and the decryption circuits respectively.

With the implementation of PRF, some changes occur in the encryption and decryption algorithms. Regarding the **encryption algorithm** we divide the plaintext into constant blocks and the XOR function is not executed between blocks and ciphertexts, but between blocks and the output of PRF. Now, the IV and the key consist the inputs of PRF. Furthermore, the counter of IV is increased by one every time a block is encrypted, e.g. the counter of IV in the first block is 0, in the second is 1, in i-th is i+1.

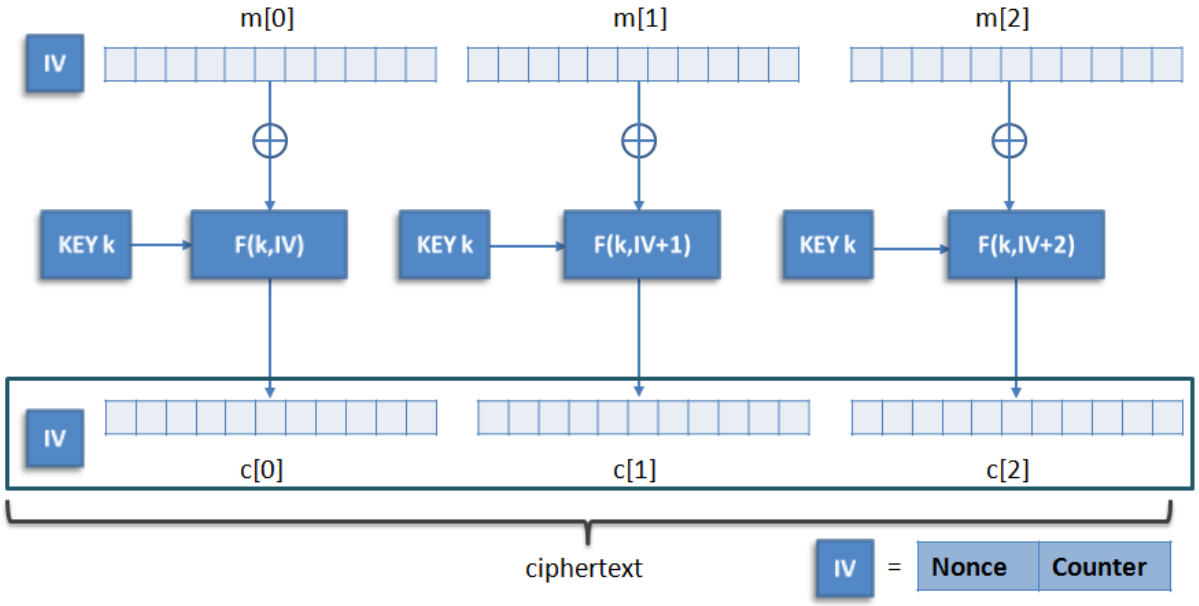


Figure 2.6: Counter (CTR) mode encryption

Similarly to the encryption algorithm, the **decryption algorithm** uses the XOR function between the encoded blocks (ciphertexts) and the PRF output. In this case, the inputs of PRF are the key that was also used in the encryption algorithm and the IV, whose counter represents the number of blocks. In order to present the CPA analysis of

CTR, we need to describe the following theorem:
CTR-mode Theorem: For any $L > 0$, if F is a secure PRF over $(\mathcal{K}, \mathcal{X}, \mathcal{X})$, then CTR-mode encryption scheme is semantically secure under CPA over $(\mathcal{K}, \mathcal{X}^L, \mathcal{X}^{L+1})$. More specifically, assume that we have a q-query adversary A attacking a CTR encryption scheme E_{CTR}. Then, a PRF adversary B exists s.t.

$$Adv_{CPA}[A, E_{CTR}] \leq 2Adv_{PRF}[B, F] + 2q^2L/|X|$$

At this point we must notice that CTR-mode **is only** secure as long as $q^2L << |X|$, where q is the number of encrypted messages with a specific secret key, and L is the length of the max message.

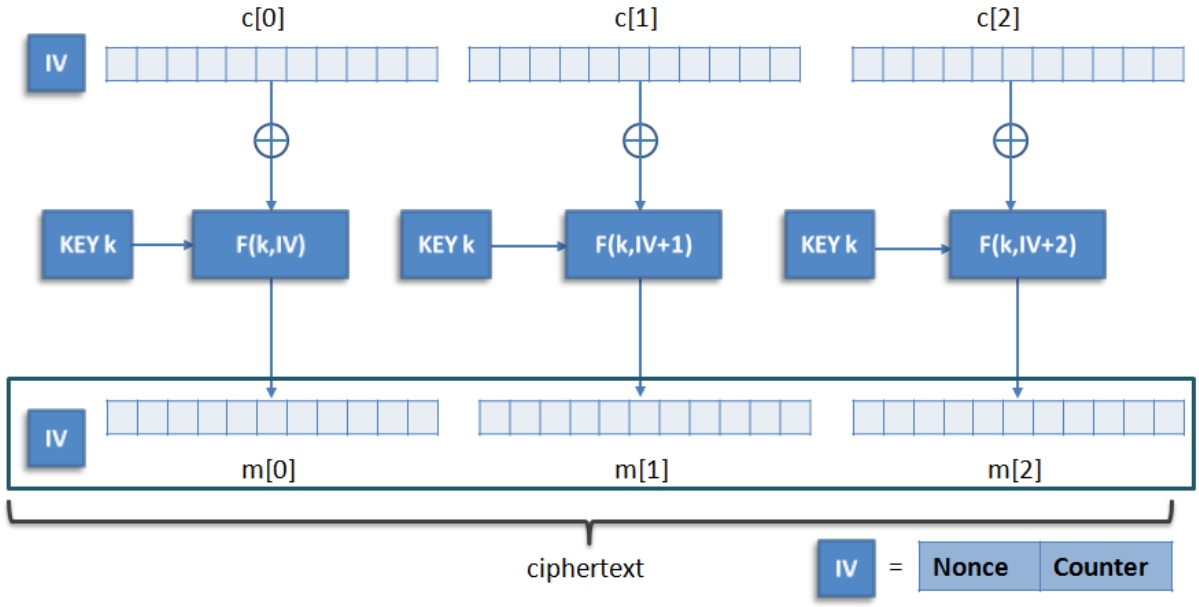


Figure 2.7: Counter (CTR) mode decryption

2.2 Adversary Models

In the previous Section we presented security definitions, proposed by the crypto community. In the same Section we considered that the adversary was only capable of obtaining ciphertexts. Now, we differentiate the adversary into two categories. The first includes those who can be considered as third party attackers, interested in attacking the Cloud

and the latter comprises only the server, where the sensitive data are stored (**untrusted server**).

Passive or curious Adversary but not malicious: The attacker in this model is able to obtain and derive data and queries; she has the 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, comparing 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.

Both of the adversaries in their attempt to extract information about the encrypted data, may know the type of the plaintext values that are being processed and stored (domain). Otherwise, some of them could also be able to guess this domain or even figure out statistical information regarding the values, the distribution of the values, the domain etc.

2.3 Dyadic Intervals

Dyadic intervals of a specific domain are intervals arranged in a certain hierarchical structure, that have some useful properties. First, the length of a dyadic interval is always equal to an integer power of two. Furthermore, a dyadic interval is included in one and only parent node and comprises two children nodes that are also dyadic intervals, but with half length. These properties are not satisfied in case the dyadic intervals correspond to single points or when they have the same starting and ending point e.g. "1-1"="1". The structure of a **dyadic tree** is shown in Figure 2.8. Also, a formal definition of dyadic intervals follows:

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

Property 1: Dyadic tree level j includes exactly 2^j dyadic intervals with each containing 2^{n-j} points from the domain.

Property 2: Dyadic tree level j, $0 \leq j \leq n$, contains dyadic intervals that consist a partition of the domain. More specifically, these intervals are disjoint and their union

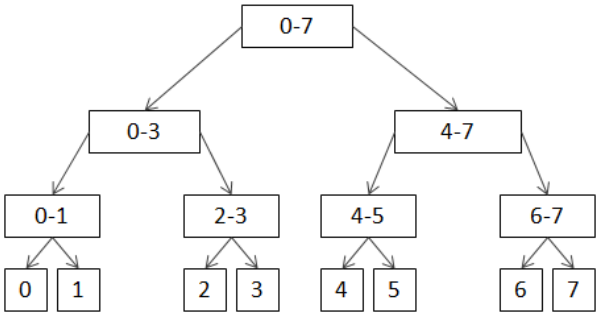


Figure 2.8: Dyadic Interval tree

equals the entire domain.

Property 3: Consider distinct and arbitrary dyadic intervals, δ_1 and δ_2. In case $\delta_1 \cap \delta_2 \neq \emptyset$, then either $\delta_1 \subset \delta_2$ or $\delta_2 \subset \delta_1$.

At this point we will describe how is it possible to express any arbitrary interval as the union of dyadic intervals; note that we do not have a unique decomposition of ranges in dyadic intervals. We introduce the notion of minimal dyadic cover as the minimal decomposition depending on the number of elements. More formally we define the minimal dyadic cover as:

Definition Minimal dyadic cover ($D([\alpha, \beta])$): 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 $\delta_1 \cup \delta_2 \cup ... \cup \delta_m = [\alpha, \beta]$.

Property 4: The minimal dyadic cover $D([\alpha, \beta])$ contains at most 2j dyadic intervals, from which at most two are from the same level.

In order to understand the notion of minimal dyadic cover we give the following example. For the dyadic tree which is represented in Figure 2.8 and the range [1,6] the minimal dyadic cover $D[1, 6] = [1, 2) \cup [2, 4) \cup [4, 6) \cup [6, 7)$ (with different notation $D[1, 6] = \{1\} \cup \{2 - 3\} \cup \{4 - 5\} \cup \{6\}$)

The **complexity** of the Minimal Dyadic Cover algorithm is equal to $O(Log_2(\beta - \alpha))$ i.e. logarithmic in the size of the range. More properties as well as definitions are presented in [7].

Chapter 3

Problem Statement and Related Work

In this Section, we introduce the concept and describe the significance of privacy preserving querying on cloud infrastructures. More specifically, we focus on the main issue that these queries face, which is encountered in the case of range queries. Furthermore, we present in Section 3.2 different approaches that implement privacy preserving range queries, that were summarized in recent 2013 turorial at ICDE [8].

3.1 Privacy Preserving Range Queries

3.1.1 Identifying the problem

In the last few years, an increasing usage of cloud computing has been observed; this trend has become recently more attractive, due to the fact that cloud infrastructures offer luring features concerning computing and storage abilities, while they also offer more essential features, such as pay per use, scalability and elasticity. However, two significant issues that prevent cloud computing from easily being adopted and spread, are security and privacy. Therefore, applications with low risk and less sensitive data constitute the early adopters. Cloud computing is also attractive, due to its ability to allow ubiquitous access on consolidated data and the sharing of infrastructures that reduces the overall cost. Nevertheless, the crucial question is whether cloud computing can guarantee at this point safety and security to the enterprises that will take advantage of it. The answer to

this question is the reason for which DBMS that manage crucial and sensitive data are not entirely moving to cloud technology yet. Cloud infrastructures tend to be tempting attack targets, because with a single attack on a service provider, many businesses and companies are threatened, leading to the disclosure of sensitive data whose exploitation could result to big profits. According to a study of Gartner's statistics in Figure 3.1, it is clear that there is a significant concern regarding the adaptation of cloud computing, in terms of security and privacy, indicating the imperative need to resolve these matters.

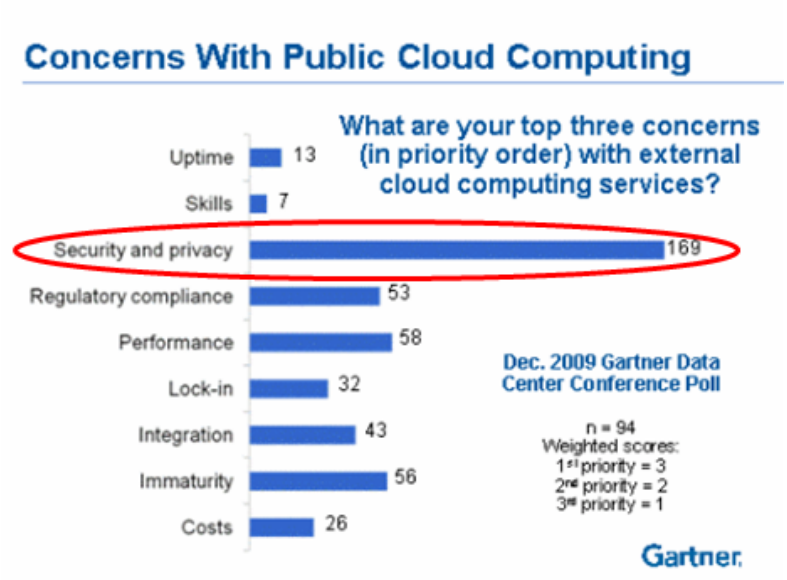


Figure 3.1: Gartner's statistical study on concerns about the use of a Public Cloud

In an attempt to find a solution to the security and privacy issues, two directions have been explored. The first is proposed by the Database Community and provides a practical solution that offers the same functionality with Database Management Systems in terms of efficiency. However, in order to achieve the desirable functionality and performance they sacrifice levels of security. The second direction comes from the Crypto Community. In this case, the proposed approaches achieve the required security but they limit the supported functionality and marginalize the efficiency matter, since many of the recommended solutions are computationally impractical. As shown in Figure 3.2, a solution to these major issues can potentially be achieved with the integration of these different techniques and algorithms proposed by the two fields.

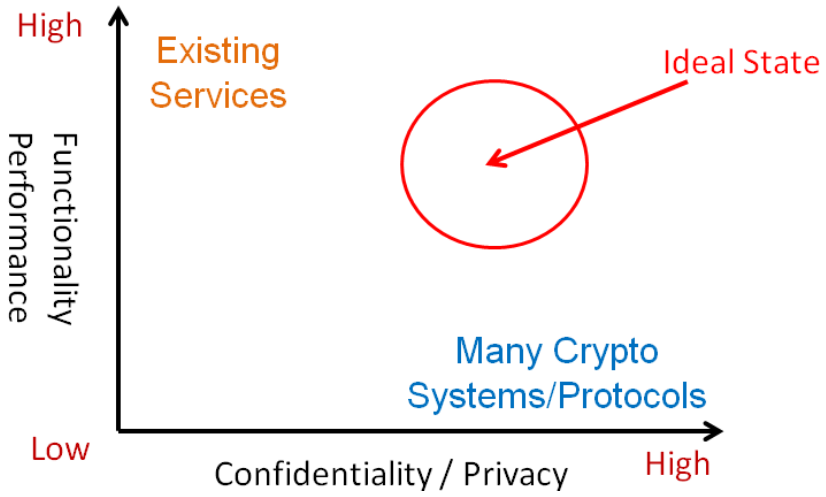


Figure 3.2: Trade-off between functionality-performance and confidentiality-privacy

In this effort to integrate approaches recommended by the Crypto and Database communities a solution has been introduced that attempts to resolve the problem of privacy preserving querying in Cloud infrastructures; CryptDB [9] presents an integrated DBMS with almost all the functionality of MySQL. More specifically, a wide range of SQL queries is supported, but there is also a significant drawback, with respect to the efficient and at the same time secure execution of range queries. CryptDB uses an OPE scheme proposed by Boldyreva in [4], that satisfies weak security definitions and also leaks additional information. Thus, the system proposed by CryptDB is completely insecure and vulnerable to severe attacks.

Another technique that attempts the integration of different algorithms is MONOMI [10], that utilizes the CryptDB approach with an enhanced efficiency system, but it still exhibits the same security matters with CryptDB.

Conclusively, the most essential problem concerning the privacy preserving querying on cloud infrastructures, is finding a solution that is efficient and simultaneously secure in the case of range queries. The 2013 ICDE tutorial also reaches the same conclusion, and therefore it poses the Privacy Preserving Range Querying as an open research subject.

Many approaches from different fields have attempted to resolve the problem of range queries, under the requirements of security and privacy. The most practiced ones are proposed by the Order Preserving Encryption family of approaches, introduced by the Crypto community. Other techniques from the Database field use the notion of buck-

etization, while others focus on the way that the data is distributed and less on the encryption used. In the following Section we briefly review.

3.1.2 An ideal Privacy Preserving Range Query Approach

In order to contribute to the effort of executing Privacy Preserving Range Queries under certain demands, we suggest requirements that could be essential for an **ideal privacy preserving range querying approach**. More specifically an optimal approach requires that:

- All queries are executed in logarithmic time
- Updates are executed efficiently
- The work assigned on the client side is as low as possible, or even negligible
- The concealment of the order of ciphertexts from the untrusted server
- Problems, such as the Query Access Pattern problem and the leak of statistical information, are resolved
- Strong security crypto definitions are achieved
- Efficiency and security requirements are satisfied simultaneously

To the best of our knowlege, **none** of the existing approaches meets the requirements of the **ideal** solution.

3.2 Related Work

3.2.1 Homomorphic and Fully Homomorphic Encryption

Homomorphic encryptions allow mathematical operations to be performed on encrypted data without compromising the encryption. Paillier's homomorphic cryptosystem allows the execution of addition directly on the ciphertexts as shown in Figure 3.3. ElGamal's cryptosystem is a similar cryptosystem that allows direct multiplication of the ciphertexts. It is shown that homomorphic encryption solutions combine the desirable security

level with a high but not impractical cost. However, in the case of range queries no homomorphic based encryption solution has been proposed.

Figure 3.3: Paillier's cryptosystem

Fully homomorphic encryptions are a generalization of homomorphic encryptions. However, unlike homorphic encryptions, fully homomorphic encryptions allow the execution of **any arbitrary** function, including range queries, on the encrypted ciphertexts, without requiring any decryption as shown in Figure 3.4. In 2009, Craig Gentry proposed the first fully homomorphic encryption scheme in his PhD thesis "A Fully Homomorphic Encryption Scheme" [11]. A more formal definition of fully homomorphic encryption is given in [11] and in [12]. Fully homomorphic encryption may seem like a solution to all problems but the essential question remains as to whether it can achieve the desired security level and with what efficiency. Fully homomorphic encryption appears to be **totally impractical**, due to the required ciphertext size and the necessary computational time, that increases sharply as we increase the security level.

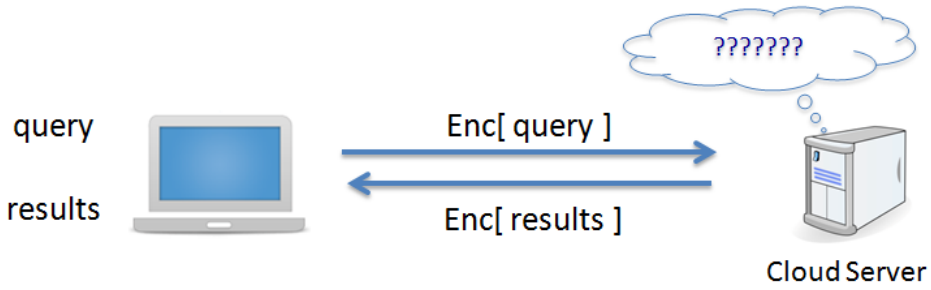


Figure 3.4: Fully Homomorphic Encryption Model

Conclusively neither homomorphic cryptosystems can support Privacy Preserving Range Queries, since an appropriate range query homomorphic cryptosystem does not exist, nor the fully homorphic cryptosystems can, since they are completely impractical.

3.2.2 Order Preserving Encryption Approaches

As we already mentioned in Chapter 2 Order Preserving Encryption (OPE) schemes are deterministic schemes that preserve the order of the plaintexts. OPE cannot be IND-CPA secure, since it does not only leak information about the order of the plaintexts, but it reveals other information too. We consider an OPE scheme perfectly secure, if the only possible leak can be caused by the order of ciphertexts. As shown in Figure 3.5, there are two different categories of OPE schemes. Category A includes schemes that apart from the order of ciphertexts, reveal other information too. In order to achieve a higher security level in such cases, the size of the ciphertext is increased. Still, this approach makes OPE schemes computationally impractical (in order to achieve the desired security level). The strongest security definition that an OPE scheme can achieve is IND-OCPA, proposed by Boldyreval in [1], but no existing work of category A has achieved it. Category B of OPE schemes comprises only the recent work of mOPE [2], the first scheme that achieves IND-OCPA i.e. does not leak any information besides of the ciphertext order. Moreover, mOPE achieves 1 to 2 orders of magnitude higher performance than any OPE scheme, as mentioned in [2] and therefore dominates all previous OPE approaches in terms of security and efficiency. For this purpose we study in detail this novel work in order to compare it with our approach. We will first describe the idea introduced by mOPE, shown in Figure 3.6.

In Figure 3.6 we note that mOPE consists of two parties the untrusted server and the secure OPE client. Regarding the first, mOPE considers two different models of adversary. The first one is **Passive or honest-but curious** and the other is **Active or malicious**. The paper first provides an analysis for passive kind of attackers and then mentions that an appropriate integration of message authentication techniques can be used in order to extend the approach to the case of malicious attackers. As shown in Figure 3.6, mOPE keeps encrypted numerical values in an OPE tree i.e. a Binary Search Tree that contains ciphertexts of the inserted values.

Each ciphertext of the OPE tree is arranged in such way, so as to allow the identification of its location simply by looking at the path from the root to down to the node. More specifically, beginning from the root and following the path towards the node of interest, we mark each left edge wih '0' and each right edge with '1'. The paths that are

Cat.	OPE scheme	Guarantees	Leakage besides order
A	Özsoyoglu '03 [13]	None	Yes
	Agrawal '04 [14]	None	Yes
	Boldyreva '09 [1] [4]	POPF [1]	Half of plaintext bits
	Agrawal '09 [15]	None	Yes
	Lee '09 [16]	None	Yes
	Kadhem '10 [17]	None	Yes
	Kadhem '10 [18]	None	Yes
	Xiao '12 [19]	None	Yes
	Xiao '12 [3]	IND-OLCPA [3]	Yes
	Yum '12 [20]	POPF [1]	Half of plaintext bits
	Liu and Wang'12 [21]	None	Most of the plaintext
	Ang et al.'12 [22]	None	Yes
	Liu and Wang'13 [23]	None	Most of the plaintext
B	Popa '13 [2](**mOPE**)	ideal : IND-OCPA	None

Figure 3.5: OPE schemes

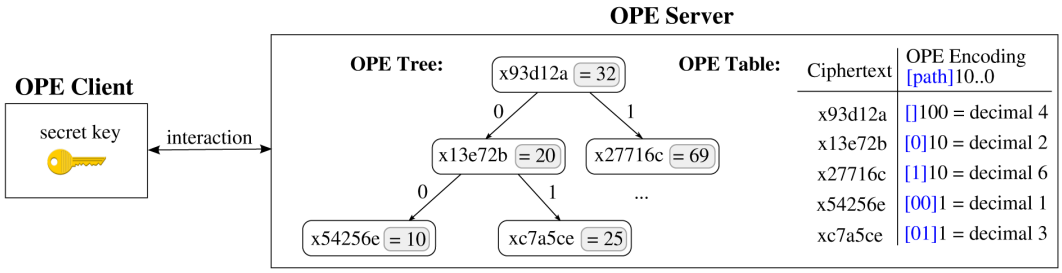


Figure 3.6: Overview of mOPE's data structure

created in the tree, with their respective labels allow us construct the OPE enconding using the following formula.

$$\text{OPE encoding} = [\text{path}]10...0$$

More specifically, an OPE encoding is constructed from the concatenation of bit values on the respective OPE tree path concatenated with '1' and followed by an appropriate zero padding. The length of an OPE encoding is fixed and predetermined. The OPE encoding construction has the interesting property of indicating the relative location of

a ciphertext in respect to the other ciphertexts. In Figure 3.6 we observe an OPE tree constructed for values (32,20,69,10,25), as well as an OPE table that contains all possible ciphertext pairs with their respective OPE encoding. Another way used in order to show that the OPE encoding works is by depicting the decimal representation of each OPE encoding that corresponds to the actual location of the ciphertext in the tree. Note that the purpose of the OPE table is not limited for the needs of an example, but mOPE [2] takes further advantage of it in order to improve efficiency and deal with issues such as stale encodings.

It is significant to maintain a balanced tree structure although it may require the implementation of balancing operations, in order to guarantee that OPE encodings stay short. Whenever the OPE tree is rebalanced the server side needs to update the stored OPE encodings to their new values, since certain ciphertext positions will have changed. Popa claims that the above procedure consists of locating and modifying previous stored OPE encodings and is completed in one pass using a summary with $O(LogN)$ size.

An OPE encoding is stored in the database for each tuple, thus allowing the execution of range queries. The function MOPE_ORDER, is responsible for calculating the OPE encodings of the intervals corresponding to a specific query range. For that purpose, mOPE first checks whether these intervals have been considered and the respective encodings are in the OPE table. If not found, these intervals are neither in the OPE tree nor in the database and in that case the OPE tree is used in order to find the tightest intervals for the specific range. Assume that the following query is to be executed:

Select * from secret where val> 5

At first, mOPE transforms the above query to the one shown below:

Select * from secret where val>MOPE_ORDER(5)

First, mOPE will search for the encryption of 5 (Enc(5)) and for its corresponding OPE encoding in the OPE table. If found, then the OPE table will return the OPE encoding of 5 and will request from the server to return all OPE encodings bigger than 5. Otherwise, mOPE looks for the tightest interval bound. For example, if encodings for the values (1,2,7,9) exist in the OPE table and there is not one for value 5, the mOPE considers value 7 as the closest upper bound and the corresponding OPE encoding is returned in order to answer the range query.

After determining the OPE encodings, the cost of retrieving the answer depends on the structure of the database. In other words, the existence of indexes allows for logarithmic execution cost of range queries.

Popa also claims that both the client work and the number of OPE encoding bits required for the above procedure are $O(LogN)$, where N is the number of encoded values, due to the logarithmic tree height. Furthermore, rebalancing requires only order information which is provided to the server by the tree and therefore the client side is not involved in this process. Note that during rebalancing few ciphertext nodes may be relocated, but the actual number of affected ciphertexts is equal to the number of children in their respective subtrees. More specifically, assume that a ciphertext is relocated to a higher tree level and hence its relative order with respect to the other ciphertexts as well as its OPE encoding are also modified. Then, the respective children will also have new OPE encodings since the order has changed. Also, two different tree structures were considered for the mOPE scheme, the scapegoat tree and the B-tree. Although the latter does not have logarithmic worst-case cost, experiments indicated that its actual cost was smaller compared to the first and there was a lower average of updated ciphertexts. Unlike B-trees, scapegoat trees which are self-balancing binary search trees, have $O(LogN)$ cost in the suggested model but are only recommended if the proposed scheme is embedded in another theoretical one with constraints of asymptotic performance on the server side.

MOPE also uses the notion of mutability on ciphertexts. **Mutable ciphertexts** are defined, as ciphertexts that change over time, for a small number of plaintext values. They show that the existence of mutability is necessary in order to satisfy the IND-OCPA security definition. More precisely, Boldyreva in [1] proves that **any** IND-OCPA secure scheme must create ciphertexts with size **exponential** to the size of the corresponding plaintexts. For example, if we wanted to encrypt a 64 bit number, with the use of an OPE scheme, then 2^{64} ciphertext bits would be required so as to guarantee the desirable security. Popa also mentions that the existence of mutability is necessary in order to render the encryption scheme practical (without exponential length size ciphertexts).

Finally, Popa claims that only in the case of database systems a stronger notion of security is possible and can be achieved. This notion is defined as, same-time OPE security (stOPE) and presupposes that the adversary is only aware of the order of items placed in the database at the same time. Moreover, it consists an improved and refined

version of mOPE since it guarantees a stronger security definition.

Drawbacks of mOPE

We present the drawbacks of mOPE after verifying whether or not it satisfies the requirements of an optimal approach mentioned in 3.1.2.

- As all OPE schemes, mOPE reveals the ciphertext order.
- The mOPE approach is the only OPE scheme that achieves the IND-OCPA security definition. Yet, mOPE still does not achieve the desired security level.
- An OPE encoding is stored for each tuple located in the database, resulting to frequent attacks caused by the untrusted server. We assume an adversary aware of the domain of the encrypted values, who can derive statistical information from any distribution leakage. Thus, she can create a mapping between actual records and OPE encodings and crack the mOPE encryption scheme.

3.2.3 Bucketization Approaches

In VLDB'04 Hore [24] proposed a privacy preserving range querying technique using the idea of bucketization, in order to make the execution of range queries more efficient. Hore's approach assumes a realistic model where a reliable client and an untrusted server exist. A similar approach is proposed by Hacigumus, in [25]. Many ideas presented in the work of Hore were initially introduced by Hacigumus, with the difference that Hore offers a more complete and applicable approach in the case of range queries and in terms of security. The key idea of this approach is that it splits the query Q into two components, $Q_{secure} + Q_{insecure}$, where $Q_{insecure}$ is executed on the server side and on encrypted values, so as to compute a super set of results of Q. Q_{secure} is executed within the safe limits of the trusted hardware (client side), in order to filter out the **false positives**. The objective is to push as much work as possible, concerning the query execution, to the server side. Regarding the bucketization approach, an attribute domain is partitioned into a set of buckets, each one of which is identified by a **tag**; these bucket tags are defined as **crypto-indices** and are used by the server in order to execute range queries. In this paper Hore, presents privacy threats arising from the creation of **bucketization-based indices**, that need to be taken into account. He also provides different algorithms for bucketization schemes, depending on whether the priority should be given on privacy or

efficiency or equally on both, so that an appropriate one can be selected depending on the requirements of the system.

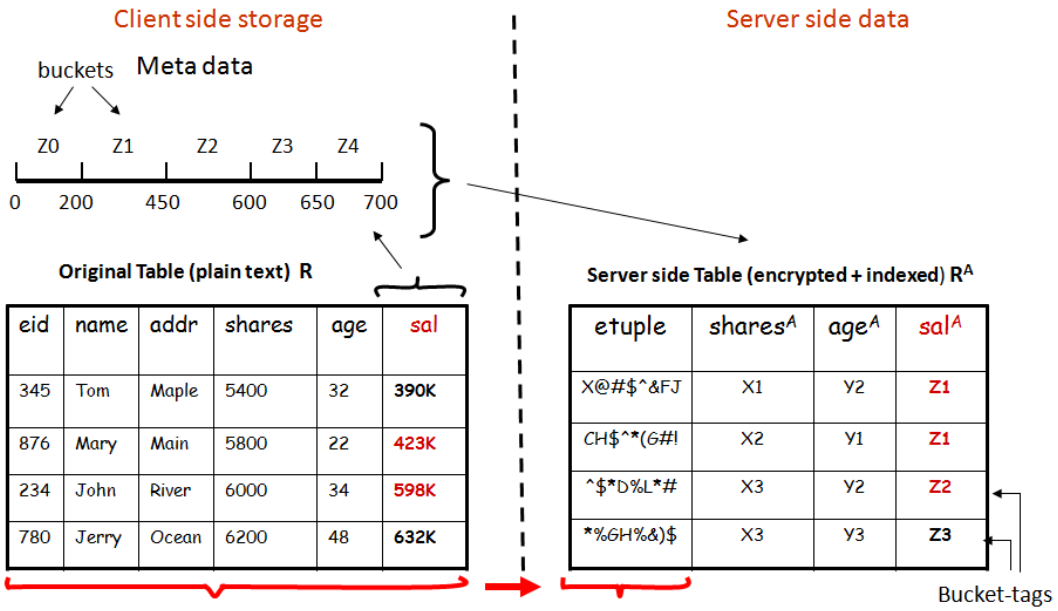


Figure 3.7: Data storage model of the Hore architecture

Using Figure 3.7 the key idea of Hore's approach will be described. A relation table is shown on the left and its server side representation is shown on the right. Hore's idea is to encrypt each row of the table and maintain it as a single **tuple** on the server. The above process is executed as follows. For each attribute to be queried, the algorithm partitions the respective values into buckets in some manner and stores the bucket-tags of these tuples as indexing information on the server **instead** of storing the actual values. The algorithm also stores this **value-to-bucket** mapping information on the client, which is used for translating user queries to server-side queries. The following example shows the execution of a simple query.

Select * from R where R.sal $\in$ [400K,600K] (Client-side query)

Using the metadata which is also shown in Figure 3.7 this query is translated into the following.

Select etuple from R^A where $R^A.sal^A = z1 \vee z2$ (Server-side query)

As a result, 3 rows are selected and returned to the client. The client-side should discard the one record that is **false-positive**.

Hore designed algorithms in order to study **privacy-efficiency trade-offs** in the case of bucketization schemes. At first he designed an optimal data bucketization-based solution for the case of range queries. In order to have some measurement parameters, he presents certain precision metrics that need to be maximized over all possible range queries. The respective cost for optimal bucketization has time complexity of $\mathbf{O}(n^2M)$ and space complexity of O(nM), where n is the number of distinct values of the dataset and M indicates the number of buckets. Afterwards, a set of privacy analysis and measurements need to be taken for every instance of the bucketized data. Hore also assumes that adversary A knows the exact distribution of values of every bucket and hence tries to resolve this issue.

Furthermore, Hore introduced the notion of Value Estimation Power **(VEP)**. This value indicates the adversary's ability to localize the value of an instance, like for example to determine the salary of an individual. Moreover, he presented the notion of **average error** that constitutes the error in the adversary's guess about a value and this number is lower bounded by the variance of the bucket distribution it comes. Note that the proposed work considers the variance of a bucket distribution as an inverse measure of VEP i.e. the bigger the variance, the smaller the chance of defining the value of an instance. Another metric regarding privacy, is the Set Estimation Power **(SEP)** of the adversary. An intuitive approach for this measurement can be the following; assume an adversary that wants to identify all salary records with values in a given range. The ability of answering the specific query is defined as the SET of the adversary. Moreover, Hore proposes Entropy as an appropriate measure of SEP; entropy is a universal measure of uncertainty and Hore considers entropy as the second metric of privacy, whereas the first was SEP. In terms of privacy, the objective goal is to maximize the SEP and Entropy metrics.

As mentioned in [24], the combination of metrics in Optimal bucketization may lead to less privacy than required. Small variance leads to partial disclosure of the numeric values, while small entropy leads with high probability to a complete leak of information. For this purpose, Hore developed a novel privacy-preserving re-bucketization technique which yields bounded overhead while maximizing the security level. This re-bucketization

technique allows trading-off bounded amount of query precision for greater variance and entropy.

The key idea of the re-bucketization technique is that it first creates Optimal Buckets and then diffuses the elements placed in these buckets in order to construct new composite buckets. The regulated diffusion that is applied, aims to maximize the entropy and the variance of the new composite buckets. In order to perceive the trade-off between efficiency and privacy, a control parameter is introduced and defined as the max degradation factor K. It is used in order to adjust the level of efficiency to the desirable level, while degrading the security level of the system. The use of Controlled Diffusion algorithms increases the metadata size from O(M) to O(KM) and the number of buckets that need to be retrieved in the case of range queries is K times greater than before.

Review of Hore's approach

Moreover, computations are executed in an efficient manner on the server side. However, the time complexity regarding range queries depends on the bucketization technique that is implemented and the security parameter K. A cost analysis is not presented in their work due to the existence of false positives. Still in the specific protocol false positives add an extra cost on the client side, since the server returns a super set of the answer. As a result the client has to decrypt the whole answer that he received, in order to filter out all false positives leading to an often prohibitive additional cost, especially in the case where the client is on a mobile device. Furthermore, updating data is expensive, since a re-distribution is required. In terms of security this approach does not provide any proofs but uses the trade-off between efficiency and privacy. Therefore, in order to achieve the desired security level, they suggest increasing security parameter K, which however implies degradation of efficiency. Finally, this approach is vulnerable to leakage of the value distribution.

3.2.4 Distribution instead of Encryption Approaches

This family of approaches aims to reduce the encryption and decryption overhead. They prefer extending the distribution of the data among the servers instead of encrypting the data. The work in [26] and [27] focuses on both, distributing the data and providing an efficient query processing technique, with the use of secret sharing algorithms. The work of Emekci [27] provides operations, such as intersections, joins and aggregations,

but it does not support range queries on the data, unlike Agrawal [26]. More particularly, Agrawal proposes an approach that offers great functionality, such as exact match, range and aggregation queries, as well as join operations. His work also considers updates and allows nominal attributes that are converted into numeric ones.

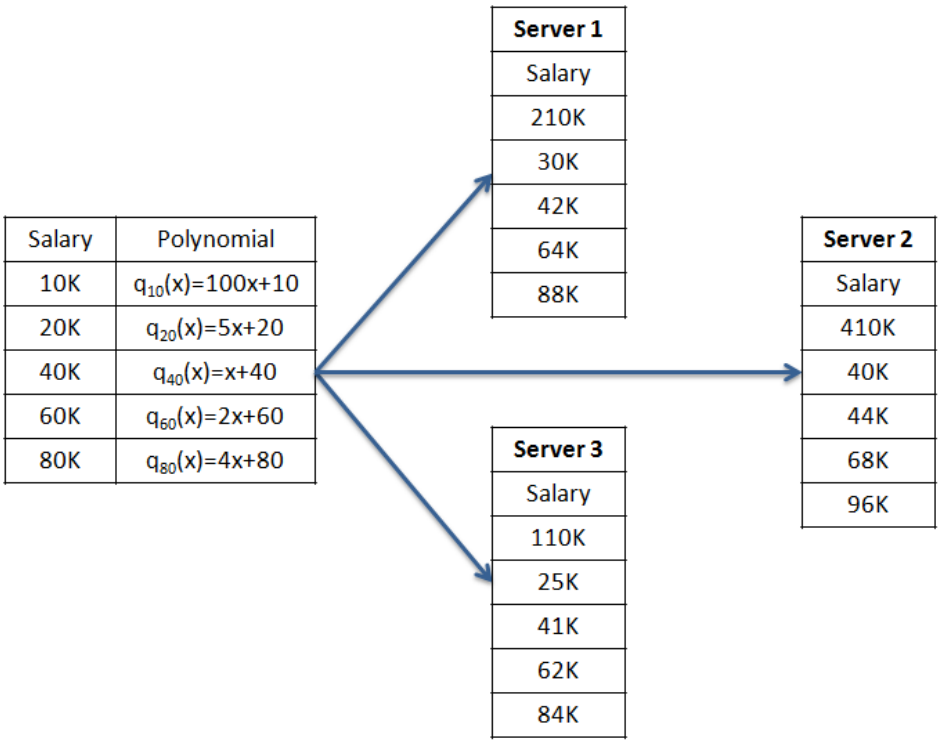


Figure 3.8: Example of Shamir's Secret Sharing Algorithm

The approaches that we are currently studying are based on Shamir's secret sharing method that operates as follows. Data source D, is responsible for dividing numeric values u_s into n partitions and storing each one of them on a different server, in order to know any $k \leq n$ partitions. Additionally, some secret information X is known only to the data source D. These two elements are sufficient in order to reconstruct the secret. This method is information theoretically secure, since even if X is known, having complete knowledge about the k-1 servers is still not sufficient in order to derive any information about the secret. As for the secret sharing method, data source D selects a random polynomial q(x) of degree k-1, where the constant term comprises the secret value u_s and secret information X which is a set of n random points, each corresponding to one

of the servers. Then, data source D computes the partition that will be given to to each server as q(x_i), where $x_i \in X$ and propagates it to the respective server. In Figure 3.8 an example is shown, of Shamir's algorithm for n=3, k=2 and X=x1=2,x2=4,x3=1, one for each server.

In order to support range queries Agrawal proposes in [26] two solutions. The first uses the labeling technique which as mentioned in the paper is completely insecure, while the second is based on a modification of Shamir's algorithm and uses an order preserving polynomial construction technique for a specific type of values. However, the latter does not provide any proof of the security achieved in the case of range queries. Still, in the case of exact match queries (joins with equality predicates) the same approach achieves the same security level with Shamir's secret sharing algorithm.

3.2.5 Tamper-Resistant Trusted Hardware

An alternative way for solving the problem of privacy preserving range querying is introduced by the Hardware community. Tamper-resistant secure hardware provides a secure execution environment (board), with restricted processing and memory resources. In the case of illicit physical handling, the devices are programmed in such way, as to destroy their internal state and shut down. The advantage of such approaches is mainly that they support almost any existing DBMS functionality, while the cost per query appears to be orders of magnitude lower than any existing software mechanism. Current work from this field is TrustedDB [28], where in order to avoid having secure CPU-related storage limitations, the outsourced data is stored in the host provider and the query processing takes place in both the server and the secure CPU side. A more optimized approach is the Cipherbase [29], which however affords limited computing and memory resources. Hence, neither a large number of clients, nor a high throughput can be supported. Furthermore, in this approach the secret key is passed from the user to the trusted hardware, which needs to be performed with great caution. Finally, in the case of range queries, the approaches studied in this Section have as much leakage as CryptDB [9] and MONOMI [10]. Therefore, the privacy preserving range querying has not been resolved either by the field of hardware.

Chapter 4

Our Approach

The basic notion of our approach is that we want to transform range queries into point queries, in order to be able to answer securely to encrypted range queries. Our aim is to answer to range queries without **revealing the order of ciphertext**. A naive approach to this direction would be to answer to range queries related to ages that belong to the range from 19 to 23, by separating the range into unique units (19,20,21,22,23), encrypting each of these values under the key and request from the server all encrypted tuples with values (19,20,21,22,23) separately. The basic drawback of the above solution is its efficiency, since for each of the unique values, the server will need to search the entire database. Furthermore, this sequential scan might lead to a leak of information, concerning these values. In order to support range queries efficiently and at the same time satisfy the security primitives, we use dyadic intervals as mentioned in Section 2.3. Our method achieves a desirable **logarithmic complexity** stemming from the dyadic intervals and provides strong security guarantees. Our approach combines deterministic and non-deterministic encryption schemes which apply to the distributed Key-Value store, stored in the cloud.

In Section 4.1 we present the basic architecture, we describe the interaction between cloud and client and the way in which we have used the dyadic intervals, so as to achieve the desirable logarithmic cost. In Section 4.2, we outline the security level of our approach. In Sections 4.3 - 4.4 we provide solutions to more difficult, realistic and complex attacks. Finally, in Sections 4.5 we discuss how we enforce and make efficient updates in our privacy preserving data structure.

4.1 Architecture

In order to provide solutions to more complex security problems than those tackled by related work, we propose a modification of the standard Key-Value store, which allows the execution of the following commands:

$$\text{get}(\ \text{Enc}(sk_{Key},\text{"id"}),\ \text{Server}_{ID})$$
$$\text{put}(\ \text{Enc}(sk_{Key},\text{"id"}),\ \text{Enc}(sk_{Value},\text{"value"}),\ \text{Server}_{ID})$$
$$\text{remove}(\ \text{Enc}(sk_{Key},\text{"id"}),\ \text{Server}_{ID})$$

These correspond to the basic operations used to enable the interaction between the cloud infrastructure, containing data organized in a **distributed Key-Value** store, and the user(data owner). This modification of the standard Key-Value store's commands allows us to determine the server on which each of them has to executed. Every **key** in the Key-Value store is encrypted and corresponds to a specific range, e.g. for range $[\alpha, \beta]$ the key is "α-β", while for identity ranges like $[\alpha, \alpha]$ the key is "α-α". The **value** in the Key-Value store corresponds to a list of tuple ids or a list that contains the tuples themselves and these lists are where the key refers to. Without loss of generality, we assume that the value in a Key-Value store is a list of tuple ids, that correspond to encrypted tuples.

We use different encryption schemes, depending on whether we want to encrypt the key or the value.

Key Encryption: We encrypt the key of a Key-Value pair using a deterministic algorithm DET, which means that every time we encrypt the value x under a specific key, the ciphertext remains the same. This property allows the existence of **search-ability**. The deterministic nature of DET also guarantees the IND-DCPA security level.

Value Encryption: We encrypt the value of a Key-Value pair using a randomized encryption scheme RND, so that every time we encrypt a message under a specific key, the produced ciphertext is different. Another reason for choosing a randomized algorithm, is because it achieves IND-CPA. In Section 4.2, we will focus on encryption schemes that belong to the family of DET and RND and a security analysis for these encryption schemes will be provided.

The ranges' formation of our keys in the Key-Value store follow the dyadic intervals' approach. As shown in Figure 4.1, each node of the tree reflects a dyadic interval and

is represented by a key in the Key-Value store. Starting a description from the bottom to the top of the tree, we need to mention that the leaves are the nodes that store the encrypted tuples of the corresponding values, while nodes in higher levels store the union of their descendants' nodes. Assuming that we want to execute range queries on the column Salary, the first leaf node with node id 1 stores all tuples corresponding to salary 1K, the second leaf node with node id 2 stores all tuples that correspond to salary 2K etc. As a result, the dyadic interval "1-2" contains the union of the tuples of both node 1 and node 2. Similarly, we can construct the dyadic tree, as displayed in Figure 4.1.

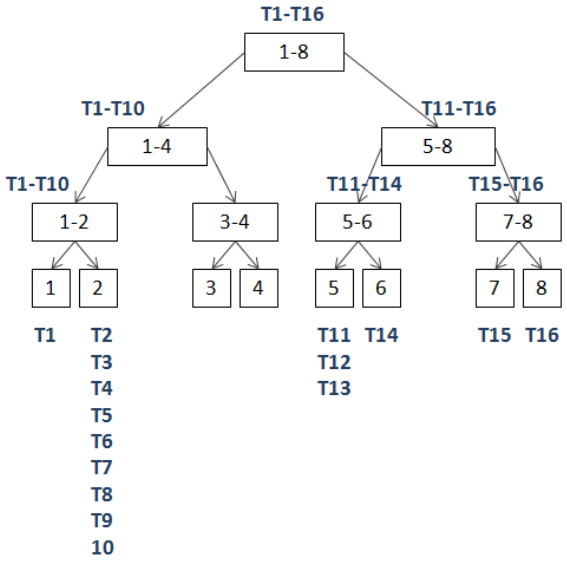


Figure 4.1: Dyadic tree

A brief description of how the range query execution is performed is essential in order to comprehend different kinds of weaknesses that need to be resolved. Consider the following range query that is performed on a column Salary:
SELECT *
FROM TABLE as T
WHERE T.SALARY >= $2K$ AND T.SALARY <= $7K$.

Firstly, we break the above range into the following dyadic intervals: **2, 3-4, 5-6** and **7**. This sequence of intervals constitutes the minimum dyadic cover, explained in Section 2.3. Then, the following commands are executed:

$$\text{get}(\text{Enc}(sk_{Key}, \textbf{"2"}),\text{Server}_1),$$

get(Enc(sk_{Key}, **"3-4"**),Server$_1$),
get(Enc(sk_{Key}, **"5-6"**),Server$_1$),
get(Enc(sk_{Key}, **"7"**),Server$_1$)

, assuming that we have already stored an encrypted form of the requested values on the server and that there is only one of them. Then, the server returns the requested values in an encrypted form and it is up to the client side to decrypt these values, under their secret key. The range queries that we described are answered in $O(Log(\beta - \alpha))$ **time complexity**, where α,β are integers which form the requested range $[\alpha, \beta]$. For simplicity purposes, we refer to this complexity by the upper bound of $O(Log(N))$, as $b - a \leq n$, where N is the size of the domain space. As for the **space complexity**, it is $O(MLogM)$, where M is the number of the tuples and N the maximum value of the domain. For example, if we have integer values , then N = 2^{32}.

4.2 Security level

In this Section we will describe the usage of encryption schemes DET, RND by our approach and their security analysis.

DET: Using the notation DET, we refer to any deterministic algorithm that has the security property of a pseudo-random function (PRF) or pseudo-random permutation. DET consists of 3 parts, a key generator, an encryption algorithm and a decryption algorithm. In order to summarize this functionality of DET we are using the following notation:

DET = (DET.KeyGen,DET.Enc,DET.Dec)

DET.KeyGen: KeyGen takes as input κ, which is a security parameter and produces as output a secret key sk. ($sk \leftarrow KeyGen(1^{\kappa})$)

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

DET.Dec: Dec is the decryption function that takes as input a ciphertext c and a secret key sk and outputs the message ($m \leftarrow Dec(sk, c)$). As we have already mentioned in the Background section, DET is characterized by the following property:

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

Using the above property, all messages $m \in \mathcal{M}$ become **search-able**.
RND: Using the notation RND, we refer to any non-deterministic algorithm that has the security property of a pseudo-random function (PRF) or a pseudo-random permutation (PRP). Similarly to DET, RND is comprises 3 parts, a key generator, an encryption algorithm and a decryption algorithm. Furthermore, in order to summarize the above functionality of RND we using the following notation:

$$\text{RND} = (\text{RND.KeyGen}, \text{RND.Enc}, \text{RND.Dec})$$

RND.KeyGen, RND.Enc and RND.Dec are defined similarly to the DET encryption scheme. As we have already mentioned in the Background section, RND has the following property:

$$\forall\ m\ \in \mathcal{M}:\ Enc(sk, m) <> Enc(sk, m)$$

We use the DET encryption scheme, in order to encrypt the key in the Key-Value store; the keys correspond to the identifiers of the dyadic intervals that are unique. This means that the encryptor (the client) never encrypts the same dyadic interval twice and consequently our DET approach achieves the IND-DCPA security definition. On the other hand, the RND scheme's usage upon the values of the Key-Value store leads to the achievement of the IND-CPA security definition.

In order to implement both of the above schemes, we make use of two Block ciphers, the AES-CBC mode and the AES-Counter (AES-Ctr) mode, which have been already described in the Chapter 2 . The only difference between these implementations is that in the case of the DET encryption scheme we use a **fixed-IV** while in the case of the RND encryption we use a **random-IV**. The security analysis of those schemes is provided in the Appendix A, where we end up to the following conclusions.

AES-Ctr can be IND-CPA in the case of RND by using a random IV, but it does not enable the design of a secure DET. Moreover, in the case of RND, AES-Ctr is more efficient than AES-CBC, since for the same security level it can encrypt more AES-blocks than AES-CBC. Besides this, it is completely parallelizeable and easy to implement due to the fact that the same circuit is used for both, encryption and decryption.

Furthermore, in the case of deterministic encryption schemes, such as AES-CBC, the required level of security is guaranteed, if the size of the message is less than 1 block i.e. 16 bytes. If the values' domain consists of integers ($n = 2^{32}$) or double int ($n = 2^{64}$), then

16 bytes are enough to index all the dyadic intervals in our approach. If the 16 bytes were not enough, we could propose other encryption schemes, such as Synthetic-IV [30] that has an extra property of DAE (Deterministic Authenticated Encryption), or CMC in [31], EME [32] and XTS [33], that are regarded as disk encryption schemes and can be used in the case of more than 16 bytes. However, in terms of efficiency and if we are encoding less than 16 bytes, AES-CBC appears to perform better than the rest of the encryption schemes we mentioned above. The conclusion of the above analysis is that we can use one of the implementations described, in order to be secure against distinct chosen plaintext attacks that aim for the key of the Key-Value pair (encrypted dyadic intervals). Another aspect of our approach, which is achieved through the use of random IVs, is the ability to encrypt messages of arbitrary size.

In this Section, we mentioned that either for the key, or the value of the Key-Value pair, we use encryption schemes that allow us to guarantee strong security definitions, as in the examples of IND-DCPA or IND-CPA. These definitions claim that all the key and value ciphertexts in the Key-Value pair are **distinctly** secure. We show that in case of a more powerful attacker, then the combination of the key and the value in the Key-Value pair may reveal information. This problem is called the distribution problem, for which we propose an efficient solution in Section 4.4. We firstly issue our approach towards resolving the Query Access Pattern problem.

4.3 Query Access Pattern Problem

Unlike OPE schemes, our approach answers efficiently to range queries, without revealing the order of the ciphertexts to an untrustful server. In order to achieve this, we propose a solution to the the Query Access Pattern problem, which will be described below.

The recent work in EDBT'13 [34] introduces a novel attack for Precise Query Protocols i.e. protocols and schemes that return the exact query answer without false positives. Related work on these protocols though, does not examine a solution to this problem.

In order to solve the Query Access Pattern problem, we could utilize two different approaches. The first one would be to apply a technique from the Oblivious RAM area. The main notion of those techniques is the replacement of single queries by sets of queries. The original Oblivious RAM model though requires $O(Log^4 n)$ queries, while in [35], a

technique is proposed that requires $O(Log^2 n)$ queries and $\sqrt{n}$ space requirements at the client side, where n is the number of encrypted data.

Another possible approach towards the tackling of the Query Access Pattern problem would be the periodical re-encryption of the ciphertexts, at a frequency greater than the one needed to launch an access pattern based attack.

The adoption of either one of the above techniques though, would incur an additional cost. We instead propose an efficient solution that exploits the structure of the dyadic interval and the nature of cloud environments i.e. the number of servers (nodes) available. The only constraint is that the servers can communicate but they cannot collaborate towards the achievement of a malicious goal. In the next Section, we describe the trade-off between the number of servers provided and the achieved security level. It is worth mentioning that the single server's case may be reduced to the existence of an arbitrary number of available servers. In this case though, all of these servers collaborate with each other, forming a single group. As a result, the case of the k-server approach corresponds to k non-collaborating and disjoint groups of k or greater number of nodes. Thus, our approach is scalable and the restriction upon the number of servers has to do with the amount of them that will not collaborate with each other.

4.3.1 Single Server Approach

First, we use an intuitive way to illustrate why the Query Access Pattern problem exists. We consider a query requesting all salaries with values between 2K and 6K. Then, we break this range into dyadic intervals and the following dyadic intervals are created: 2, 3-4 and 5-6, which we will be later requested in an encrypted form. The mapping created for the three intervals we have obtained is as follows:

$$a \leftarrow \text{"}2\text{"}, b \leftarrow \text{"}3-4\text{"}, c \leftarrow \text{"}5-6\text{"}$$

The number of possible combinations that can be formed is equal to 3! and all possible combinations are displayed below:

1)abc, 2)acb, 3)bac, 4)bca, 5)cab, 6)cba.

Assume that a legitimate user additionally executes queries 2K - 4K (which is the ab sequence) and 3K - 6K (which is the bc sequence). A server observing these queries

understands that a is a neighbor of b, but it will not be able to determine whether a has a bigger value than b or vice versa. Similarly, in the case of the bc sequence, the server will not be able to determine whether b has a bigger value than c or vice versa. However, using the above information the server is aware that a is a neighbor of b and c is a neighbor of b. Therefore, it can exclude all cases, where a and b, as well as b and c are not neighbors, i.e. cases 2, 3, 4 and 5, in the above example. Immediately, it becomes clear that the server has reduced the 6 combinations to 2 ($n! \rightarrow 2$), but it cannot decide whether the first or the sixth is valid. Thus, the server ends up having **two feasible sequences**, the **abc** and the **cba**. Notice that this can be observed for any sequence that can be created during a query. The first of these sequences constitutes the actual and the latter its mirroring sequence. This is defined as the **mirror property** and we will show that it applies for any Precise Query Protocol that does not examine the Query Access Pattern problem. The analysis of this claim is provided in the Appendix B.

The worst-case scenario of the single server approach occurs when all the possible range queries have been executed. By observing them, the untrusted server is able to determine the partial ordering of all those ciphertexts, leading to the ability of deducing two equally probable orders of them (mirror property). Even in this extreme case, our solution provides a higher security level than all OPE schemes, which by definition reveal the ordering of data. Despite this advantage of our approach against OPE schemes, in the next Sections, we illustrate the manner in which the Query Access Pattern problem is completely resolved.

4.3.2 Two-Server Approach

In the mirror property described above, we concluded that the $n!$ possible ordering combinations are reduced to 2 by observing the answers (dyadic intervals) of length 2. All these pairs are defined as **bad neighbors**, as their co-existence as part of an answer is the most harmful towards revealing the data order. For example, the two nodes "2" and "3-4" are considered as "bad neighbors", as they set the continuous interval 2-4. Similarly, the two nodes "2" and "3" are considered as "bad neighbors", as they set the continuous interval 2-3.

So as to avoid the above information leakage, our effort focuses on placing these "bad neighbors" into different servers. We distribute the nodes to the two servers as shown

in Figure 4.2. More specifically, each left child node is assigned to server 0 and each right child node is assigned to server 1. The root node can be assigned to any of the two servers. This partitioning prevents privacy leakage in the case of some of the overlapping queries that were provoking problems with the single-server approach. As an example, consider the request of the range "3-6", when the returned dyadic intervals are "3-4", "5-6", which are returned from different servers ($S1$ and $S0$ respectively). Thus, in this case, we avoid the information leakage described in our single-server approach.

On the other hand, querying for the range "3-8" will be decomposed to two dyadic intervals, "3-4" and "5-8", which will both be redirected to the same server. Therefore, our solution does not yet succeed to completely isolate the bad neighbors.

In order to achieve this, we relax our restriction upon returning the minimal dyadic cover as the query's answer. On the contrary, we break the interval that is located in the higher tree level and retrieve its information from its children. For example, we can retrieve the answers for the range 3-8 from the nodes "3-4" and the children of "5-8", i.e. "5-6" and "7-8". The time complexity remains $O(LogN)$.

Each of the two servers has half of the domain values, which means that his knowledge about the overall ordering of all dyadic intervals equals to:

$$2 * (2!) * (n/2)!$$

The use of the above 3 factors is justified as follows: The factor 2 that appears first reflects the fact that the adversary can observe only half of the domain. In this case, because of the mirror property the number of all possible permutations is reduced to 2. Furthermore, the number (2!) indicates the lack of knowledge concerning the relative position of the data stored in each of the two servers. The factor (n/2)! indicates that the attacker cannot observe sets of queries and responses, concerning the remaining half of the domain, for which she makes no observations. Thus, she is not able to exclude any permutations, as of the other server's contents. In Figure 4.2 we present a dyadic tree in which the domain is partitioned to two servers.

4.3.3 k-Server Approach

In this case, we can compute a lower bound for the number of combinations that an adversary can perceive. We assume that the answers' domain is equally and uniformly

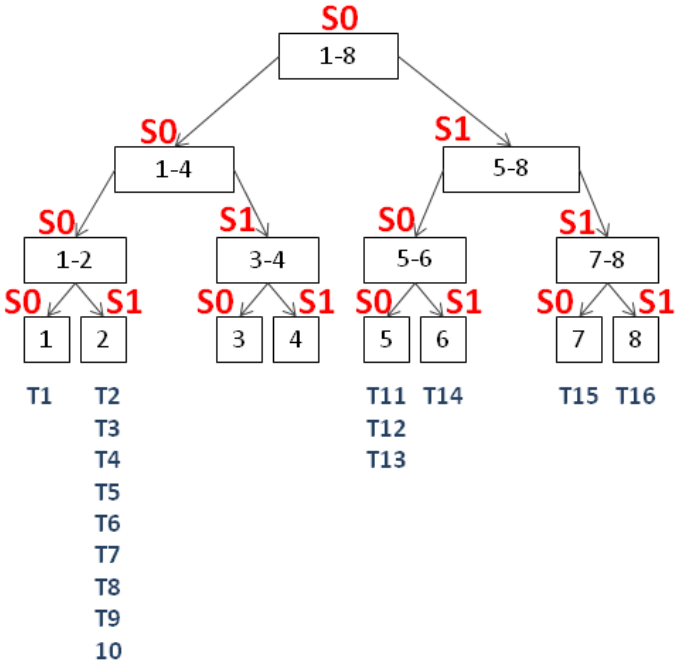


Figure 4.2: Distributed Dyadic tree in 2 servers

distributed to the servers. The following formula presents the number of remaining permutations in the case of k servers.

$$2 * (k!) * (2 * (k-1) * n/k)!$$

As to explain the above formula, we need to mention that in the case of k-servers approach, a server holds n/k of the overall dyadic intervals. In other words, by observing all the possible queries, an adversary is able to reduce the number of permutations from (n/k)! to 2; for this reason we first multiply by 2. Furthermore, the factor k! indicates that the specific server cannot determine the starting point of his sequence, compared to the other k-1 sequences. We already know that all the possible combinations that can be formed, are k!. Finally, the factor (2*(k-1)*n/k)! refers to the unknown portion of data, for which the other $k-1$ servers are responsible for.

The reason for which the above formula constitutes a lower bound of combinations, lies in the fact that when a server provides more than one dyadic intervals as part of an answer, it is not aware of the dyadic intervals' relative distance. Besides this, it knows neither the parameter k, nor the maximum path of dyadic intervals which forms an answer. In

general , it is straightforward that the greater the number of non-collaborating servers, the stronger the security guarantees that are achieved. It is worth mentioning that the parameter k is bounded by the maximum length of the minimal dyadic cover.

4.3.4 LogN-Server and 2LogN-Server Approach

We assume that we have in our disposal $LogN$ servers, which are exploited via assigning the contents of each level of the dyadic tree to a different one. This assignment is depicted in Figure 4.3.

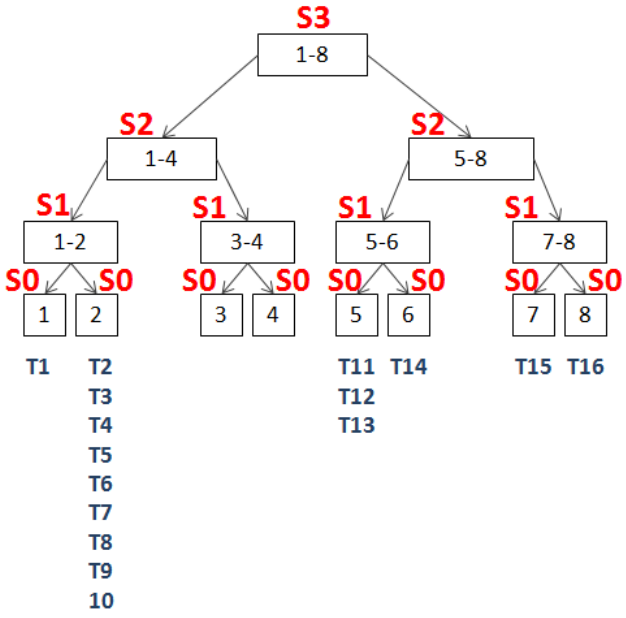


Figure 4.3: Distributed Dyadic tree in $LogN$ servers

This assignment policy guarantees that each node contributes by at most two ciphertexts, as a result of Property 4 described in Section 2.3. Also the $LogN$ number of servers is equal to **half** of the upper bound of the dyadic interval's maximum search path.

We claim that the assignment policy used in the $LogN$-Server approach prevents any information leakage, only if the server is not aware that the dyadic intervals technique is applied. Otherwise, this approach suffers from information leakage. For example, assume that we have a range query requesting for all salaries that belong to the range 2K to 7K. Then we will get the following dyadic intervals: "2", "3-4", "5-6", "7". We can see in

figure 4.3 that the first server contributes with the dyadic intervals "2" and "7". Now assuming it knows that our approach uses the dyadic intervals' technique, it still cannot be sure about which one of the two intervals comes first. However, due to the fact that dyadic intervals "1" and "8" are always requested alone, none of the requested dyadic intervals could be one of them. Conclusively, the number of ciphertext rearrangement combinations is not feasible to reach the desired n!, which corresponds to a completely secure solution.

The above limitation is avoided when we have $2LogN$ servers at our disposal. Since $2LogN$ is the longest possible path forming a minimal dyadic cover, there exists at least one suitable assignment that guarantees that each server contributes by at most **one** dyadic interval, as shown in Figure 4.4. A proof of this argument can be obtained by using the aforementioned dyadic tree's Property 4, the assignment of each left and right child to a different server and the fact that each answer will not consist of more than one left or right dyadic interval from the same level.

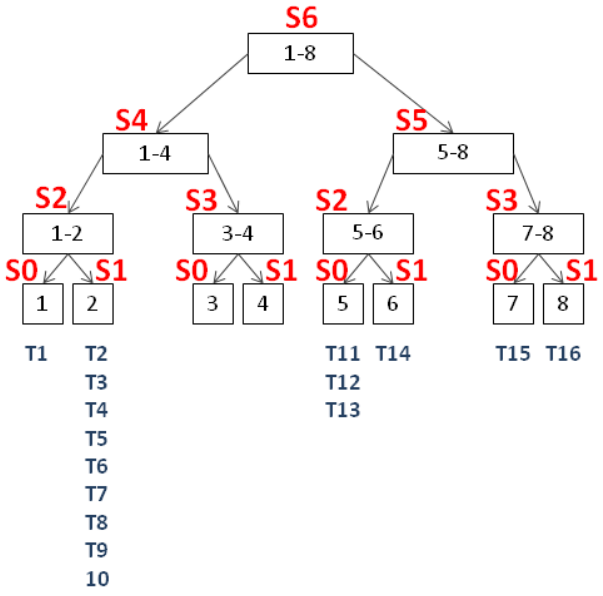


Figure 4.4: Distributed Dyadic tree in $2LogN$ servers

Beyond the solution of the Query Access Pattern problem, we have to face the leakage of the distribution followed by the stored values. This Distribution problem is not efficiently tackled by any related work. Our approach towards its solution follows.

4.4 Distribution

The distribution problem is illustrated through the following example: As shown in Figures 4.3, 4.4, server 1 contributes with nine tuples, in the case of 2K, while in cases 4K and 6K the same server contributes with zero or one tuple respectively. Therefore, the untrusted server draws the conclusion that the tuple corresponding to the 2K value is **more frequent** than the rest of the tuples. As a result, it gains knowledge about the distribution of the data, provoking information leakage. The occurrence of this problem is also met in an example that studies the salaries of a less developed country's population. In this case, the observation of a frequent tuple residing in the Key-Value store, would result to the assumption that this tuple corresponds to the country's minimum salary - as this is obtained by the majority of the population.

In order to address the aforementioned problem, we introduce an additional dyadic tree index. Briefly, the first level dyadic tree index stores a mapping between the actual values and their position on a secondary structure. This is represented by a second level dyadic tree that facilitates the distribution of all information in a **uniformed** way. It is clear, that if the distribution is uniformed at the lowest level of the tree, then it will be uniformed and at the rest of the tree. One way to achieve the desirable distribution, is if we convert the tree that is shown in Figure 4.4 to the tree that is shown in Figure 4.6. In order to perform this conversion we follow the steps that are described in the initialization algorithm.

In the initialization phase, we perform the group by function on the desired column, in order execute the range query. The group by is performed on entire tuples or tuple IDs. Then, we partition the groups into buckets of constant size. If the created buckets are not filled, then we use the zero padding technique, in order to hide from the server the actual size of the buckets. Each bucket is assigned to an id, which is based on an incremental counter and also corresponds to the leaves of the second level dyadic index, as shown in Figure 4.6. Consecutive leaves in the second level index, indicate that the value of the leaf i is smaller or equal to the value of the leaf i+1. In order to store the starting and ending point of each of the second level's index buckets, we need to save the corresponding ids in the leaves of the first dyadic interval tree. The rest of the procedure, consists of the construction of the rest of the tree i.e. the construction of the rest of

the dyadic intervals in the first and second level index. The algorithm of Initialization is presented in 1.

Algorithm 1 Initialization Algorithm

Input: Table as t, BucketSize as b,Secret key sk_{KEY} ,Secret key sk_{VALUE}

1: Group by t.Salary
2: $w \leftarrow 1$
3: **for** each group Gi **do**
4: Partition Gi into bucket size of b
5: $start \leftarrow w$
6: **for** each bucket bj **do**
7: put($Enc(sk_{KEY}, w), Enc(sk_{VALUE}, b_j)$)
8: w++;
9: put($Enc(sk_{KEY}, t.Salary[i]), Enc(sk_{VALUE}, start||" - "||w - 1)$)
10: Build the 1st level dyadic index
11: Build the 2nd level dyadic index

We are considering range queries on encrypted data and therefore the input of the Search Phase 1 algorithm includes a lower and an upper bound, that define the range we are interested in. Firstly, this range is broken into dyadic intervals. For each dyadic interval, we request from the first level index the corresponding encrypted value , using the command get(Enc(sk, "id")), as mentioned above. Then, we decrypt this value in the client side, under the secret key. Our aim in this phase is to find the minimum and the maximum value among all the dyadic intervals that we have obtained. These two values, are given as inputs in the Search Phase 2 algorithm. At this phase, the range that is defined by the min and max value, is broken into dyadic intervals. We request each one of the dyadic intervals that we have obtained from the second level dyadic tree and then we add them to our final answer.

For example, lets assume that we have the following query:
SELECT *
FROM TABLE as T
WHERE T.SALARY >= $2K$ AND T.SALARY <= $7K$.
Firstly, we execute the Search Phase 1 algorithm, using as inputs the values 2K and 7K. This range will be broken into the dyadic intervals: "2", "3-4", "5-6", "7". For each one

of the intervals we have obtained, we request from the first level index the Enc(sk,"2"), Enc(sk,"3-4"), Enc(sk,"5-6"), Enc(sk,"7"); we are querying the fist level index as it is shown in Figure 4.5. Then in the client side we decrypt the servers' response and we find the starting point and the ending point among all intervals, which in our case are the values 2 and 15 respectively. These values constitute the input of the Search Phase 2. Once more the input range is broken into the dyadic intervals: "2", "3-4", "5-8", "9-12", "13-14", "15". Finally, each of the above intervals is requested from the second level dyadic tree Enc(sk,"2"), Enc(sk,"3-4"), Enc(sk,"5-8"), Enc(sk,"9-12"), Enc(sk,"13-14"), Enc(sk,"15") and the servers' responses are added to the final answer.

Notice that our solution now addresses both the Query Acces Pattern problem and the distribution leakage problem; each server contributes with at most one key to the query's answer (assuming the availability of $2LogN$ servers) and the amount of information eventually contributed by each server to the final answer is always the same, regardless of the query or the distribution of values in the indexed data. The time complexity for each range query is $O(LogN + LogM)$, where N is the maximum unsigned value of the domain's data type and M is the number of $tuples/b$, where b is the bucket size. The space complexity is$O(NLogM + MLogM)$.

Algorithm 2 SearchPhase1

Input: LBound,UBound

Output: Answer

1: Break the range to dyadic intervals
2: Find min (starting point) & max (ending point) of all dyadic intervals
3: $Answer \leftarrow SearchPhase2(min, max)$

Algorithm 3 SearchPhase2

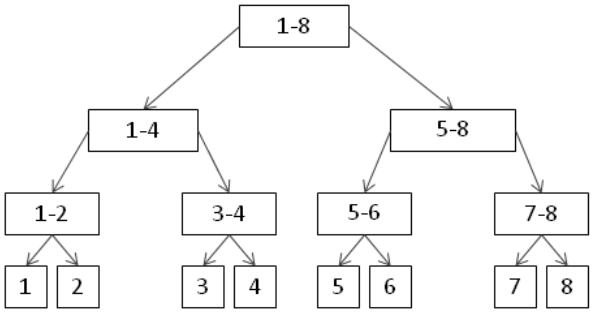
Input: min,max

Output: Answer

1: Break the range to dyadic intervals
2: **for** each dyadic interval i **do**
3: $\quad Answer+ = get(E(sk, i))$

Space reduction can be achieved, due to the fact that some of the dyadic intervals are never used. For example, assuming that we want to retrieve the salary 2K in Figure 4.6,

we need to retrieve this information from the nodes with value "2", "3-4","5-8","9-10". However, the 2K salary tuples are also stored in nodes "3", "4", "5", "6", "5-6", "7", "8", "7-8", "9", "10", that will never be retrieved individually. This enables us to avoid storing these buckets at the corresponding servers.



Actual Value	Starts	Ends
1K	1	1
2K	2	10
3K	0	0
4K	0	0
5K	11	13
6K	14	14
7K	15	15
8K	16	16
1-2K	1	10
3-4K	0	0
5-6K	11	14
7-8K	15	16
1-4K	1	10
5-8K	11	16
1-8K	1	16

Figure 4.5: 1st-level index

4.5 Updates

In this section, a description will be made about how to perform updates on our encrypted structure. We have already mentioned that during the initialization process buckets of size b are used and in case of a non-filled bucket, zero padding is applied. This is because we want to hide the distribution of tuple ids from the adversary. However, by allowing updates, the distribution of values in the dyadic tree is affected. In other words, the use of a second level dyadic interval tree is useless (we do not conceal the distribution).

In order to solve the above problem we use the idea of sparsity and we apply it in the buckets, using zero padding. We define a specific percentage of sparsity and if the bucket

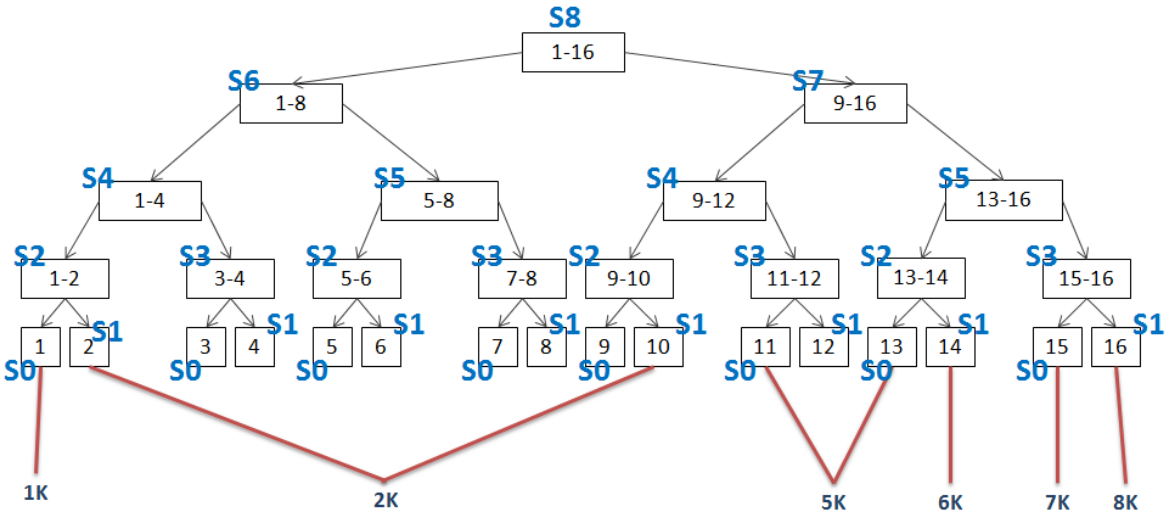


Figure 4.6: 2nd-level index

is not completely filled then we continue to implement zero padding. Nonetheless, during the initialization process, the empty buckets are created and encrypted according to the idea of sparsity.

We define as **good updates**, the updates that correspond to a value that has free space and therefore is convenient for insertion execution. On the other hand, we introduce the notion of **bad updates** as updates that do not offer free space for a new insertion.

In the case of bad updates the 1st and 2nd level dyadic tree intervals need to be rebuilt while in the case of good updates we can estimate whether or not there is enough free space in the corresponding nodes so as to execute the insertion. For this purpose we need to enrich the information contained in the first level tree by adding a supplementary column indicating the first empty point of the bucket in order to start the insertion process from there. This number is calculated by subtracting from the bucket size, the number of tuples that are already in it. As we have already mentioned the bucket size is fixed since we used zero padding and achieved equi-size buckets in the leaves of the 2nd level index. The first level dyadic tree shows if a good or a bad update is going to occur as well as the respective id in the second level index where the insertion should be executed. A significant observation at this point is that the querying happens only on the leaves of the first level index.

In order to execute a good update, firstly we need to update the corresponding information that is located in the first level index i.e. we need to increase by one the number of tuples. Then in the second level index, we need to propagate the update towards the ancestors of the leaf, where the new value was inserted.

Finally we need to mention that updates in the 1st and 2nd level index are executed in a specific order. At first the respective key-value pair is requested from the key-value store. Then the value is modified according to the updated value and the old key-value pair is replaced with the new one that contains the new value. The cost of the above procedures is equal to **O(n)** for every **insertion**.

However, in order to perform the above procedure more efficiently, we modify the storage policy of our structure. More specifically, the first level index stores additional information beyond that described previously, so as to achieve a more desirable performance in the case of insertions. On the leaves of the second level tree index we continue to store buckets, with respect to the sparsity approach. The same does not apply for the higher levels of the second level index, where the value in the key-value pair corresponds now to an array list of buckets located on the leaves. In Figure 4.7 we can observe the new storage policy. At this point we need to mention that buckets (B1,B2,...,Bn)are encrypted. In comparison to the previous storage policy, now each node does not contain a union of its descendants, but a union of the leaves of the tree, that corresponds to the respective dyadic interval. As we have already mentioned the value of the key-value pair contains an array list, that stores buckets that are located on the leaves of the tree. The position where each bucket is placed, depicts a path. This path starts from a reference node (current dyadic interval) and ends at the corresponding child (bucket); we mark every right child with the value '1' and every left child with the value '0'.

Lets assume that we want to execute a good update for a given value. Firstly, we request from the first level index the starting and ending point of the given value, in the second level index. Then, taking into account the information concerning the size of the bucket and the current number of values that are already stored in it, we can determine which leaf is appropriate in the second level index for the insertion. At that point, we retrieve the appropriate bucket and we decrypt it on the client side. Afterwards, we add the new value to the bucket and therefore the old bucket needs to be replaced by the updated one in every dyadic interval it appears in. The difference with the previous implementation is that now we do not need to download all key-value pairs that are

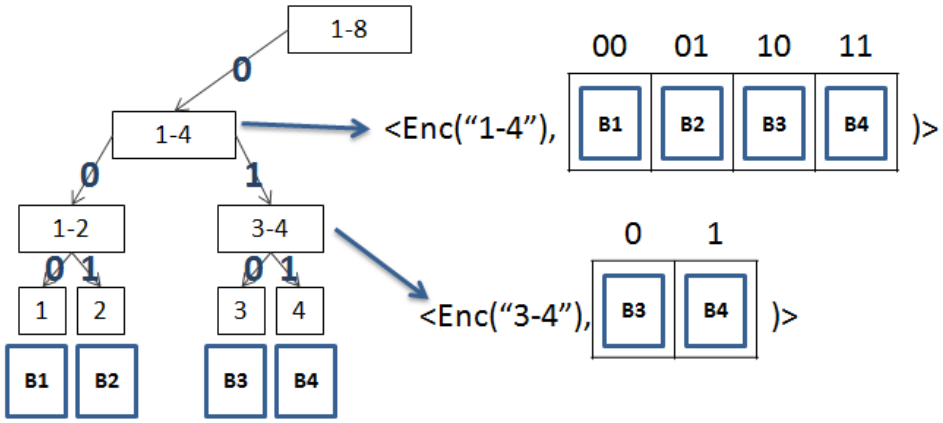


Figure 4.7: Update storage policy (2nd-level index)

located on the ancestors of the reference node, but we already know the exact position of the node with respect to its ancestors and can execute a **good** insertion with cost $O(LogN)$. The final step involves the updating of the first level index, with the new values that are produced after the insertion of the new value; we need to update only the corresponding leaf in the first level index, since updates interact only with the first level's index leaves.

We have decreased the computation complexity for a good update, but we have to consider the security level of the above procedure. We can observe that every dyadic interval in the second level index contains an array list of buckets in a sorted sequence, which means that the server is aware of the order of the encrypted values in the buckets, which forms a specific dyadic interval. Additionally the server that contains the root of the second dyadic tree, knows the order of all encrypted buckets, but it cannot identify the actual value included in a bucket. However, it can perceive the order of the buckets i.e. it can realize that bucket n follows bucket n-1 and is before bucket n+1. However this revealing of the order is less than the one presented in OPE encryption schemes, something that will be shown in the following paragraphs. We also show how is it possible in the case of good updates to hide any other ordering, besides the one regarding the buckets, without sacrificing the $O(LogN)$ complexity cost. This can happen by storing additional information in the leaves of the second level index.

Deletions can be handled in a similar manner, by first using Search Phase 1 and Search Phase 2, in order to retrieve the matching tuples existing in the second level

index. After receiving the answer of these dyadic intervals, we decrypt them and identify the desired value for deletion. These values enable us to modify the corresponding buckets, re-encrypt them and propagate the update to the respective dyadical interval leaves. Depending on the precise point within each bucket that is affected by an update, an additional transformation may occur, in order to preserve the aforementioned zero-padding accordingly. This computational cost sums up to: $O(logm + logn + log^2 n)$, due to the execution of both Search Phases, as well as relocating the affected buckets, which is asymptotically equal to $O(log^2 n)$.

Comparison of the security guarantees of our algorithm with updates of other OPE schemes

Our approach reveals less information than the other OPE schemes because in our case the order of the ciphertexts does not indicate the order of the plaintexts. Also, in order to encrypt the value of the key-value pair, we use a non-deterministic algorithm, when the OPE schemes use deterministic algorithms. More particularly, we have already mentioned that the security definition of IND-OCPA, that is used by deterministic encryption schemes, preserves the OPE property. The OPE schemes, that try to meet the IND-OCPA security definition suffer from **big jump attacks**, as first mentioned by Boldyreva [1]. Conclusively, any OPE encryption scheme should have ciphertexts with size at least exponential to the size of the plaintext in order to face the big jump attack. The mOPE [2] approach, replaces the ciphertexts with a size at least exponential to the size of the plaintext, using the idea of mutability. Our update policy does not suffer from the big jump attack problem, due to the fact that we use a non-deterministic algorithm which does not have the OPE property. Furthermore our update policy does not allow the execution of range queries in a subset of values of the dyadic intervals. However, our approach reveals the order of the buckets, which means that it is exposed to a server that has knowledge of statistical information and is aware of the domain of the values. For example if the server knows that the domain of the values corresponds to Greek salaries in 2013, then it can easily find out about the minimum salary in Greece and hence know the value of the first bucket. However, the same thing applies for the OPE schemes too. Finally, the main reason that renders our approach less vulnerable to the server compared to the OPE schemes, is that it resolves the problem of distribution as we have described in 4.4, while OPE schemes and the state-of-the-art mOPE approach, do not propose a solution to this problem.

Concealing the order of the encrypted buckets from the attacker

We have already shown that even if the encrypted buckets are stored in a sorted way, based on the order of the actual values, our encryption scheme still achieves IND-CPA. Nevertheless, if we want to hide this information from the untrusted server, we can randomly shuffle the buckets. Furthermore, in order to achieve the desirable logarithmic cost, in the case of good updates, we can store in the leaves of the second level index all positions where the specific bucket can be found in higher tree levels. In Figure 4.8 an example of this approach is shown.

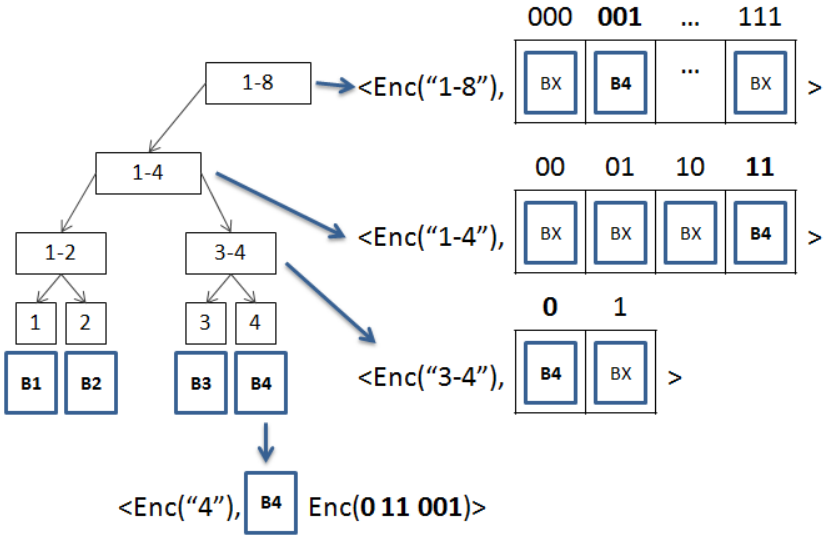


Figure 4.8: Update storage policy hiding the order (2nd-level index)

Additionally we can accurately specify the additional storage cost for each leaf with following formula.

$$1 + 2 + 3 + ... + Log_2N = \sum_{i=1}^{Log_2N} i = \frac{Log_2N(Log_2N+1)}{2}$$

, where N is the number of leaves (ids) in the second level index. The above formula shows that we require 1 bit in order to determine the position of the specific leaf in his ancestor node, 2 bits for the position in the ancestor of his ancestor node, ... , Log_2N bits in the case of the root node. We need to mention that the additional information, that is stored in the leaves is also in an encrypted form.

Our approach not only supports insertion queries, but deletion queries too. Firstly, we need to identify in the first level index which dyadic interval contains the tuple id that we wish to delete. Then, we request from the second level index the corresponding dyadic interval and we decrypt the respective buckets until we find the one that contains either the value or the tuple id of interest and we execute deletion. The old bucket is replaced with an updated one and this happens in all the dyadic intervals, where it is found. The final step is to update the number of elements in the first level index, that are respective to the given value, on which the deletion was performed. The cost of the above procedure, is the same as the one for a simple search query (search phase 1, search phase 2), which is logarithmic. We also take into account, the cost of replacing the bucket is also logarithmic and therefore the computational complexity is $O(LogN)$.

We have described how is it possible to efficiently execute updates in the case of **good updates**. In the case of **bad updates** we need to reconstruct the first and second level of the index and the respective cost equals to $O(NLogN)$.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Conclusively, our initial objectives have been met, since we are able to execute range queries on encrypted data in logarithmic time, without revealing the order of the ciphertexts and therefore preserving the semantically secure property which is defined by the crypto definition of security. We also presented a summary that comprises the weaknesses of approaches that have been proposed until now and to the best of our knowledge we are the first to simultaneously satisfy the desired efficiency and security requirements, while resolving the frequently occurring and serious issues of query access patterns and value distribution, since both of them remain hidden from the adversary. Moreover, update queries can be performed on our scheme. In the case where the value distribution problem is not considered, as happens in mOPE [2], then the update queries are answered in a logarithmic time and with the concealment of the ciphertext ordering. Otherwise, we differentiate and handle in a different manner the good and bad updates, where in the first case insertions and deletions are executed in logarithmic time, while in the latter both the first and second dyadic tree index should be reconstructed, with respect to the update. Additionally, assumptions regarding the adversary have been made and more specifically, we expect her to know statistical information about the distribution of the values, we presume that she is aware of the domain and we consider her honest but curious. Finally, in our approach we assumed that our scheme works in an environment of $2LogN$ cooperative servers.

5.2 Future Work

A significant matter not addressed in this thesis is the case of multi-dimensional range queries. Therefore, developing a generalization of the proposed approach is an important aim of the future work, in order to appropriately adapt to this family of queries.

The integration of different techniques into the proposed approach, could allow the execution of demanding and complex queries, apart from the range queries that are studied in this work, with respect to the desirable security and efficiency requirements.

Furthermore, an improvement of efficiency in the case of bad updates is essential, since currently a reconstruction of both dyadic trees is required and therefore the execution cost is significantly increased. It is also desirable to disengage the client from the query execution of bad updates and transfer the respective execution cost to the server side.

Moreover, the proposed approach could be integrated with techniques provided by Private Information Retrieval and Oblivious RAMs in order to satisfy constraints on the number of available servers, but with an additional computational and storage cost.

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).

Chapter 6

Appendix

6.1 Appendix A

Firstly we will use CBC and Counter mode theorems in order to show the security level of RND encryption schemes and then we will determine the security level of DET.

Based on the CBC theorem, that is described in subsection 2.1.3, we reach to the following conclusion:

$$Adv_{CPA}[A, E_{CBC}] \leq 2 * Adv_{PRP}[A, E_{B,E}] + 2q^2L^2/|X|$$

E_{CBC} is semantic secure under CPA (**IND-CPA**) as long as $q^2L^2 << |X|$, where q is equal to the number of the encrypted messages under the key k, L is equal to the maximum message length.

In practice, a negligible advantage of the attacker it is describe with the following notation:

$$Adv_{CPA}[A, E_{CBC}] \leq 1/2^{32} \Rightarrow$$
$$q^2L^2/|X| < 1/2^{32}$$

We consider that the first term is negligible ($2 * Adv_{PRP}[A, E_{B,E}]$ is negl.) as long as we are using a secure PRP. In the case of AES-CBC, whenever the key size is 128 bit long our encryption approach remains secure, if we change the key after every 2^{48} AES blocks, as shown in the following equation:

$$|X| = 2^{128} \Rightarrow qL < 2^{48}$$

The above security analysis for RND is based on the fact that we are using a random IV. In the case of DET we have to use a **fixed-IV**. Then AES-CBC cannot be indistinguishable under Chosen Plaintext Attacks (IND-CPA), as happens with all deterministic encryption schemes that have been already shown in 4.2. Therefore, in the same section we introduced the notion of indistinguishable under distinct Chosen Plaintext Attacks (IND-DCPA), in case of deterministic encryption schemes. The above definition of security is equated with the IND-CPA definition, under the assumption that the same message-key pair is never encrypted twice. In our approach, the above assumption is satisfied, since the key in the Key-Value pair consists a unique id. Moreover, AES-CBC is not DCPA secure, if the maximum message length is greater than 16 bytes. The following type of attack indicates that the adversary is able to win the security game, with non-negligible advantage.

Insecure CBC with fixed IV(FIV) ($|m| > 16$ bytes):
We play the IND-DCPA security game:
1. The attacker outputs $m_0 = 0^n1^n$, $m1 = 0^n1^n$
2. The encryptor sends to the attacker $c_1 \leftarrow [FIV, Enc(k, 0^n \oplus FIV), ...]$
3. The attacker outputs $m_0 = 0^n$,$m_1 = 1^n$
4. The encryptor sends to the attacker $c_2 \leftarrow [FIV, Enc(k, FIV)]$ or $c_2 \leftarrow FIV, Enc(k, 1^n \oplus FIV)$ Then the attacker outputs b=0 if $c_2[1] = c_1[1]$ with $ADV_{DCPA}[A, E_{CBC}] = 1$.

Lets assume that we encrypt less than 16 bytes, thus the CBC is allowed to encrypt only **one** block of 16 bytes. Then the previous kind of attack cannot occur, because then the attacker is not allowed to output messages which produce more than one AES blocks. In that case we also assume, that we never encrypt the same message twice under the same key. We have also already mentioned that we want to use DET on the key of the key-value pair, which is a unique id. Considering the above, the previous conclusion is transformed to the following:

$$Adv_{DCPA}[A, E_{CBC}] \leq 2 * Adv_{PRP}[A, E_{B,E}] + 2q^2L^2/|X|$$

CBC is indistinguishable under distinct Chosen Plaintext Attack (IND-DCPA) as long as $q^2L^2 << |X|$, as happens in the case of RND.

In particular, after 2^{48} AES blocks, the key must be changed, in order to achieve IND-DCPA (indistinguishable under distinct chosen plaintext attacks).

We can increase the number of AES blocks that can be encrypted under the same key, if we use AES-CBC with a 256 bit length key.

Otherwise, we can consider a different block cipher, like for example the **ctr-mode**. Based on the **Counter-mode theorem**, that was described in 2.1.3 we reach to the following conclusion in the case of RND (with Random IV):

$$Adv_{CPA}[A, E_{CTR}] \leq 2 * Adv_{PRP}[A, E_{B,E}] + 2q^2L/|X|$$

E_{CTR} is semantically secure under CPA (**IND-CPA**) as long as $q^2L << |X|$, where q is equal to the number of the encrypted messages under the key k, L is equal to the length of max message.

In practice a negligible advantage of the attacker could be the following.

$$Adv_{CPA}[A, E_{CTR}] \leq 1/2^{32} \Rightarrow$$
$$q^2L/|X| < 1/2^{32}$$

since the first term is negligible ($2 * Adv_{PRF}[A, E_{B,E}]$ is negl.) while we are using a secure PRP. Then in the case of AES-Ctr with key size 128 bit our encryption approach is secure when after 2^{32} ciphertexts of length 2^{32} , we change the key, the total number of AES block is 2^{64}, as it is shown in the following equation.

$$|X| = 2^{128} \Rightarrow qL^{1/2} < 2^{48}$$

Again the above security analysis is for RND in which we use a random IV. In the case of DET (using **fixed-IV**) the AES-Ctr cannot be IND-DCPA secure, even if we force the assumption that we encrypt less than 16 bytes which means only 1 AES block. We propose the following attack against the AES Ctr-mode.

Insecure Counter-mode with constant IV ($|m| > 16$ bytes):
We play the IND-DCPA security game:
1. The attacker outputs m_* ,m_*
2. The encryptor sends to attacker $c_1 \leftarrow m_* \oplus F(k, FIV)$
3. The attacker outputs m_0,m_1
4. The encryptor sends to attacker $c_2 \leftarrow m_b \oplus F(k, FIV)$ Then the attacker outputs b=0 if $c_2[1] \oplus c_1[1] = m_* \oplus m_0$ with $ADV_{DCPA}[A, E_{CBC}] = 1$.

6.2 Appendix B

Mirror property analysis: Let n all the possible answers that can be observed by an attacker. Then, all the different ways of arranging the n answers into a sequence (all the possible permutations) are factorial of n (n!). Lets assume that the attacker has observed all the possible range queries that have been executed, without taking into account the length of the range in the query. Firstly, we prove that in case of a query, where the upper and lower bound of the range differ by one, then the possible permutations decrease from n! to 2. Furthermore, we will prove that the execution of range queries with a range greater than 2 (3,4,...,n-1,n) does not provide more information about the ordering.

Assume that $t_1, t_2, ..., t_{n-1}, t_n$ are the possible answers. All the possible answers of range 2 are $t_1 \leftrightarrow t_2, t_2 \leftrightarrow t_3, ..., t_{n-2} \leftrightarrow t_{n-1}, t_{n-1} \leftrightarrow t_n$. The number of these answers is exactly n-1. The symbol "$\leftrightarrow$" implies that the attacker can only observe neighboring relations, e.g. $t_1 \leftrightarrow t_2$ means that t_1 and t_2 are neighbors. However, the attacker can not decide if a is greater than b or vice versa. At this point, we reveal one by one all of the above pairs to the attacker and he tries to configure the in between relations (n! are all the possible permutations of the objects)

(Step 1:) Firstly, we know that the attacker will observe the relation ($t_1 \leftrightarrow t_2$) and that he will realize that t_1 and t_2 are neighbors. Therefore, we can cross out all combinations that do not consider t_1 and t_2 as neighbors. Immediately, the number of the remaining combinations is equal to 2*(n-1)!, since we can define the $t_1 \leftrightarrow t_2$ relation as t_x. At this point, the sequence that needs ordering has range n-1 ((n-1)! permutations)and we multiply the number of permutations with a factor 2, because although we know that t_1 and t_2 are neighbors, we still do not whether t_1 precedes t_2 or vice versa.
Consecutively the sequence becomes $t_1 \leftrightarrow t_2, t_3, t_4, ..., t_{n-1}, t_n$ and has range (n-1).

(Step 2:) Then, we know that the attacker will observe the ($t_2 \leftrightarrow t_3$) relation and therefore we can cross out all the combinations of the 2*(n-1)! permutation list, that were produced in the previous step and do not take into account the adjacency relation between $t2$ and t_3. Now,the $t_1 \leftrightarrow t_2 \leftrightarrow t_3$ relation is defined as t_x, which implies that the number of the remaining combinations is 2*(n-2)!, since the attacker is aware of $t_1 \leftrightarrow t2(1)$ and $t_2 \leftrightarrow t_3(2)$. Also, the relations (1) and (2) should be satisfied simultaneously and this happens whenever $t_1 \rightarrow t_2 \rightarrow t_3$ or $t_1 \leftarrow t_2 \leftarrow t_3$. In this step 2*(n-2)! permutations are produced, where factor 2 derives from the two previous relations. Consecutively the

sequence becomes $t_x, t_4, ..., t_{n-1}, t_n$ and has range (n-2).
(Step k:) Similarly, we can conclude that the number of permutations after step k is 2*(n-k)!
(Step n-1:) In this step, k = n-1 and therefore, the remaining permutations are 2. So the attacker cannot decide between $t_1 \rightarrow t_2 \rightarrow, ..., \rightarrow t_{n-1} \rightarrow t_n$ and $t_1 \leftarrow t_2 \leftarrow ... \leftarrow t_{n-1} \leftarrow t_n$.
A step n does not exist, because assuming that we have a domain n, then the possible range queries that can occur for range 2 are n-1.
Point queries do not reveal adjacency relations to the attacker, since there are no permutations (the number of possible permutations equals to one). Hence, the user cannot use the above information to his advantage and the number of all possible sequences is n! sequences.

We have shown that range queries of range 2, can reduce the initial n! number of permutations to 2 and although the attacker is aware of the tuples' partial order, he still cannot decide whether the original or the reversed sequence was sent. Queries of range m, where $m > 2$ provide less or equal uncertainty compared to the respective queries having a range $m = 2$, since every query of range m can be decomposed to $m - 1$ queries of length 2. Hence, the system is not more susceptible to the attacker in this case.

At this point we need to mention that the mirroring property appears in any Precise Query Protocol and if the attacker observes all the possible queries, then he can reach to a conclusion about the partial ordering of the objects or the ciphertexts that we have set.

References

[1] Boldyreva, A., Chenette, N., Lee, Y., O Neill, A.: Order-preserving symmetric encryption. In: Advances in Cryptology-EUROCRYPT 2009. Number *. Springer (2009) 224–241 14, 15, 28, 29, 31, 58

[2] Popa, R.A., Li, F.H., Zeldovich, N.: An ideal-security protocol for order-preserving encoding. IACR Cryptology ePrint Archive **2013**(*) (2013) 129 14, 28, 29, 30, 58, 61

[3] 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 15, 29

[4] Boldyreva, A., Chenette, N., O'Neill, A.: Order-preserving encryption revisited: improved security analysis and alternative solutions. In: Advances in Cryptology–CRYPTO 2011. Number *. Springer (2011) 578–595 15, 25, 29

[5] Xiao, L., Yen, I.L.: Security analysis for order preserving encryption schemes. In: Information Sciences and Systems (CISS), 2012 46th Annual Conference on. Number *, IEEE (2012) 1–6 15

[6] Goldreich, O.: Foundation of cryptography fragments of a book. february 1995. (1998) 17, 18

[7] Rusu, F.I.: Sketches for aggregate estimations over data streams. PhD thesis, University of Florida (2009) 21

[8] Agrawal, D., Abbadi, A., Wang, S.: Secure and privacy preserving database services in the cloud. In: Data Engineering (ICDE), 2013 IEEE 29th International Conference on, IEEE (2013) 1268–1271 23

[9] 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 25, 37

[10] 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. Number *, VLDB Endowment (2013) 289–300 25, 37

[11] Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009) 27

[12] Gentry, C.: Computing arbitrary functions of encrypted data. Communications of the ACM **53**(3) (2010) 97–105 27

[13] Özsoyoglu, G., Singer, D.A., Chung, S.S.: Anti-tamper databases: Querying encrypted databases. In: DBSec. Number * (2003) 133–146 29

[14] 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. Number *, ACM (2004) 563–574 29

[15] 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. Number *, IEEE (2009) 1709–1716 29

[16] 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 29

[17] 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 29

[18] Kadhem, H., Amagasa, T., Kitagawa, H.: A secure and efficient order preserving encryption scheme for relational databases. In: KMIS. Number * (2010) 25–35 29

[19] Xiao, L., Yen, I.L., Huynh, D.T.: Extending order preserving encryption for multi-user systems. IACR Cryptology ePrint Archive **2012** (2012) 192 29

[20] 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 29

[21] 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 29

[22] Ang, G.W., Woelfel, J.H., Woloszyn, T.P.: System and method of sort-order preserving tokenization (2012) US Patent 20,120,278,897. 29

[23] Liu, D., Wang, S.: Nonlinear order preserving index for encrypted database query in service cloud environments. Concurrency and Computation: Practice and Experience (*) (2013) 29

[24] 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 32, 34

[25] 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 32

[26] Agrawal, D., El Abbadi, A., Emekci, F., Metwally, A., Wang, S.: Secure data management service on cloud computing infrastructures. In: New Frontiers in Information and Software as Services. Springer (2011) 57–80 35, 36, 37

[27] Emekci, F., Agrawal, D., Abbadi, A.E., Gulbeden, A.: Privacy preserving query processing using third parties. In: Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on, IEEE (2006) 27–27 35

[28] 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 37

[29] Arasu, A., Blanas, S., Eguro, K., Kaushik, R., Kossmann, D., Ramamurthy, R., Venkatesan, R.: Orthogonal security with cipherbase. In: CIDR. (2013) 37

[30] Rogaway, P., Shrimpton, T.: The siv mode of operation for deterministic authenticated-encryption (key wrap) and misuse-resistant nonce-based authenticated-encryption. Unpublished specification document corresponding to the above (*) (2007) 44

[31] Halevi, S., Rogaway, P.: A tweakable enciphering mode. (*) (2003) 44

[32] Halevi, S., Rogaway, P.: A parallelizable enciphering mode. In: Topics in Cryptology–CT-RSA 2004. Springer (2004) 292–304 44

[33] Martin, L.: Xts: A mode of aes for encrypting hard disks. Security & Privacy, IEEE **8**(3) (2010) 68–69 44

[34] 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 44

[35] Pinkas, B., Reinman, T.: Oblivious ram revisited. In: Advances in Cryptology–CRYPTO 2010. Springer (2010) 502–519 44