PRIVACY PRESERVING DATABASE EXTERNAL LAYER CONSTRUCTION ALGORITHM VIA SECURE DECOMPOSITION FOR ATTRIBUTE-BASED SECURITY POLICIES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

Uğur Turan

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

MARCH 2018

Approval of the thesis:

PRIVACY PRESERVING DATABASE EXTERNAL LAYER CONSTRUCTION ALGORITHM VIA SECURE DECOMPOSITION FOR ATTRIBUTE-BASED SECURITY POLICIES

submitted by **Uğur Turan** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Halit Oğuztüzün _____
Head of Department, **Computer Engineering**

Prof. Dr. İsmail Hakkı Toroslu _____
Supervisor, **Computer Engineering Department, METU**

Prof. Dr. Murat Kantarcıoğlu _____
Co-supervisor, **Computer Science Dept., University of Texas at Dallas**

Examining Committee Members:

Prof. Dr. Özgür Ulusoy
Computer Engineering Department, Bilkent University _____

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _____

Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _____

Assoc. Prof. Dr. Osman Abul
Computer Engineering Department, TOBB ETÜ _____

Assoc. Prof. Dr. İsmail Sengör Altıngövde
Computer Engineering Department, METU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Uğur Turan

Signature :

ABSTRACT

PRIVACY PRESERVING DATABASE EXTERNAL LAYER CONSTRUCTION ALGORITHM VIA SECURE DECOMPOSITION FOR ATTRIBUTE-BASED SECURITY POLICIES

Turan, Uğur
Ph.D., Department of Computer Engineering
Supervisor : Prof. Dr. İsmail Hakkı Toroslu
Co-Supervisor : Prof. Dr. Murat Kantarcıoğlu

March 2018, 88 pages

Relational DBMS's continue to dominate the market and inference problem on external schema has preserved its importance in terms of data privacy. Especially for the last 10 years, external schema construction for application-specific database usage has increased its independency from the conceptual schema, as the definitions and implementations of views and procedures have been optimized. After defining all mathematical background, this work offers an optimized decomposition strategy for the external schema, which concentrates on the privacy policy and required associations of attributes for the intended user role. The method given in this article performs a proactive decomposition for the external schema, which satisfies the inhibited and required associations of attributes. The idea is represented by using graph theory (using attribute-sets as vertices and dependencies as edges) and we redefine the problem of inference like a shared root tree finding process in between related attributes, using complete schema functional dependency graph. The optimization of decomposition aims to result in an external schema, which prevents inference of inhibited attribute sets and satisfies association of required attribute sets with minimal loss of associa-

tion between other attributes. Our technique is purely proactive like a normalization stage and owing to the usage independency of external schema construction tools, it can be easily applied to any ongoing systems without rewriting data access layer of applications. Our extensive experimental analysis shows the usage of this optimized proactive strategy offers applicable timing costs, even being proactive, for a wide portion of logical schema volumes. Additionally, we shared a real-life case study to emphasize the importance of using this strategy for privacy policy preservation during external schema definition and the observed benefits after getting this technique in production.

Keywords: Privacy Preserving Decomposition, Inference on Relational Databases, Attribute-Based Access Control

ÖZ

ALAN BAZLI GÜVENLİK İÇEREN VERİTABANLARI İÇİN GİZLİLİĞİ KORUYAN GÜVENLİ PARÇALAMA YÖNTEMİ İLE KULLANICI DIŞ KATMANININ OLUŞTURULMASI

Turan, Uğur
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. İsmail Hakkı Toroslu
Ortak Tez Yöneticisi : Prof. Dr. Murat Kantarcıoğlu

Mart 2018 , 88 sayfa

Günümüzde kullanımı giderek artan bağıntısal veritabanı mekanizmalarında, kullanıcıya sunulan şemadaki veri gizliliği ve güvenliği önem taşımaktadır. Son 10 yıl içerisinde, kullanıcı katmanı seviyesinde geliştirilen araçlar sayesinde, bu katman diğer alt katmanlardan çok daha bağımsız hale gelmiştir ve gizliliği arttırmak adına alan bazlı bir yaklaşım gerekli olmuştur. Bu çalışmada sunulan mekanizma ile veri tabanı kullanıcı katmanı güvenli biçimde parçalanmıştır. Bu parçalama esnasında en az sayıda mantıksal bağ kaybedilmesi ve kullanıcının ihtiyacı olan erişim kümeleri de dikkate alınmıştır. Yapılan deneyler sayesinde, bu yöntemin veri tabanının oluşturulması esnasında çalışması sayesinde, dinamik çalışan mekanizmalara göre çok daha etkili ve kolay kullanıma sahip olduğu kanıtlanmıştır. Çalışma içerisinde, bu yöntemin uygulanması gerekliliğini vurgulayan ve uygulama sonrası durumun da gözlemlendiği bir gerçek hayat örneği paylaşılmıştır.

Anahtar Kelimeler: Gizliliği Koruyarak Parçalama, Bağıntısal Veritabanlarında Mantıksal Çıkarım, Sütun Bazlı Erişim Kontrol

For My Beautiful Mother...

ACKNOWLEDGEMENTS

I would like to thank my supervisor İsmail Hakkı Toroslu, my co-supervisor Murat Kantarcıoğlu and all thesis committee members for all their support and guidance during this work. It was a great honor for me to work with them.

Nothing would be so beatiful and honorful in my life, if my wife Pelin Turan did not love me.

I hereby welcome my new handsome man, Emin Kaya Turan.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

DBMS	Database Management System
NA	Not Applicable

CHAPTER 1

INTRODUCTION

The evolution of technology in business applications has increased the importance of protecting sensitive data, that needs to be accessed by many different users with different access privileges. Fine grained inference control has increasingly become crucial to prevent unwanted sensitive information inferences from the data disclosed to legitimate users. In relational database management systems, usually views have been used for this purpose. In order to satisfy the given privacy policies, in view based solutions, basically direct inferences of sensitive data is prevented without considering existence of the functional and probabilistic relationships among the attributes in the database schema. These approaches have very simple purpose; that is, to determine whether to grant or deny the access, based on the predefined constraints related to the role of the user. Especially, for the attribute based access control, essentially the attributes that are going to be related with each other in a query are used in making the grant or deny decision of the query. However, using functional and/or probabilistic dependencies, it might be possible to obtain sensitive relationships among attributes indirectly. The aim of this document is to propose a formal model to address this kind of sensitivity problem and describe a decomposition based solution to it.

In order to illustrate the problem of indirect inference of the relationships among security dependent attributes [1] through functional and probabilistic dependencies, consider a simple database application with an `EMPLOYEE` relation. Assume that the company would like to adjust the salaries of its employees based on their departments, positions and years of experiences according to market standards. The `EMPLOYEE` relation has the following schema:

[1] i.e., a set of attributes in which the association among corresponding values are sensitive and should not be obtained

EMPLOYEE = (*id*, *phoneNumber*, name, gender, salary,

department, position, yearsOfExperience)

In order to eliminate the discriminative effect of the gender of employees in determining the salaries, we would like to create an external view layer to the person who is responsible from this task, and, prevent this person to obtain salary and gender relationship. Also, assume that in this relation id and phoneNumber fields are two keys, and id is selected as the primary key. Therefore, probable join queries should be checked in order to guarantee that salary and gender fields cannot be related with each other by the help of these keys. In addition, we have assumed that there is no functional dependency other than the ones which make id and phoneNumber candidate keys for the EMPLOYEE relation.

A natural solution to this problem is basically decomposing the EMPLOYEE relation into two views as follows:

$EMPLOYEE_1$ = (*id*, name, gender, department, position,

phoneNumber, yearsOfExperience)

$EMPLOYEE_2$ = (*id*, name, salary, department, position,

phoneNumber, yearsOfExperience)

However, this decomposition is incorrect since salary and gender relationship can be generated using the following query:

```
SELECT e1.gender, e2.salary
FROM EMPLOYEE1 e1, EMPLOYEE2 e2
WHERE e1.id = e2.id
```

To prevent this kind of join queries on keys, a further decomposition can be done:

$EMPLOYEE_1$ = (*id*, name, department, position,
phoneNumber, yearsOfExperience)

$EMPLOYEE_2$ = (name, salary, department, position,
yearsOfExperience)

$EMPLOYEE_3$ = (name, gender, department, position,
yearsOfExperience)

This decomposition may initially seem to be correct as it is not possible to join $EMPLOYEE_2$ and $EMPLOYEE_2$ using the given keys in order to relate `salary` and `gender` fields. However, `name` and `salary` fields appear together in $EMPLOYEE_2$ relation, and it is possible to infer `gender` from the `name` of an employee with high probability. Thus, in a correct solution, the possibility of relating `salary` with `name` should also be prevented as in the following decomposition:

$EMPLOYEE_1$ = (*id*, name, department, position,*phoneNumber*,
yearsOfExperience)

$EMPLOYEE_2$ = (salary, department, position,
yearsOfExperience)

$EMPLOYEE_3$ = (name, gender, department, position,
yearsOfExperience)

One of the most important subject for this example is, breaking the dependencies of

gender and salary attributes to the keys. Although, it looks like this decomposition may not be needed to satisfy the desired privacy policy, consider the following situation:

$EMPLOYEE_1$ = (*id*, name, department, position, *phoneNumber*, yearsOfExperience)

$EMPLOYEE_2$ = (*id*, salary, department, position, yearsOfExperience)

$EMPLOYEE_3$ = (name, gender, department, position, yearsOfExperience, *phoneNumber*)

Notice that in the above decomposition, the dependencies of gender and salary attributes from all the keys are broken since, if the id attribute was not removed from $EMPLOYEE_2$ relation, an equijoin operation on the key id field can be used to relate salary and gender attributes. Although for this example having keys in $EMPLOYEE_2$ and in $EMPLOYEE_3$ relations will always violate the given privacy policy, in general, some other key distribution alternatives may be possible for the decomposed relations, and this issue will be discussed later in the document.

By using the schema obtained with the last decomposition, it is not possible to define any query which relates salary and gender attributes neither through equijoins on keys, nor through using the given probabilistic dependencies. Therefore, this decomposition satisfies the given security policy. Also note that there may be keyless relations obtained after the decomposition, as it is usual in views, and this issue will also be discussed in the following sections of the document.

As in this example, a view based solution can be generated to satisfy a given privacy policy. There can be several policy rules, and views should be constructed in order to satisfy all constraints of these policies. The need for defining different external layers for different inference control policies has increased by web based data sharing

trend [1]. Therefore, a formal approach is needed to build a secure external layer by decomposing the relations into sub relations according to privacy policy rules and to generate relevant secure logical schema. These sub-relations can be used to generate accessible views for the user. Notice that, if any attribute other than candidate keys, such as position in the example, can be used as a key due to data distribution; then, this attribute should be stated as a candidate key or should be perturbed. Most likely, this pseudo-key situation may happen either at the beginning or may occur after data is inserted to the schema. In order to solve this issue, pseudo-key can easily be added as a key to the system, and decomposition can be processed again as the external layer is consisting of views only.

Most of the research addressing inference control problem is mainly focused on dynamic mechanisms employing query investigation or modification methods, and by also tracking the query history [2, 3, 4, 5, 6, 7]. On the other hand, our strategy in this document is to decompose the relation into views in advance, for preventing the time spent by costly query modification or history tracking operations [8]. To the best of our knowledge, this is the first attempt in the literature to handle privacy for context dependent attribute based inference control using a proactive approach. The approach is labeled as ”proactive”, since it does not need to perform costly for query history investigation, query modification, or row/attribute based joins or calculations as in a reactive strategy. Another disadvantage of a reactive strategy is about the user satisfaction in the system. Basically, in a reactive strategy, the queries that can be issued by the user will dynamically change based on their usage history. Therefore, two users with the same job titles may not be able to issue the same queries if their query history is different. This could create usability problems from a regular user point of view. In addition to those, our method can be easily adapted for validating existing external schema against given attribute based policy rules. The proactive decomposition method described in this document can easily be combined with other constraints such as row based policy rules during implementation.

Moreover, as the demand towards automated systems and processes have increased, the evolution of technology in business applications have focused on to two different usages: application usage and statistical analysis. In each usage, inference based privacy preserving techniques, in terms of databases, have been an important topic to

protect the sensitive data. Modern approaches like differential privacy[9] preserving techniques[10] or intentionally deception mechanisms provide secure ways to represent statistical results without revealing sensitive data, however their usage cannot be applied on traditional applications [11]. Many business applications aim to monitor and update single entity data, a brief example can be given as a call center module of a bank. Let a customer wishes to apply for a campaign and the operator should check her transaction history for prerequisites. Transactions are sensitive data, but they cannot be altered by adding noise or any hypothetical rows cannot be added for deception, the financial transactions should be viewed as they are. If the domain were a statistical analysis on transactions, then both techniques could be applied to protect inference mechanisms on sensitive data, but these kinds of usages are mainly based on a single entity row business procedure, named application usage. From this perspective, the call center user should access to a set of sensitive data according to assigned user role and the point is, external layer of database presented to this user role should not reveal any more information other than required. This objective needs three different perspectives. Firstly, the schema of external layer should be decomposed with a fine-grained attribute-based approach which preserves required associations for the user role and prevents any other inferences. This objective is the focus of this article, as necessary theorems and algorithms is proposed in this document. The second perspective deals with inferences based on dynamic data distribution and the last is about collaboration attacks [12, 13], in order to satisfy privacy of sensitive data. The mechanisms for the last two inference channels, can be applied as important add-ons to the strategy given in this document, however the main and first objective should be arranging the external layer for a specific user to prevent unwanted inference channels. This step is proactive, and it should be viewed as a policy-based normalization stage in terms of privacy. This document focuses on this subject and gives a complete mechanism to satisfy the goal. The given algorithms are based on Functional Dependency Graph representation of database schema, which is constructed by arranging attribute-sets as vertices and dependencies as edges. The aim is to find an external layer decomposition which strictly allows the required attribute associations and prevents inhibited associations, both in compliance with the privacy policy for a specific user role. Owing to the nature of domain, the mechanism is based on attribute based granularity and the advantage of rearranging external layer without making a change

on other layers, by the help of views, is used.

There may be different approaches for this kind of secure decomposition, intended in this document. The decomposition may be based on prevented attribute sets or required attributes sets for the user role may be focused and no other information may be leaked [14]. The strategy given in this document is an optimized combination of these two approaches and the most crucial step proposed in this document, is to check the required sets with prevented sets. This control step assures a fully compliant policy proactively. Moreover, the decomposition should not be much more lossy then needed. This fact is a critical optimization problem and to the best of our knowledge, this is the first attempt in literature to construct a secure decomposition satisfying the policy, by minimizing the dependency loss afterwards.

It should be stated that, if the external layer will allow only the required queries of user roles, then set of stored procedures may be adequate and there will be no need for a decomposition. However, the set of stored procedures should be checked for the forbidden sets for verification and there is again a need for a formal mechanism, at least for this control.

In this document, the first problem is stated as decomposing the logical schema according to the given constraints. Secondly, the solution is optimized as the number of lost dependencies after the decomposition is minimized up to a degree. Lastly, the problem is extended by defining required attribute sets and decomposition is performed accordingly. The first part of the document uses first-order predicate logic, where as graph-based modelling is selected for the other two parts as it is more suitable for this kind of optimization.

CHAPTER 2

A REAL-LIFE CASE

In this section, a real-life case study of the idea is given with the requirements, the situation, the solution and its results. The overall story is simplified for a better understanding. The project is about recycling in a municipality of Antalya, Turkey, completed by a software company. The product is about a web and point-of-sale application in which all citizens have smart cards and they give their recycling waste to the waste-collector companies and these companies load credits to their card according to the current expected market value of waste, grouped by type of waste. The product has been used since August 2015 and the web application has different modules for the collector companies, citizens and town management. In 2016, a citizen proceeded against the system and declared that she has been individually identified by other roles, as she is always getting messages in consistent with her consumption within specific periods of the year. It may be claimed that many citizens may have nearly same consumption and thoroughly, waste statistics distributed within a year, but the case is different as the claimant owns a hand-made gift company and has much more glass waste in November during preparation of gifts for new-year's day. Afterwards, the system has been checked for the information leak to the collector companies and town management. Collector companies should only query the time-based collection statistics of the cars and the town management should only view the usage statistics of the system with totals. Additionally, collector's views are defined only as a subset of the town management external view set.

For a simplified description, the views can be simplified as:

[Available for the Collector and Town Management]

$View_1$ = (CarId, DateTime, GPSCoordinates, TotalWasteWeight, WasteWeight, WasteType)

[Available for only Town Management]

$View_2$ = (CitizenId, Name, Surname, Address, PhoneNumber)

The problem is that, a malicious worker in town management can use GPSCoordinates attribute in association with the Address attribute and find a small subset of citizens which have given glass waste so much in a period of year. This malicious worker has shared this knowledge with an advertisement company and they used this information in favor of their glass-producer clients. It is not surprising to get messages from other glass-producer clients while you are managing a company which always purchases glass for gifts, but surprisingly this small gift company (the claimant) are a part of a charity association and they use glasses they have collected from their members all year. As a result, the privacy of the citizen has been violated. The main reason behind the problem is the lack of security policy while creating views and the attribute association based cross-control in between views, which should be made proactively. This point is the focus of this article and if the system designer's have used the strategy proposed in this document, they will not face with such a problem, caused by external layer structure. As a solution, all security dependent sets have been formed for the original database schema and the algorithms proposed in this work have been applied. It should be noted that the relationship between GPSCoordinates and Address attributes are a kind of probabilistic dependency and as a result these views are generated:

New $View_1$ = (CarId, DateTime, TotalWasteWeight, WasteWeight, WasteType)

New $View_2$ = (CarId, GPSCoordinates)

New $View_3$ = (CitizenId, Name, Surname, Address, PhoneNumber)

(Any join between New $View_1$ and New $View_2$ is not a meaningful join as CarId

is only foreign key, not a primary or candidate key, in these relations)

The structural external layer schema problem has been omitted by this way. The problem may only occur according to the data distribution, which is out of scope of this work and different strategies (discussed in Introduction) may be used additionally.

It should be noted that, there exist many tested and deployed systems in production when this solution is applied. Because of this reason, a new view is created by using $\texttt{New View}_1$ and $\texttt{New View}_2$ to generate $\texttt{View}_1$ for the users. By this strategy, any code change is not necessary for adaption as everything is solved on database layer. As materialized views are used in the database, the performance degradation may be neglected (Nearly 10K rows for a single table in a year, only less than a second change in report query duration)

As a result, the strategy of developing privacy preserving relational database external schema, given in this thesis is very useful and easy-to-operate, not to experience such surprising privacy deficiencies.

CHAPTER 3

BACKGROUND RELATED WORK

The field of database security is very popular, and several works in this field have influenced the idea proposed in this document [2, 15, 3, 16, 4, 5, 17, 18, 6, 7, 19, 20, 21, 22, 12, 23, 24, 25, 26, 27, 13, 28, 29, 30, 31, 32, 33, 34, 35, 36]. The approach of updating the query dynamically depending on the context and the policy has been studied for a long time in the literature [4]. In this method, the query can be modified by adding predicates and the main purpose is the row based security. Adding more predicates to `where` clause can only restrict the rows extracted by the query [4]. Actually the security mechanism in [4] gives user a set of views which are permitted to be queried and then performs row based elimination by adding predicates whose idea can be treated as an additional functionality for the work in this document. However another work, [5], states that the former algorithm is not maximal and limits some permitted answers. In [5] some flexibility has been added as the query may depend on any view or sub view or meta-relations. That means extra work should be conducted in order to find which permitted views are involved. These two approaches may have performance problems and modifying the query can be costly [8], nevertheless it should be noted that their query modification strategies are done mainly for row filtering, whereas this document focuses on context dependent attribute based inference control in a proactive manner.

In addition to this, Oracle introduces Virtual Private Database [7] and performs the security totally by query modification on original relations. The modification can be as row based by adding predicates or column based by making null of the unwanted attributes. Bertino [18] calls this type of query modification approaches as “Truman Models” [17], since they answer each time, nevertheless the answer may not be max-

imal because of restrictions. These models have simple attribute based policy rules as just checking the existence of attributes in the query result. Beside this, data perturbation [22] is another run-time consuming method and may be used for Truman Models. In addition to that, “k-anonymity” [3] has been proposed to divide the relation to views which are targeted not to extract "*id*"s. Moreover the security policy does not have to satisfy the anonymity only. For instance, one can define a policy rule as *gender* and *address* should not be obtained together even though neither of them is adequate for identification [21].

Furthermore, Purpose Oriented Access Control scheme [15] offers role - purpose - column mapping, however two purposes may serve to another unwanted purpose. For instance let `a`, `b`, `c` to be attributes and purpose-1 needs `a`, `b`; purpose-2 needs `a`, `c` and non-existing and unwanted purpose-3 needs `b`, `c`. In this example first two purposes can serve to the unwanted third purpose. That example presents the notion of query history [12] whose deep investigation makes the computation costly. To get rid of these, attribute mutability term [16] provides a mechanism as Chinese-Wall method [19] with historical data, but performance requirements may be critical.

Beside this, “Non-Truman” models have been proposed [17] which reject the unauthorized query according to the authorized views. Hippocratic databases [6] combine many security issues stated in this section, however the addressed problem in this document is a bit different. In all these works, the main problem is to maintain security and privacy; nevertheless, dynamic security modeling with query modification, attribute mutation, historical query tracking or grant/reject mechanism may have performance problems because of their run time executions. These performance problems may become vital for generally used simple queries. This document constructs a proactive security mechanism as building an external layer with a secure logical schema to user by a decomposition algorithm in which user is free to query anything on decomposed relations. The term “Attribute-Based” in this document is used for the ability of defining the inference control rules on the attributes of relations. The same term has been used differently in [25] to build the access control with the help of dedicated attributes. It is important to note that the notion of modeling access control rules on attributes according to the application semantics is another important work discussed in [11] which is not in the scope of this document. The discussion in

[11] can be used for defining security dependent sets which is used by decomposition process in this document.

A relevant study, targeted a similar problem with this work, is reducing inference control to access control [2]. However their solution labels the normalized schema relations and the solution is not proactive, only more efficient than query controlling. Furthermore, another highly relevant work can be given as [37], as the method in that document can be used to store decomposed relations in different parties while preserving privacy of sensitive data against those parties. The document [37] aims to reconstruct original database from decomposed relations while preserving privacy, whereas our document aims to decompose external layer to views in such a way that, the user has access to all views in related external schema but is not able to reconstruct the database in order to access the protected sensitive data.

Database security and inference problem has been very popular as several works in this field have helped to construct the strategy given in this document. Inference problem has been discussed in many documents [16, 4, 18, 6, 26] but most of them are about reactive solutions. The approaches should be classified into two groups for different usage scenarios.

3.1 Statistical Usage

This kind of usage includes query rewriting mechanisms [17], data perturbation methods [7, 22], deception strategies[38] and decomposition-based approaches[39, 40] for satisfying privacy. The basis of main data perturbation methods is “Differential Privacy” and main researches on this field try to build a noise which prevents identification but outputs meaningful data. The strategy is a big step in literature to prevent inference attacks, however its usage is limited to the statistical analysis. Another approach “deception” mechanisms aims to corrupt the data by inserting anonymous data or structures, however the resulting behavior applicability is not much more different from the “Differential Privacy”. K-anonymity [3] is a leading research on this field but differential privacy has proven the hardness of satisfying “non-identifiability” problem[41] according to dynamic data distribution.

All these techniques aim to prevent the identifiability of the individuals. The target user is assumed to query the database to infer the statistical information about the data, therefore any mechanism which changes real data can be used if the resulting statistics reflect the natural distribution of the original set. However, these kind of approaches cannot be used for application usage. This restriction can be illustrated with a simple example:

Let a clerk in a hotel reservation system checks if a customer is above 18 years old or not. If differential privacy is applied here with a sample noise, then the age of the customer can be outputted as 10 or 40, which is a useless result for the user.

Beside this, the nature of the problem is different for statistical databases. Application usage tends to query the database upon the single row identification, which is the opposite of statistical approach. Therefore, these techniques cannot be used in application usage. This thesis concentrates only on application usage.

3.2 Application Usage

Application usage of database strictly needs single-row identification with real values (as described in Introduction). For this usage, the method can be divided as being reactive or proactive. Reactive methods tend to behave dynamically according to the policy or data distribution. Query rewriting techniques (as in Truman and Non-Truman Models called by Elisa Bertino) are reactive solutions to the inference problem. Query history tracking mechanisms and Chinese-Wall method [9] like approaches are subject to performance issues, because of being reactive. It is a major step to check purpose [15] of the user during privacy protection. During these checks, attribute-based granularity [8] should be used to preserve precise privacy. The idea proposed in this document is totally proactive, such a normalization process, and ready to welcome any reactive method to construct a complete mechanism. Database security policy should be checked against visibility [5] requirements and the external layer should be constructed accordingly.

This document states a complete, optimized and applicable decomposition strategy compared to [39] and [40]. The aim of this document is to perform the decomposi-

tion with policy check, minimal loss of dependencies and by taking care of indirect dependencies. The related works propose an effective way of decomposition database in a somehow similar manner, nevertheless this document combines maximal availability, intended privacy, policy check and indirect dependencies to carry out a definite decomposition for the external layer.

CHAPTER 4

PROBLEM DEFINITION AND SOLUTION WITH STRONG-CUT ALGORITHM

4.1 Preliminaries and Problem Definition

In this section, we give the basic terms and concepts used in the chapter. This chapter has two main objectives, namely, formally defining a secure logical schema which is in compliance with the given security constraints (security dependent sets), and developing a decomposition algorithm which divides relations into sub-relations to be able to satisfy the security constraints. The main reason for decomposition is to prevent obtaining securely dependent attributes together directly in a relation or through a join.

Therefore, first, the definition of the logical schema is given in terms of three sets as relational schema, probabilistic, and (non-reflexive and non-partial) functional dependencies. After that, the closures of relations and functional dependencies with the given rules, are defined. The closure of relation schema is very important, since it describes how new relations can be generated using only equijoins on foreign keys. Moreover, the closure of functional dependencies is used to define identifiers for attributes, how they can be inferred, and how two or more attributes can be associated with each other. Combining these definitions, we then define a secure logical schema, which simply prevents obtaining the attributes of each given security dependent set together by joins. We also prove that secure logical schema guarantees that it is not possible to obtain any association among the set of attributes of security dependent set.

Following these, we define a decomposition operation which decomposes a logical

schema according to a given secure dependent sets in order to form a secure logical schema. Afterwards, we prove that the new schema obtained by employing the decomposition operation is a secure logical schema, which means that it is not possible to associate attributes of secure dependent sets by joins using the relations constructed after the decomposition. By this way, the inference of association of the attributes together in each security dependent set can be prevented.

Definition 1. ***Relational Schema*** *A relation schema is defined a set of attribute names concatenated with relation name (using underscore) in order to prevent the vagueness caused by having same attribute name in different relations. For the sake of simplicity, relation schema is referred as relation and the concatenation on attribute names will not be shown unless needed.*

For example a relation

`USERS={`*`id_users`*`,name_users,surname_users,email_users}`

is defined as a set of concatenated attribute names. Using this definition, it is guaranteed that all attribute names in a database will be unique owing to unique relation names by default.

Definition 2. ***Logical Schema*** *A logical schema for a database is defined as a 3-tuple* $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$ *such as;*

- $\mathcal{R}$ *is defined as set of all relation schemas in a database.*
- $\mathcal{F}$ *is defined as set of functional dependencies among attributes in all relation schemas in* $\mathcal{R}$ *excluding reflexive and partial functional dependencies. It is important to note that there exists a functional dependency between a key and its occurence in another relation as foreign key owing to the reflexive property.*
- $\mathcal{P}$ *is defined as probabilistic dependencies (shown as* $\mapsto$*) which includes the attribute pairs as guessing the value of the dependent attribute is possible by the help of determining attribute. Reflexive probabilistic dependencies are trivial and omitted.*

Since a foreign and relevant key in two different relations have different names according to the definition of relational schema, the functional dependency in between them is not treated as reflexive and should be in $\mathcal{F}$ as given in the example below.

It should be noted that this kind of functional dependencies have a special importance since they are used to perfom equijoins on keys while inference, which will be discussed below.

For example,

$\mathcal{R} = \{USERS, LOGS\}$ as

`USERS = {`*`id_users`*`, name_users, gender_users, surname_users}`

`LOGS = {`*`userid_logs`*`, action_logs, date_logs}`

`userid_logs` is a foreign key referencing `USERS(id_users)`.

$$\mathcal{F} = \left\{ \begin{array}{l} (\boldsymbol{userid\_logs \rightarrow id\_users}), \\ (\boldsymbol{id\_users \rightarrow userid\_logs}), \\ (id\_users \rightarrow name\_users), \\ (id\_users \rightarrow surname\_users), \\ (userid\_logs \rightarrow action\_logs), \\ (userid\_logs \rightarrow date\_logs) \end{array} \right\}$$

$$\mathcal{P} = \Big\{ (name\_users \mapsto gender\_users) \Big\}$$

The functional dependencies given in bold expresses the dependency between original key and a foreign key. In addition to that, the probabilistic dependency is given for the `gender` to the `name`, as it is likely to guess the `gender` of a user by knowing the value of `name` attribute.

Using the join operation, the new relations can be obtained from the relations of logical schema. Similarly, the properties of functional dependencies can be used to generate new functional dependencies from the existing ones. Below, we define two

closures for these two.

Definition 3. $\mathcal{F}^+$***: Closure of*** $\mathcal{F}$ *Closure set of given* $\mathcal{F}$ *that can be obtained by using the transition, union and decomposition properties of functional dependencies [42], excluding reflexive and partial functional dependencies.*

Definition 4. $\mathcal{R}^+$***: Closure of*** $\mathcal{R}$ *Closure of* $\mathcal{R}$ *is defined as* $\mathcal{R}^+$*, composing all probable relation schemas obtained by performing any database query over* $\mathcal{R}$ *for which join operations used in the query should only be equijoins on keys and related foreign keys (named* ***meaningful join*** *thereafter) in order not to produce spurious tuples.*

For example, by using the sample decomposed relations given in introduction section:

`EMPLOYEE`$_1$ `= (`*`id`*`, name, department, position,` *`phoneNumber`*`, yearsOfExperience)`

`EMPLOYEE`$_2$ `= (salary, department, position, yearsOfExperience)`

the query which produces spurious tuples can be given as:

```
SELECT *

FROM EMPLOYEE1 e1, EMPLOYEE2 e2

WHERE e1.department = e2.department
```

The join operation is an equijoin but not on keys, therefore spurious tuples are generated by associating different employees working on the same `department`.

Let $\mathcal{U}_\mathcal{R}$ be union of all attributes existing in all relations of $\mathcal{R}$ (i.e., $\mathcal{U}_\mathcal{R} = \bigcup_{\mathcal{R}_i \in \mathcal{R}} \{ x \mid x \in \mathcal{R}_i \}$). Rather than defining which relations can be constructed from $\mathcal{R}$ as subsets of $\mathcal{U}_\mathcal{R}$, we can specify the set of attributes that cannot be obtained together in $\mathcal{R}^+$ as follows:

Property of $\mathcal{R}^+$: An attribute set $\mathcal{A}$ cannot be subset of any derived relation schema in $\mathcal{R}^+$, if and only if, $\mathcal{A}$ cannot be a subset of existing relation schema and cannot be functionally dependent to any set of attributes in $\mathcal{U}_\mathcal{R}$ so it becomes impossible to relate with equijoins on foreign keys by the definition of $\mathcal{R}^+$. Note that, according to

Definition 2 if there is a foreign key relationship, then it is represented as functional dependency as $((A_i \rightarrow A_j) \in \mathcal{F})$. Since all meaningful joins can only be executed using this kind of functional dependencies, the following logical formula expresses that a set of attributes $\mathcal{A}$ cannot be a subset of any relation in $\mathcal{R}^+$ if and only if, there is no functional dependency relationship to $\mathcal{A}$ in $\mathcal{F}^+$ and there is no relation containing $\mathcal{A}$.

$$\forall \mathcal{R}_k \in \mathcal{R}^+, \forall \mathcal{A} \subseteq \mathcal{U}_{\mathcal{R}}[\mathcal{A} \not\subseteq \mathcal{R}_k \Leftrightarrow \forall \mathcal{R}_j \in \mathcal{R}(\mathcal{A} \not\subseteq \mathcal{R}_j) \wedge \forall \mathcal{A}_i \subseteq \mathcal{U}_{\mathcal{R}}((\mathcal{A}_i \rightarrow \mathcal{A}) \notin \mathcal{F}^+)] \tag{4.1}$$

As it can be seen from above definitions, there is a strong condition which says that in order not to be able to obtain a subset of attributes from a logical schema it should not be possible to perform a meaningful join. In order to perform meaningful join; keys and related foreign keys are used, and they correspond to functional dependencies. Therefore, we need the following definitions to represent these relationships.

Definition 5. ***Set of Identifier Sets*** *The set of identifier set of an attribute α for a given $\mathcal{F}$, as $i_\alpha^{\mathcal{F}}$, is defined as follows:*

$$i_\alpha^{\mathcal{F}} = \left\{ x \;\middle|\; x \not\supseteq \{\alpha\} \wedge (x \rightarrow \alpha) \in \mathcal{F}^+ \right\} \tag{4.2}$$

Each element of $i_\alpha^{\mathcal{F}}$ is also called as *identifier set* of attibute α.

For example,

$$\alpha = name$$

$$\mathcal{F}^+ = \begin{Bmatrix} id \rightarrow name, \\ id \rightarrow surname, \\ id \rightarrow age, \\ id \rightarrow email, \\ email \rightarrow name, \\ email \rightarrow surname \end{Bmatrix}$$

$$i_\alpha^{\mathcal{F}} = \{\{id\}, \{email\}\}$$

The definition could be simply extended for an attribute set $\mathcal{A}$ as follows:

$$\mathcal{I}_{\mathcal{A}}^{\mathcal{F}} = \{\ x \mid x \not\supseteq \mathcal{A} \wedge (x \rightarrow \mathcal{A}) \in \mathcal{F}^{+}\ \} \tag{4.3}$$

These two definitions can be related as for an attribute set $\mathcal{A}$, as $\mathcal{I}_{\mathcal{A}}^{\mathcal{F}}$ contains the shared elements in $i_{\alpha}^{\mathcal{F}}$ for all α which are attributes in $\mathcal{A}$.

$$\mathcal{I}_{\mathcal{A}}^{\mathcal{F}} = \{\ x \mid \forall \alpha \in \mathcal{A}(x \in i_{\alpha}^{\mathcal{F}})\ \} \tag{4.4}$$

Identifiable Property: Each attribute of an *identifiable set* (i.e, ($\mathcal{I}_{\mathcal{A}}^{\mathcal{F}} \neq \emptyset$)), should be in the same relational schema with at least one of its identifier set. In other words, for a $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$;

$$\forall \mathcal{A} \subseteq \mathcal{U}_{\mathcal{R}}(\mathcal{I}_{\mathcal{A}}^{\mathcal{F}} \neq \emptyset \Leftrightarrow \forall \alpha \in \mathcal{A}, \exists \mathcal{R}_i \in \mathcal{R}, \exists \mathcal{D} \in i_{\alpha}^{\mathcal{F}}((\alpha \cup \mathcal{D}) \subseteq \mathcal{R}_i)) \tag{4.5}$$

The same property can be thought as it is impossible to identify an attribute in a logical schema if the attribute does not have any identifier set in its relation schema, which makes it impossible to discover other identifier sets by using meaningful joins. This issue is also a matter of database normalization however in this document, it is assumed that the relational schema is given as relations up to 3NF.

Definition 6. ***Inferability*** *A set of attributes $\mathcal{A}_1 \subseteq \mathcal{U}_{\mathcal{R}}$ can be inferable from a set of attributes $\mathcal{A}_2 \subseteq \mathcal{U}_{\mathcal{R}}$ for a given $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$, shown as $\mathcal{A}_1 \overset{\mathcal{F}}{\rightrightarrows} \mathcal{A}_2$, as defined below.*

$$\forall \mathcal{A}_1 \forall \mathcal{A}_2((\mathcal{A}_1 \overset{\mathcal{F}}{\rightrightarrows} \mathcal{A}_2) \Leftrightarrow ((\mathcal{A}_1 \rightarrow \mathcal{A}_2) \in \mathcal{F}^{+})) \tag{4.6}$$

The definition of inferability is given by using closure set of functional dependencies since relation based key constraints may not be adequate as there may be functional dependencies among non-prime attributes in a schema which is not normalized.

Definition 7. ***Inference of Association among a Set of Attributes*** *For a given $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$, the inference of association among a set of attributes $\mathcal{A} \subseteq \mathcal{U}_{\mathcal{R}}$,shown as $\mathcal{X}_{\mathcal{L}}(A)$, means that either $\mathcal{A}$ should be inferable from a subset of $\mathcal{U}_{\mathcal{R}}$ or be subset of any existing relation schema in order to be associated. More formally:*

$$\mathcal{X}_{\mathcal{L}}(A) \Leftrightarrow \exists \mathcal{A}_i \subseteq \mathcal{U}_{\mathcal{R}}(\mathcal{A}_i \overset{\mathcal{F}}{\rightrightarrows} \mathcal{A}) \vee \exists \mathcal{R}_i \in \mathcal{R}(\mathcal{A} \subseteq \mathcal{R}_i) \tag{4.7}$$

In this chapter, the set of attributes are defined as to have **security dependency** among them if the inference of association among them should be prevented. In addition to that, the purpose of this chapter is to inhibit the inference of association a subset of $\mathcal{U}_{\mathcal{R}}$ with at least two attributes (each named a **security dependent set** thereafter) for a logical schema $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$ by building a **secure logical schema**. It should be noted that security dependency sets cannot include the attributes which have probabilistic dependency together. If there is a probabilistic dependency as $\mathcal{A}_1 \mapsto \mathcal{A}_2$, then all security dependent sets containing $\mathcal{A}_2$(such as $\{\mathcal{A}_2, \mathcal{A}_3\}$), will have an additional version which has $\mathcal{A}_1$ instead of $\mathcal{A}_2$(as $\{\mathcal{A}_1, \mathcal{A}_3\}$). This step of inserting additional security dependent sets according to the probabilistic dependencies will be given later in the algorithm.

Definition 8. ***Secure Logical Schema*** *A secure logical schema is a logical schema $\mathcal{L}_{\mathcal{S}}^{sec} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$ such that for a set of security dependent sets $\mathcal{S}$, there should not be any relation in $\mathcal{R}^+$, containing the attributes of any set in $\mathcal{S}$. Formally:*

$$\mathcal{L}_{\mathcal{S}}^{sec} \Leftrightarrow \forall \mathcal{S}_i \in \mathcal{S}, \nexists \mathcal{R}_i \in \mathcal{R}^+(\mathcal{S}_i \subseteq \mathcal{R}_i) \tag{4.8}$$

It should be emphasized that by the definition of $\mathcal{R}^+$, only meaningful joins are taken into consideration as queries should only have equijoins on keys and related foreign keys and by this way spurious tuples cannot be generated.

By the definition of secure logical schema, it can be stated that the inference of association among attributes of each security dependent set is impossible with a secure logical schema since the attributes of any security dependent sets cannot be functionally dependent to any subset of attributes in logical schema (excluding reflexive and partial dependencies as given in Definition 2) or in the same relation as given as a theorem below.

Theorem 4.1.1. *The inference of association among attributes of each security dependent set cannot be performed in secure logical schemas; that is,*

$$\forall \mathcal{S}_i \in \mathcal{S}(\neg \mathcal{X}_{\mathcal{L}_{\mathcal{S}}^{sec}}(\mathcal{S}_i)) \tag{4.9}$$

Proof. The formal proof given in Appendix briefly states that in order to perform the disallowed inference, either the attributes should be in the same relation or a mean-

ingful join should be done for an inter relation inference as both cases are impossible because of secure logical schema definition. □

The next step is to define transformation of a logical schema to a secure logical schema for given security dependency sets.

Definition 9. ***Secure Decomposition*** *A secure decomposition is decomposition of* $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$ *according to the set of security dependency sets* $\mathcal{S}$ *to a new logical schema* $\mathcal{L}'_{\mathcal{S}} = (\mathcal{R}', \mathcal{F}', \mathcal{P})$ *, having the following features:*

1. *Any attribute should not be lost after decomposition. In other words:*

$$\mathcal{U}_{\mathcal{R}} = \mathcal{U}_{\mathcal{R}'} \tag{4.10}$$

2. *The new set of functional dependencies should be subset of existing set of functional dependencies as there can't be any new functional dependency moreover a loss in existing functional dependencies is expected to inhibit the inference of associations among the elements of security dependent sets.*

$$(\mathcal{F}' \subseteq \mathcal{F}) \wedge (\mathcal{F}'^{+} \subseteq \mathcal{F}^{+}) \tag{4.11}$$

3. *Any of the decomposed relations should not be a superset of any security dependent set.*

$$\forall \mathcal{S}_i \in \mathcal{S}, \nexists \mathcal{R}_i \in \mathcal{R}' (\mathcal{S}_i \subseteq \mathcal{R}_i) \tag{4.12}$$

4. *Any of the attributes in a security dependent set should not coexist in the same decomposed relation with any of its identifier set.*

$$\forall \mathcal{S}_i \in \mathcal{S}, \forall \mathcal{R}_i \in \mathcal{R}', \forall \sigma \in \mathcal{S}_i, \nexists \tau \in i_{\sigma}^{\mathcal{F}} ((\{\sigma\} \cup \tau) \subseteq \mathcal{R}_i) \tag{4.13}$$

The fourth property of secure decomposition is a strong requirement since it makes all the attributes in any security dependent set, uninferrable after the decomposition. It should be noted that this property is a requirement for a totally proactive solution. If any mechanism intends to have proactive and run time components together, then this requirement can be relaxed.

The aim of the secure decomposition is to transform a logical schema to a secure logical schema by the help of security dependency sets, which is given as a theorem below.

Theorem 4.1.2. *If $\mathcal{L}'_{\mathcal{S}} = (\mathcal{R}', \mathcal{F}', \mathcal{P})$ is the logical schema obtained after performing secure decomposition to $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$ with the set of security dependency sets $\mathcal{S}$, then $\mathcal{L}'_{\mathcal{S}}$ is a $\mathcal{L}^{sec}_{\mathcal{S}}$.*

Proof. The formal proof given in Appendix briefly states that in order not to be a secure logical schema, a security dependent set should be inferable or should be a part of an original relation. The former is impossible as the attributes of security dependent sets can not be in the same relation with any of their identifier sets by the fourth property of the definition of secure decomposition. It is also impossible for the latter due to the third property of secure decomposition. □

4.2 Decomposition Algorithm

The main purpose of the decomposition algorithm is to achieve the secure decomposition (Definition 9) which is defined as resulting in a secure logical schema. In order to satisfy the goal, it is clear that the elements of each security dependent set should not be in the same sub-relation obtained after the decomposition of original relations. Furthermore, it should not be possible to meaningfully join sub-relations containing securely dependent attributes separately. Below we define an algorithm which exhaustively generates all the subsets of the attributes of all relations and eliminates the ones that do not satisfy the conditions mentioned above. After that, it also eliminates redundant sub-relations.

Algorithm 1 Decomposition Algorithm

Require:
$\mathcal{L}$: logical schema as $(\mathcal{R}, \mathcal{F}, \mathcal{P})$,
$\mathcal{S}$: set of security dependent sets for $\mathcal{L}$
Ensure:
$\mathcal{M}_{\mathcal{R}}$: set of maximal subsets of $\mathcal{R}$ according to $\mathcal{S}$

begin
isSChanged = true //Step-1: Handle Probabilistic Dependencies
while isSChanged **do**
    isSChanged = false
    for each $\mathcal{A}_i \mapsto \mathcal{A}_j$ in $\mathcal{P}$ **do**
        for each $\mathcal{S}_i$ in $\mathcal{S}$ **do**
            if $\mathcal{A}_j \in \mathcal{S}_i$ **then**
                add $(\mathcal{S}_i \setminus \{\mathcal{A}_j\}) \cup \{\mathcal{A}_i\}$ to $\mathcal{S}$
                isSChanged = true
            end if
        end for
    end for
end while
$\mathcal{M}_{\mathcal{R}} = \emptyset$ //Step-2: Decomposition Phase
for each $\mathcal{R}_x$ in $\mathcal{R}$ **do**
    $\mathcal{M}_{\mathcal{R}x}$ = Power Set of $\mathcal{R}_x$
    for each $\mathcal{S}_i$ in $\mathcal{S}$ **do**
        for each $\mathcal{Z}_i$ in $\mathcal{M}_{\mathcal{R}x}$ **do**
            if $\mathcal{S}_i \subseteq \mathcal{Z}_i$ **then**
                remove $\mathcal{Z}_i$ from $\mathcal{M}_{\mathcal{R}x}$
            end if
            for each α in $\mathcal{S}_i$ **do**
                for each λ in $i_{\alpha}^{\mathcal{F}}$ **do**
                    if $(\{\alpha\} \cup \lambda) \subseteq \mathcal{Z}_i$ **then**
                        remove $\mathcal{Z}_i$ from $\mathcal{M}_{\mathcal{R}x}$
                    end if
                end for
            end for
        end for
    end for
    for each $\mathcal{V}_i$ in $\mathcal{M}_{\mathcal{R}x}$ **do**
        for each $\mathcal{W}_i$ in $\mathcal{M}_{\mathcal{R}x}$ **do**
            if $\mathcal{V}_i \subseteq \mathcal{W}_i$ **then**
                remove $\mathcal{V}_i$ from $\mathcal{M}_{\mathcal{R}x}$
            end if
        end for
    end for
    $\mathcal{M}_{\mathcal{R}} = \mathcal{M}_{\mathcal{R}} \cup \mathcal{M}_{\mathcal{R}x}$
end for
return $\mathcal{M}_{\mathcal{R}}$
end

Secure decomposition algorithm for the $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$ with the given security dependencies set $\mathcal{S}$ is given in Algorithm 1. The secure decomposing algorithm works as follows:

- Step-1: Until there is no change in $\mathcal{S}$, for all probabilistic dependencies in $\mathcal{P}$, each element of $\mathcal{S}$ is checked whether to include the right hand side of a probabilistic dependency. If included, a new security dependent set is produced containing the left hand side, in order to preserve privacy against probabilistic attacks.(lines 3-14)

- Step-2: For all relational schemas in $\mathcal{R}$,

    1. Firstly, power set of the a relational schema is generated, which is called as $\mathcal{M}_{\mathcal{R}x}$ in the algorithm (line (18)).
    2. Then, for each security dependency set in $\mathcal{S}$ (line (19)) each element of $\mathcal{M}_{\mathcal{R}x}$ (line (20)) is processed. The set is eliminated if:
        - it contains all attributes of that security dependent set together (lines (21-23)), or,
        - it contains one of the attributes of the security dependent set with the attribute's any identifier set together (lines (24-30))
    3. After that, among the remaining subsets; redundant ones (used for unnecessary sub-relations composed by other sub-relations) are also eliminated (lines (33-39)).

The elimination strategy is aimed to create a secure logical schema. It is important to note that all of the work in this chapter of document is concentrated on security dependent sets. Actually there may be some basic policy rules as a single attribute should not be accessed in any context and these basic cases can be easily handled with simple extensions to the algorithm. However it is left as a future work to define a complete attribute based inference control mechanism.

Theorem 4.2.1. *Decomposition algorithm performs secure decomposition on given* $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$ *for a given* $\mathcal{S}$.

Proof. For the proof, we should revisit the properties of secure decomposition given in Definition 9. Before that it is important to notice that the algorithm modifies the set of the security dependent sets by introducing new ones according to the probabilistic dependencies at the beginning.

1. Any attribute cannot be lost after decomposition algorithm as the algorithm cannot remove one element subsets of each relation since;
    - Security dependency sets should have at least two elements by its definition and cannot be included by a one element subset.
    - An attribute in a security dependent set cannot be its identifier as reflexive functional dependencies are excluded in the definition of $\mathcal{F}^+$ and the set of identifier set thereby. So again, at least two element set (attribute and its identifier which has one attribute minimally) cannot be subset of one element subset.
2. Any new functional dependencies cannot be introduced besides some existing ones may be lost because some subsets of each relation schema are eliminated.
3. Subsets containing security dependent sets are eliminated (lines (19-21)).
4. All subsets containing an attribute from a security dependent set and it's any identifier set are eliminated (lines (22-28)).

□

The following parameters of $\mathcal{L} = (\mathcal{R}, \mathcal{F}, \mathcal{P})$ and $\mathcal{S}$ affect the performance of decomposition algorithm.

- $\rho : \#probabilistic\ dependencies \in \mathcal{P}$
- $\pi : \#relations \in \mathcal{R}$
- $\psi : \#security\ dependency\ sets \in \mathcal{S}$
- $\epsilon : max_{\mathcal{R}_i \in \mathcal{R}}\{|\mathcal{R}_i|\}$

- $\eta : max_{\mathcal{S}_i \in \mathcal{S}}\{|\mathcal{S}_i|\}$

- $\mu : max_{\alpha \in \mathcal{S}_i (\mathcal{S}_i \in \mathcal{S})}\{\left|i_\alpha^{\mathcal{F}}\right|\}$

The decomposition algorithm works at a cost of $\mathcal{O}(\psi \cdot \rho + \pi \cdot \psi \cdot 2^{\epsilon} \cdot \eta \cdot \mu)$. The decomposition problem is a more complicated version of maximal independent sets problem [43] in an undirected graph, in which attributes can be thought as vertices and security dependencies as edges. Our algorithm is more complex since, the algorithm takes the functional dependencies into account while performing decomposition and generating new dependency sets(edges) accordingly, which makes maximal independent sets finding algorithms unusable for this problem. In [43] it is shown that generating maximal independent sets is an NP-Hard optimization problem.

We have concentrated on building a lossy decomposion as the aim is to inhibit the possibility of relating the attributes in a security dependent set. The best solution for this problem can be losing minimum number of functional dependencies to achieve the goal. For that purpose, each algorithmic solution should discard the relations in which the attributes in a security dependent set resides together, as our proposed algorithm does in lines [21-23]. The next step is breaking the transitive connection among the attributes through joins, to prevent their inference. There can be two different approaches; as the first may cut all transitive join chain probabilities from all connection centers of the chain, and the second may prefer to make it impossible starting from the starts of the chain as given in following example schema.

$\texttt{Entity}_1$ = ($\underline{\texttt{id}_1}$, $\texttt{att}_1$)

$\texttt{Entity}_2$ = ($\underline{\texttt{id}_2}$, $\texttt{att}_2$)

$\texttt{Relation}_1$ = ($\underline{\texttt{id}_1}$, $\underline{\texttt{id}_i}$)

$\texttt{Relation}_2$ = ($\underline{\texttt{id}_i}$, $\underline{\texttt{id}_j}$)

$\texttt{Relation}_3$ = ($\underline{\texttt{id}_j}$, $\underline{\texttt{id}_k}$)

$\texttt{Relation}_4$ = ($\underline{\texttt{id}_2}$, $\underline{\texttt{id}_k}$)

$\texttt{Relation}_5$ = ($\underline{\texttt{id}_1}$, $\underline{\texttt{id}_2}$)

Assume that $\texttt{id}_i$, $\texttt{id}_j$, $\texttt{id}_k$ are the keys of different entities and there is a security dependent set as $\{\texttt{att}_1, \texttt{att}_2\}$.

First approach will try to cut the chain by performaing decomposition on $\texttt{Relation}_3$ and $\texttt{Relation}_5$ and the second will carry out decomposition on $\texttt{Entity}_1$ and $\texttt{Entity}_2$. Both approaches destroy the transitive join chain among the attributes of the security dependent set and both satisfies the optimal solution. The first approach has a drawback as breaking the join chain for all attributes using the same chain for joins, whereas second makes the securely dependent attributes unidentifiable. We prefer to use the second approach since there can be many different join chain centers(as in $\texttt{Relation}_3$ and $\texttt{Relation}_5$) in the logical schema so first approach can lead to many decomposed relations while preventing lots of legal queries.

It is important to note that being a proactive solution, the exponential complexity of decomposition algorithm is not a critical problem since it is only executed once as preprocessing phase.

Another point for the decomposition is that, this may lead to relations with no key and because of that reason, duplicate rows may occur in views. Handling mechanism for duplicate rows can change up to the implementation strategy, and keyless relations are not a problem since they are a fact of anonymity.

4.3 Discussions

The aim of this chapter is to construct proactive context dependent attribute based security mechanism for relational database system using given security dependent sets. Formal model of the system and the decomposition algorithm is given together with the proofs that the algorithm produces a schema that is in compliance with the given security dependencies.

The main objective in this work is to prevent inference of association of the attributes in each security dependent set, and this is accomplished by performing a secure de-

composition which transforms the relevant logical schema to a secure logical schema. We have formally proven that, on the secure logical schema it is impossible to infer association among the attributes of any security dependent set. Furthermore, we have also proven that the decomposition algorithm produces secure logical schema.

In order to show the advantages of this proactive strategy several experiments are conducted. We have used TPC-H schema and artificially generated databases in order to show the effectiveness of our proactive approach over the best reactive solution. Our experiments show that proactive approach is always preferable over reactive alternatives.

It should also be noted that all work in this paper are about building an external schema of the database according to the given logical schema (including relational schemas, functional dependencies, probabilistic dependencies and even given pseudo-keys) and security dependent sets, and therefore, it can be implemented independently from logical and physical model. As a result, different external schemas for all different roles of users in database can be achieved, and each user can access to the database through a different view from the point of security.

CHAPTER 5

OPTIMIZED DECOMPOSITION WITH RELAXED-CUT

5.1 Motivation

Secure decomposition of the external schema to prevent unwanted inferences has been covered firstly in literature by [39] and [40]. These works concantrate on the required attribute sets (visibility constraints) and produce minimal sized fragments according to the dependencies and constraints. We have improved this process in chapter 4 and we have defined, security dependency set concept and secure decomposition problem formally and we have proposed a decomposition algorithm. The algorithm aims to have maximal fragments (minimal dependency loss) according to forbidden set of attributes.

In chapter 4, a decomposition algorithm is defined to construct a decomposition which prevents to associate any security dependent set. This chapter intoduces a relaxation phase to the decomposition mechanism as the algorithm given in chapter 4 breaks all dependencies to the attributes of secure dependent sets with their identifiers. This will restrict more than wanted, as illustrated with an example below.

Let the relation $R_k = \{A, B, C, D\}$ and A is the primary key, single identifier for all other attributes, as the dependencies are illustrated in Figure 5.1. Additionally, there is a single security dependent set as $\{B, C\}$.

Therefore, the subsets containing both B and C needs to be eliminated in order to prevent the association between B and C. Moreover, the subsets containing $\{A, B\}$ and $\{A, C\}$ will be eliminated as well, since A is the identifier. After the elimination of trivial subsets, secure deposition of R_k can be given as:

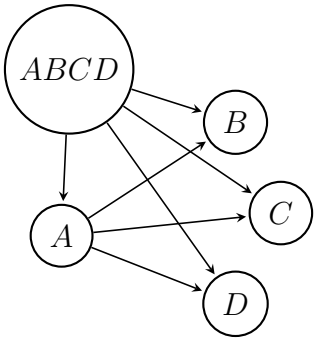

Figure 5.1: Dependency Graph

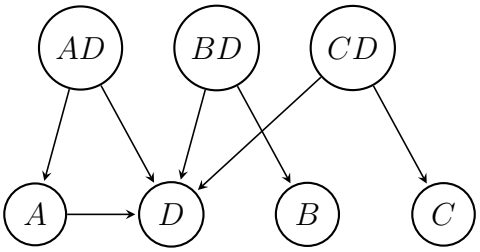

Figure 5.2: Dependency Graph After Strong-Cut

$R_{k_1} = \{A, D\} \quad R_{k_2} = \{B, D\} \quad R_{k_3} = \{C, D\}$

As it can be seen from the new decomposed schema, there is no way to perform a meaningful join between decomposed sets to associate B and C. As a graph notation (details will be discussed later) Figure 5.2 shows the dependecnies formed after the decomposition and according to this figure, there is no way to associate B and C together, starting from a vertex in this graph. In other words, if the ways of associate secure dependent sets attributes are defined as a chain of functional dependencies through meaningful joins, the algorithm breaks these chains from both sides for both attributes. It is obvious that the relations containing the security dependent set should be removed, but the algorithm in chapter 4, breaks association of each attribute in security dependent set with its identifiers to prevent all meaningful joins. In this work, we call this strategy as a $strong - cut$ approach.

However, this $strong - cut$ approach can be relaxed by cutting the chains only at a single point by producing:

$R_{k_1} = \{A, C, D\} \quad R_{k_2} = \{B, D\}$

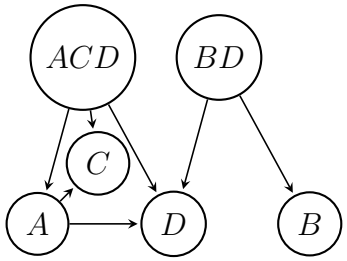

Figure 5.3: Dependency Graph After Weak-Cut

The dependencies of this schema is depicted in Figure 5.3. There is also another possible schema as:

$$R_{k_1} = \{A, B, D\} \quad R_{k_2} = \{C, D\}$$

Both of these schemas are consistent with the privacy constraint defined by security dependency set. The aim of this work is to develop a $relaxed - cut$ algorithm that decomposes the original schema with a minimum loss of functional dependencies while satisfying the security constraints.

The motivation of this work can be described as developing a decomposition that should not be much lossier than needed. This requirement defines an optimization problem and to the best of our knowledge, this is the first attempt in literature to construct an optimized secure decomposition satisfying the policy, while minimizing the dependency loss.

Directed graph representation is selected as the most suitable mathematical model to represent the problem, since a functional dependency can be easily demonstrated as a directed edge and by this way, all algorithmic background in the graph theory can be used for further enhancements of the concept and algorithm. As a result, a new algorithm will be proposed for secure decomposition concept, which aims to decompose the original schema minimally by preserving the idea of prohibiting decomposed relations to be used in meaningful joins to associate the attributes in a security dependent set.

The problem is basically building a decomposition of the original schema, as any set of securely dependent attributes cannot be associated by joins on keys.

5.2 Modelling the Problem with Graph-Theory

We firstly give the basic definitions by using graph notation.

Definition 10. ***Functional Dependency Graph*** *(denoted as FDG hereafter, for a schema) The given functional dependency set (F) of a logical schema (where F is decomposed - i.e., there is a single element on the right hand side - and thus for each functional dependency F_i such as $A_i \rightarrow A_j$, A_i is an attribute set and A_j is a single attribute) can be represented as a directed graph $G = (V, E)$ as follows:*

- *In a normalized schema, all attributes are expected to exist in A_j, but the schema may not be normalized, so each attribute should also be element of V individually.*
- *Each one of A_i and A_j is a single node in V*
- *Each relation (attribute sets) is an individual node in V.*
- *Each dependencies of F_i^+ is an edge in E, if both sides of the dependency exist as a different node in V.*

Example-1: Assume that the logical schema S consists of four relations R_1, R_2, R_3 and R_4:

$R_1 = \{A, B, C, D\}$

$F_{R_1} = \{A \rightarrow B, A \rightarrow C, A \rightarrow D,$

$D \rightarrow A, D \rightarrow B, D \rightarrow C\}$

$R_2 = \{E, F, G, H\}$

$F_{R_2} = \{E \rightarrow F, E \rightarrow G, E \rightarrow H,$

$H \rightarrow E, H \rightarrow F, H \rightarrow G\}$

$R_3 = \{A, E\}$ where A and E are foreign keys.

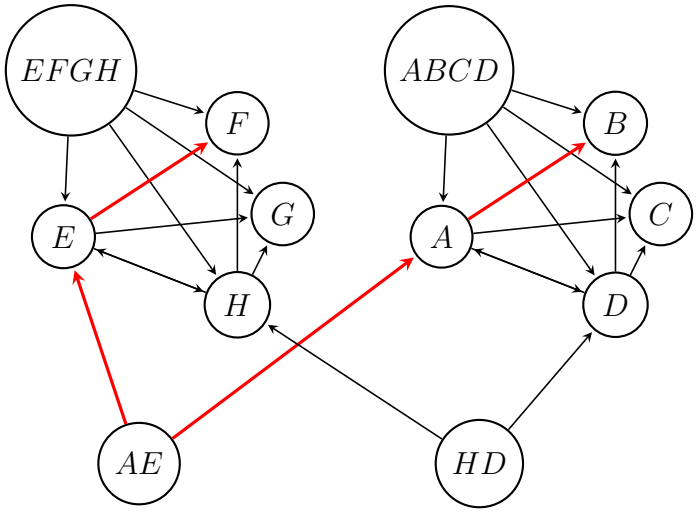

Figure 5.4: FDG of Example-1

$R_4 = \{H, D\}$ where H and D are foreign keys.

Then, the graph (FDG) constructed for this schema is given in Figure 5.4.

The steps of the FDG construction algorithm is as follows:

1. Decompose all functional dependencies in F, such that each functional dependency will have a single element in the right-hand side.

2. Create an individual vertex for all attributes in schema and add to V.

3. Create vertices for the attribute sets with more than one element, which exist in left-hand side of any functional dependency and does not exist in V.

4. Create additional vertices, which include the attributes of a relation in R and does not exist in V.

5. Generate F^+

6. For each $X \rightarrow Y$ in F^+, add an edge to E if X and Y are different vertices in V.

Algorithm 2 Constructing Functional Dependency Graph (FDG)

Require:
S: logical schema as (R, F),
Ensure:
$FGD_S = (V, E)$: functional dependency graph of S

begin
$V \leftarrow \{\}$
$E \leftarrow \{\}$
//Step-1
for each $F_i(X_i \rightarrow Y_i) \in F$ **do**
  if $|Y_i| > 1$ **then**
    remove F_i from F
    for each $Y_j \in Y_i$ **do**
      add $X_i \rightarrow Y_j$ to F
    end for
  end if
end for
//Step-2
for each $R_i \in R$ **do**
  for each $A_j \in R_i$ **do**
    add A_j to V
  end for
end for
//Step-3
for each $F_i(X_i \rightarrow Y_i) \in F$ **do**
  if $|X_i| > 1$ and $X_i \notin V$ **then**
    add X_i to V
  end if
end for
//Step-4
for each $R_i \in R$ **do**
  if $R_i \notin V$ **then**
    add R_i to V
  end if
end for
//Step-5
$F^+ \leftarrow$ Closure set of F
//Step-6
for each $F_i(X_i \rightarrow Y_i) \in F^+$ **do**
  if $X_i \neq Y_i$ and $X_i \in V$ and $Y_i \in V$ **then**
    add $X_i \rightarrow Y_i$ to E
  end if
end for
end

The graph in Figure 5.4 is obtained by the above algorithm for example-1.

Lemma 1. *The edges of transitive closure of FDG is equal to F^+*

Proof. (SKETCH) It can be easily seen that the transitive definition is same for functional dependencies and its corresponding graph. The equivalency is based on the transitive property on the graphs and functional dependencies.

□

Definition 11. ***Common Ancestor of an Attribute Set*** *is a vertex in FDG, from which there exist simple paths to each element of attribute set.*

In Figure 5.4, AE is one of the Common Ancestors for the set of vertices $\{F, B\}$.

Definition 12. ***Join Chain of an Attribute Set*** *(Denoted as JC hereafter) is the set of edges of simple paths in FDG, from a common ancestor to the attribute set.*

The attribute sets may be a forbidden set (i.e. Secure Dependent Set) or a required set (which will be defined later). Let the relational schema is as given in Figure 5.4. Let the forbidden set is $\{F, B\}$, and the functional dependency graph is constructed as in Figure 5.4. The join chain sets according to the forbidden set is given as below. The first join chain is emphasized with red colour and bold for an ilustrative example.

$JC_1 = \{AE \to E, E \to F, AE \to A, A \to B\}$

$JC_2 = \{HD \to H, H \to F, HD \to D, D \to B\}$

$JC_3 = \{AE \to E, E \to H, H \to F,$

$AE \to A, A \to D, D \to B\}$

$JC_4 = \{AE \to E, E \to H, H \to F, AE \to A, A \to B\}$

$JC_5 = \{AE \to E, E \to F, AE \to A, A \to D, D \to B\}$

$JC_6 = \{HD \to H, H \to E, E \to F,$

$HD \to D, D \to A, A \to B\}$

$JC_7 = \{HD \to H, H \to F, HD \to D, D \to A, A \to B\}$

$JC_8 = \{HD \to H, H \to E, E \to F, HD \to D, D \to B\}$

Algorithm 3 Generating Join Chain Set Algorithm

Require:
FDG: functional dependency graph(V, E) of schema
A: attribute set
Ensure:
JC: join chain set
1: //Step-1
2: **begin**
3: $JC \leftarrow \{\}$
4: **for each** $(X_i \to Y_i) \in E$ **do**
5: remove $X_i \to Y_i$ from E
6: add $Y_i \to X_i$ to E
7: **end for**
8: //Step-2
9: Initialize $TargetArr$ as array of array of vertices
10: Initialize $PathArr$ as array of array of set of edges
11: **for each** $A_i \in A$ **do**
12: $C_{A_i} \leftarrow$ empty set of connected vertices
13: $P_{A_i} \leftarrow$ empty set of path edge sets
14: $C_{A_i}, P_{A_i} \leftarrow$ apply DFS to FDG with starting vertex A_i
15: $j \leftarrow 0$
16: **for each** C_j in C_{A_i} **do**
17: $TargetArr[i][j] \leftarrow C_j$
18: $PathArr[i][j] \leftarrow$ simple path to C_j in P_{A_i}
19: $j \leftarrow j + 1$
20: **end for**
21: **end for**
22: //Step-3
23: $SharedArr \leftarrow$ array of shared vertices by all TargetArr rows
24: **for each** $S_i \in SharedArr$ **do**
25: add $\bigcup(PathArr[k][l]$ as $TargetArr[k][l] = S_i)$ to JC
26: **end for**
27: //Step-4
28: **for each** $JC_i \in JC$ **do**
29: **for each** $JC_j \in JC$ **do**
30: **if** $JC_i \supseteq JC_j$ **then**
31: remove JC_i from JC
32: **end if**
33: **end for**
34: **end for**
35: **end**

The steps of the Join Chain Construction algorithm is as follows:

1. All edges are reversed.
2. Taking each element of attribute set (A) as starting vertex, apply DFS up to all connected vertices and all possible simple paths are determined for each end vertex.
3. If there exist simple paths to the same end vertex, which are common (common ancestors in original FDG) for all set attributes (assumed to be starting vertices), all combinations of constructed simple paths, starting from different set attribute and ending in the same vertex is a join chain.
4. If a chain composes another chain, it is discarded.

In example-1, HD and AE are determined as common ancestors and all combinations of simple paths as $AE \rightarrow B$ (2 alternatives) & $AE \rightarrow F$ (2 alternatives) and $HD \rightarrow B$ (2 alternatives) and $HD \rightarrow F$ (2 alternatives) are given as different join chains.

5.3 Relaxed Cut Decomposition Algorithm

Definition 13. ***Minimum-Cut Secure Decomposition:*** *Decomposing the relational schema by removing the minimum number of functional dependencies (i.e., not allowing the attributes of the lost functional dependency $A_i \rightarrow A_j$ in the same relation) to satisfy all security requirements.*

Minimized-Cut Secure Decomposition Problem is equivalent to Minimum Hitting Set Problem [44, 45] and thus it is NP-Complete. We propose a simple greedy heuristic algorithm to solve Relaxed-Cut Secure Decomposition problem (which is defined below), and due to the structure of our problem, we observed that this greedy approach mostly determines the optimal solution.

Definition 14. ***Relaxed-Cut Secure Decomposition:*** *Decomposing the relational schema by greedly removing the functional dependencies in ordet to cut the dependencies of secure dependent attributes at least through one of the join chains.*

Unlike strong cut which cuts the identifiers of all attributes of the secure dependent sets, relaxed cut aims to remove the functional dependencies as little as possible. The steps of the Relaxed-Cut Secure Decomposition algorithm are as follows:

1. Calculate all join chains(JC_i) for each security dependent set (Algorithm-2).
2. For each edge in the FDG, determine the number of times (SecurityCount) it appears in join chains.
3. Sort the edges first according to their SecurityCount in descending order, then, the number of attributes on the nodes at both sides of the edge, in ascending order (in order to cut lower chains first).
4. Traverse the sorted list and mark each join chain as cut, if the edge is contained. These edges are selected ones and to be a selected one, an edge should be an element of at least one unmarked join chain. Set of selected edges are named as new security dependent sets.
5. All subsets of the attributes of the relational schema are generated, which is called as PSR_x in the algorithm. Then, for each new security dependency set, each element of PSR_x is processed. The element set is eliminated if it contains all attributes of that security dependency set together.
6. After that, among the remaining subsets redundant ones (used for unnecessary sub-relations composed by other sub-relations) are also eliminated.

The steps 1 through 4 can be named as "Relaxation Stage" and steps 5 and 6 as "Decomposition Stage". Decomposition stage is a subpart of the secure decomposition algorithm proposed in chapter 4 except the identifier elimination stage.

Example-2: Consider the following schema and the forbidden sets.

$R_1 = \{\underline{A}, \underline{B}, C, D\}$

$R_2 = \{\underline{E}, F, G\}$

$R_3 = \{\underline{A, E, M}\}$

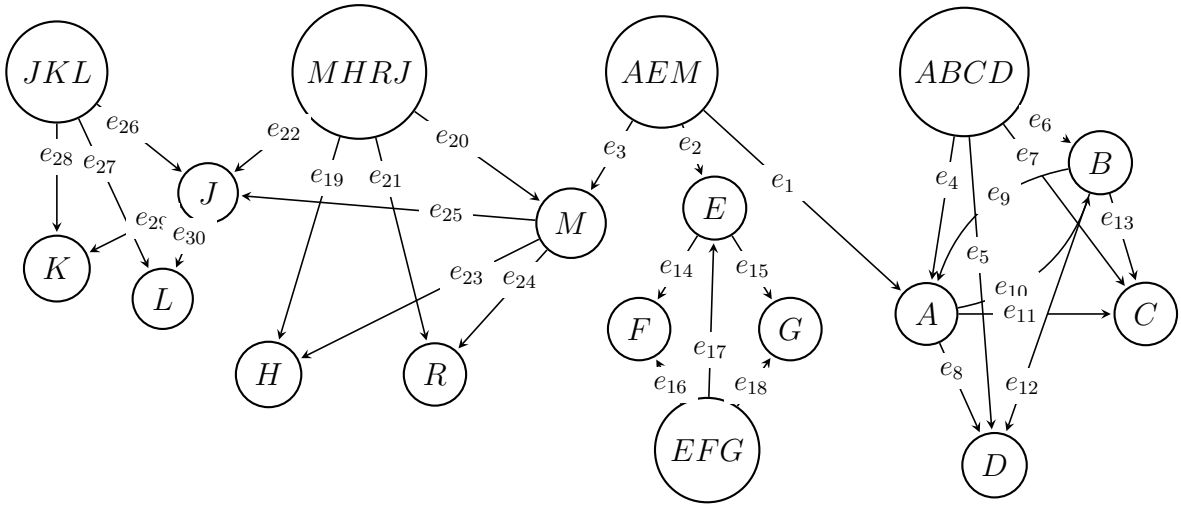

Figure 5.5: FDG of Example-3

$R_4 = \{\underline{J}, K, L\}$

$R_5 = \{\underline{M}, H, R, J\}$

Forbidden Sets = $\{A, D\}, \{D, F\}, \{K, H\}$

Functional dependency graph is constructed as in Figure 5.5. Join chains are determine for this example as follows:

For $\{A, D\}$:

$JC_1 = \{e_8\}$ $JC_2 = \{e_{10}, e_{12}\}$

$JC_3 = \{e_4, e_5\}$ $JC_4 = \{e_6, e_{12}, e_4\}$

$JC_5 = \{e_5, e_6, e_9\}$ $JC_6 = \{e_9, e_{12}\}$

For $\{D, F\}$:

$JC_7 = \{e_1, e_2, e_8, e_{14}\}$ $JC_8 = \{e_1, e_2, e_{10}, e_{12}, e_{14}\}$

For $\{K, H\}$:

$JC_9 = \{e_{23}, e_{25}, e_{29}\}$ $JC_{10} = \{e_{19}, e_{22}, e_{29}\}$

$JC_{11} = \{e_{20}, e_{22}, e_{23}, e_{29}\}$

Algorithm 4 Relaxed-Cut Secure Decomposition Algorithm

Require:
LS: logical schema as (R, F), SD: set of security dependent sets for LS
Ensure:
PS_R: a subset of maximal subsets of R satisfying security decomposition

1: **begin**
2: $JC \leftarrow$ empty array of join chain sets //Step-1
3: **for each** $SD_i \in SD$ **do**
4: $JC_i \leftarrow$ join chain set for SD_i (Algorithm-2) and add JC_i to JC
5: **end for**
6: $FDG(V, E) \leftarrow$ FDG of LS and $SecurityCount \leftarrow$ array of 0, size $|E|$
7: **for each** $E_i \in E$ **do** //Step-2
8: **for each** $JC_k \in JC$ **do**
9: **for each** $JC_{k_i} \in JC_k$ **do**
10: **if** $E_i \in JC_{k_i}$ **then** $SecurityCount[i]++$
11: **end if**
12: **end for**
13: **end for**
14: **end for**
15: $E \leftarrow$ sort edges in descending order (uses $SecurityCount$) //Step-3
16: $Selection \leftarrow$ empty set of edges
17: **for each** $E_i \in E$ **do** //Step-4
18: **for each** $JC_k \in JC$ **do**
19: **for each** $JC_{k_i} \in JC_k$ **do**
20: **if** $E_i \in JC_{k_i}$ and JC_{k_i} is unmarked **then**
21: mark JC_{k_i} and add E to $Selection$
22: **end if**
23: **end for**
24: **end for**
25: **end for**
26: $SD_{new} \leftarrow$ empty set of attribute sets
27: **for each** $(X_i \rightarrow Y_i) \in Selection$ **do** add $(X_i \cup Y_i)$ to SD_{new}
28: **end for**
29: **for each** $R_x \in R$ **do** //Step-5
30: $PSR_x \leftarrow$ Power Set of R_x
31: **for each** $D_i \in SD_new$ **do**
32: **for each** $S_j \in PSR_x$ **do**
33: **if** $D_i \subseteq S_j$ **then** remove S_j from PSR_x
34: **end if**
35: **end for**
36: **end for**
37: **for each** $SS_i \in PSR_x$ **do** //Step-6
38: **for each** $S_j \in SS_i$ **do**
39: **if** $S_j \subseteq SS_i$ **then** remove S_j from PSR_x
40: **end if**
41: **end for**
42: **end for**
43: **end for**
44: **end**

Finally, relaxed cut secure dependency algorithm is executed using Security Counts. The edges are shown up to all marked ones in Table-1. Plus (+) sign in a row indicates that this edge is selected and the join chain on the column is marked. Minus (-) sign is used for already marked join chains and the edges without plus (+) sign in the row is not selected.

As a result, the following edges are selected:

$\{B \rightarrow D, A \rightarrow D, J \rightarrow K, B \rightarrow A, ABCD \rightarrow A\}$

Output of Algorithm in chapter 4 with the Security Dependent Set $\{A, D\}, \{D, F\}, \{K, H\}$ would be as (i.e. with strong-cut):

$R_{1_1} = \{A, C\} \quad R_{1_2} = \{B, C\} \quad R_{1_3} = \{C, D\}$

$R_{2_1} = \{E, G\} \quad R_{2_2} = \{F, G\}$

$R_3 = \{A, E, M\}$

$R_{4_1} = \{J, L\} \quad R_{4_2} = \{K, L\}$

$R_{5_1} = \{M, R, J\}\ R_{5_2} = \{J, R, H\}$

On the other hand the output of Algorithm-3 with Security Dependent Sets

$\{B, D\}, \{A, D\}, \{J, K\}, \{A, B, C, D\}$ will be as follows:

$R_{11} = \{A, C\} \quad R_{12} = \{B, C\} \quad R_{13} = \{C, D\}$

$R_2 = \{E, F, G\}$

$R_3 = \{A, E, M\}$

$R_{41} = \{J, L\} \quad R_{42} = \{K, L\}$

$R_5 = \{M, H, R, J\}$

The algorithm can be improved by defining a total participation count to all edges for all possible join chains of combination of attributes but it will result in a high time-cost.

Theorem 5.3.1. *Algorithm-3 generates a secure logical schema.*

Proof. Assume that the resulting decomposed relations can be joined by foreign keys to associate securely dependent attributes. Then the functional dependency graph of new schema should contain a join chain for the attributes of this security dependent set. However, this cannot happen since each join chain has been cut at least by an edge and the attributes of these cut edges are given as new security dependent sets, which means their coexistence is prevented.

Therefore, resulting decomposed relations form a secure logical schema and the new forbidden sets, serve for the same privacy degree with respect to secure decomposition in chapter 4 by using original forbidden sets.

□

CHAPTER 6

EXTENDING THE PROBLEM WITH REQUIRED SETS

In this chapter, the problem and the solution is extended with the notion of required sets. Traditionally, the user roles and their use-cases are defined during design phase. Therefore, a verification strategy is needed for the privacy preserving decomposition of external layer.

Definition 15. ***Required Attribute Set*** *(Denoted as RS hereafter) is a set of attributes in the relational schema, which should be associated with a series of meaningful joins to satisfy a functionality of the applicational usage.*

It is important to note that, each functionality of a user role should be mapped to a set of RS.

Required and forbidden sets must be consistent, and the decompostion algorithm must satisfy both requirements.

Definition 16. ***Consistency Check Between Required Sets and Forbidden Sets:*** *Required and Forbidden Sets are consistent with each other if there is a "cut set" that contains at least one element from each one of the join chains for each forbidden set (each forbidden set forms a set of join chains) and there is at least one set in join chains corresponding to each required set (each required set forms a set of join chains) that do not contain any element from the cut set.*

In the case of any inconsistency discovered, security policy and association set needs should be revised by the designer.

Consistency check (CC) problem can be simplified as follows: the edges are mapped to letters, and thus join chains become set of letters. The consistency check problem

can be defined as to determine a set of letters (cut set) such that at least one letter from each set of the forbidden sets and none of the letters in at least one of the sets of each required set must be in the cut set.

Consider the following CC problem instance:

$$\text{Join Chains}_{FS} = \left\{ \begin{array}{c} \{e, f, g\} \\ \{a, b, c, d\} \\ \{a, e, c, d\} \\ \{b, f, g\} \end{array} \right\}$$

$$\text{Join Chains}_{RS} = \left\{ \begin{array}{c} \left\{ \begin{array}{c} \{a, b, f, g\} \\ \{b, c, f\} \end{array} \right\} \\ \left\{ \begin{array}{c} \{d, e\} \\ \{c, g\} \end{array} \right\} \end{array} \right\}$$

Cut-Set $= \{a, g\}$

Preserved Required Words $= \{\{d, e\}, \{b, c, f\}\}$

The above CC instance is consistents since at least one element of the cut set {a, g} is in each forbidden set, and one set for each set of required sets (as {d, e} and {b, c, f}) do not contain any element of the cut set.

On the other hand, the following CC instance is inconsistent since no cut set satisfying the requirements exists.

$$\text{Join Chains}_{FS} = \left\{ \begin{array}{c} \{e, f, g\} \\ \{a, b, c, d\} \\ \{a, e, c, d\} \\ \{b, f, g\} \end{array} \right\}$$

$$\text{Join Chains}_{RS} = \left\{ \begin{Bmatrix} \{a,b,f,g\} \\ \{b,f\} \end{Bmatrix} \begin{Bmatrix} \{d,g\} \\ \{c,g\} \end{Bmatrix} \right\}$$

Found = Inconsistency

Theorem 6.0.1. *Consistency Check (CC) Problem is NP-Complete.*

Proof. Input: Set of sets A and set of sets of sets B as follows:

$\mathcal{A} = \{\{a_{1_1}, a_{1_2}, ...\}, \{a_{2_1}, ...\}, ...\}$

$\mathcal{B} = \left\{\left\{\left\{b_{1_{1_1}}, b_{1_{1_2}}, ...\right\}, \left\{b_{1_{2_1}}, ...\right\}, ...\right\}, \left\{\left\{b_{2_{1_1}}, ...\right\}, ...\right\}, ...\right\}$

Problem: Given A and B as above determine (i.e. it is consistent), if there is a set of a_{i_j} for each i in A, there is at least one r for each k in B such that b_{k_r} does not include any of a_{i_j}.

More formally;

Given (A, B) sets, the system is consistent

if there exists $C = \{a_{i_j} | \forall i \exists j (a_{i_j} \in A)\}$

such that $\forall k \exists r (b_{k_r} \in B \wedge b_{k_r} \cap C = \emptyset)$

NPC Proof: Given 3SAT instance constuct an instance of CC as follows:

3SAT instance: $INS_{3SAT} = (p_{1_1} \vee p_{1_2} \vee p_{1_3}) \wedge (p_{2_1} \vee p_{2_2} \vee p_{2_3}) \wedge ...(p_{s_1}..)$ such that each p_{i_j} is either q_k or $\neg q_k$ and there are exactly k different propositional variables q_1,...q_k

Construct instance INS_{CC} as follows:

Generate A = $\{\{q_1, \neg q_1\}, ..., \{q_k, \neg q_k\}\}$

Generate B $= \{\{\{p_{1_1}\}, \{p_{1_2}\}, \{p_{1_3}\}\}, ...\}$

INS_{3SAT} is satisfiable if and only if INS_{CC} is consistent.

INS_{3SAT} is satisfiable if there is a truth assignment for each literal to make all clauses as true. This is equivalent to INS_{CC}, such that for each literal one element from each $\{q_i, \neg q_i\}$ is selected to be in set C (that is equivalent to false in INS_{3SAT} or an edge to be cut in original consistency check problem). If each clause in satisfiable in INS_{3SAT}, then, at least 1 literal of p_{i_1} or p_{i_2} or p_{i_3} must be true. That means either $\{p_{i_1}\} \cap C = \emptyset$ or $\{p_{i_2}\} \cap C = \emptyset$ or $\{p_{i_3}\} \cap C = \emptyset$.

□

The following algorithm checks for inconsistency for given required sets RS and forbidden sets SD. The initial iterations are the same to find a suitable decomposition, if exists.

The steps of the Algorithm-4 is as follows:

1. Calculate all join chains(JC_x) for each security dependent set SD (Algorithm-2).

2. Calculate all join chains(JC_y) for each required attribute set RS (Algorithm-2).

3. For each different edge combination set, in which each single edge is chosen from a single join chain in all JC_y's, check if these selected edges are unbroken while breaking all forbidden sets's join chains, then there is a disjoint edge set which breaks all join chains in JC_x. If such an edge combination cannot be found then inconsistency exists.

The process will continue with decomposition stage of Algorithm-3 with $CutSet$ output to find a secure decomposition.

Table 6.1: Greedy Edge Selection Phase

EDGE	**ID**	**SC**	1	2	3	4	5	6	7	8	9	10	11
$B \rightarrow D$	e_{12}	**4**		+		+		+		+			
$J \rightarrow K$	e_{29}	**3**									+	+	+
$A \rightarrow D$	e_8	**2**	+						+				
$B \rightarrow A$	e_9	**2**					+	-					
-	e_{10}	2		-						-			
-	e_{14}	2							-	-			
-	e_{23}	2									-		-
-	e_1	2							-	-			
-	e_2	2							-	-			
$ABCD \rightarrow A$	e_4	**2**			+	-							

Algorithm 5 Consistency Check and Determining Cut Set Algorithm

Require:
RS: set of required sets for logical schema, SD: set of security dependent sets
Ensure:
$Inconsistent$: true or false, $CutSet$: possible forbidden sets for decomposition

1: **begin**
2: $JC_x \leftarrow$ set of join chain sets, $JC_y \leftarrow$ array of set of join chain sets
3: **for each** $SD_i \in SD$ **do** //Step-1
4: $Frb_i \leftarrow$ join chain sets for SD_i (Algorithm-2)
5: add Frb_i to JC_x
6: **end for**
7: $j \leftarrow 0$
8: **for each** $RS_i \in RS$ **do** //Step-2
9: $Req_i \leftarrow$ join chain sets for AS_i (Algorithm-2)
10: add Req_i to $JC_{y[j]}$ and $j \leftarrow j + 1$
11: **end for**
12: **for each** possible edge set $CutSet$, as one edge is selected from an
13: element of all JC_y members **do** //Step-3
14: $found \leftarrow$ true
15: **for each** $JC_{x_k} \in JC_x$ **do**
16: **if** $JC_{x_k} \cap CutSet \neq \emptyset$ **then** $found \leftarrow$ false and break
17: **end if**
18: **end for**
19: **if** $found$ **then** break
20: **end if**
21: **end for**
22: **end**

CHAPTER 7

EXPERIMENTS

7.1 Experiments for Strong-Cut Approach

In this chapter, comparison between proactive and reactive strategies for context dependent attribute based inference control mechanisms is presented through experiments. In these experiments, TPC-H schema [46] and related, artificially populated data sets are used. TPC-H schema is a well-known benchmark, which is used for query evaluation and its original schema is given in Appendix. In our experiments we have mainly concentrated on the overhead duration of the reactive algorithm since proactive strategy does not have any overhead while processing the query. We have determined both the absolute overhead time and the percentage of overhead ratio to the query running time.

Firstly, for a sample application on TPC-H schema, set of security dependencies and probabilistic dependencies will be given in this section. Afterwards, the new schema generated by the decomposition algorithm of our approach will be presented. Then, the implementation alternatives of the reactive strategy will be discussed and the overhead duration and percentage ratios will be investigated for different dimensions. In addition to that, new and larger sample database schemas are generated using the characteristics of TPC-H schema and further analysis of the overhead duration will be discussed for these schemas. Next, we will move on TPC-H queries given in [46] and present timing analysis for these queries. Lastly, the comments and discussions about the experiment results will be given with a comparative perspective among proactive and reactive strategies.

7.1.1 Sample User Definition with Secure Logical Schema

Assume that a company using TPC-H schema for its database and there is an application on this schema in the company in which inference of association between the financial details of the orders of customers and the individual details of the related customer needs to be prevented. The set of attributes for the financial details of the orders can be given as follows:

```
{L_EXTENDEDPRICE,L_TAX,L_DISCOUNT,O_TOTALPRICE}
```

Additionally, the set of the attributes containing individual details of the customers are as follows:

```
{C_NAME,C_ADDRESS,C_PHONE}
```

The prevention of inference among the attributes of these two sets can be achieved through defining following security dependent sets:

$$\mathcal{S} = \left\{ \begin{array}{l} \{C\_NAME, L\_EXTENDEDPRICE\} \\ \{C\_ADDRESS, L\_EXTENDEDPRICE\} \\ \{C\_PHONE, L\_EXTENDEDPRICE\} \\ \{C\_NAME, L\_TAX\} \\ \{C\_ADDRESS, L\_TAX\} \\ \{C\_PHONE, L\_TAX\} \\ \{C\_NAME, L\_DISCOUNT\} \\ \{C\_ADDRESS, L\_DISCOUNT\} \\ \{C\_PHONE, L\_DISCOUNT\} \\ \{C\_NAME, O\_TOTALPRICE\} \\ \{C\_ADDRESS, O\_TOTALPRICE\} \\ \{C\_PHONE, O\_TOTALPRICE\} \end{array} \right\}$$

In addition to these security dependent sets, the attributes such as date, comment, and account balance potentially can behave as a key according to the data distribution. These attributes having key-like behaviors should be introduced to the system

as functional dependencies and as they become identifiers for securely dependent attributes. These attributes are:

```
{C_ACCTBAL,C_COMMENT,L_SHIPDATE,L_COMMITDATE,
        L_RECEIPTDATE,L_COMMENT,O_COMMENT,O_ORDERDATE}
```

Assume that according to company regulation, the customers are mapped to the discount rates based on the quantity of their orders. It should be noted that, this behavior is not a key-like behavior as key-like behavior should use the stored data in order to identify the attribute. However in this case, it is possible to determine the discount rates when quantity is known. This behavior is not related to the data distribution, and there is no need to access to the rest of the data in order to find out such probabilistic dependency. Therefore, in this schema, the quantity attribute may form a probabilistic dependency for the discount as defined below:

$$\mathcal{P} = \Big\{(L\_QUANTITY \mapsto L\_DISCOUNT)\Big\}$$

As given in decomposition algorithm, this probabilistic dependency will lead to new security dependent sets.

$$\left\{\begin{array}{l}\{C\_NAME, L\_QUANTITY\}\\ \{C\_ADDRESS, L\_QUANTITY\}\\ \{C\_PHONE, L\_QUANTITY\}\\ \{L\_SHIPDATE, L\_QUANTITY\}\\ \{L\_COMMITDATE, L\_QUANTITY\}\\ \{L\_RECEIPTDATE, L\_QUANTITY\}\\ \{L\_COMMENT, L\_QUANTITY\}\end{array}\right\}$$

The resulting secure logical schema obtained after the decomposition algorithm and original TPC-H schema [46] are given in Appendix.

7.1.2 Implementation Alternatives of Reactive Strategy

In proactive method, the original schema is converted to a secure schema by using the decomposition algorithm, and secure schema satisfies all the security dependency constraints. That means, on the secure schema all kinds of queries that can be written are valid from the security point of view, and therefore, there is no need to do any kind of verification. However, in reactive method, each query should be checked if it will violate any of the security constraints. Therefore, we have tried to find out the most efficient reactive implementation that will spend minimum time for checking the applicability of each query under the given security and probabilistic dependencies.

The first remarkable reactive implementation attempt in literature was given in [14], as the nature of the algorithm depends on checking the whole query history repeatedly for each request. In that solution, the overhead time will increase with respect to growing history of the system for each user, and is expected to be intolerable for the users after some period of time. A major improvement to that strategy has been proposed in [47], as all conditions creating a security leak are precomputed, and, the satisfied ones during each query are marked. That is, if the query is rejected, then, there is no need to change the status of any entry, whereas, if the query is permitted to be executed, then, the status changes should be done permanent in order to remember the previous attempts on that entry. In this method, the number of entries will be proportional to the number of the security rules and will not grow with increasing number of queries in history.

In order to apply this strategy into our problem, namely to the context dependent attribute based inference control problem, two database relations must be defined for storing the related security conditions. The first table should store the attributes for each security dependent set as given in Appendix for the sample TPC-H schema. Only the entries for just three security dependent sets are shown in this table. As the number of security dependent sets depends only on the characteristics of the logical schema, the table size will be constant during the whole query history. Consequently, the first check of the reactive strategy is defined to check whether the query includes an association among the attributes of a security dependent set by checking this relation. There is no need to store any status variable for this relation since accessing

any proper subset of a security dependent set (or any set of attributes participating in some security dependent set) should be taken into account in a different relation. The second relation in Appendix stores all chains of associations, which may be formed by executing different queries in to be able to relate the attributes of the security dependent sets. For the sake of performance in reactive strategy, all chains are assumed to be precomputed as in the table in Appendix. It is important to note that these chains should be precomputed by the help of functional dependencies in logical schema. Therefore, the second step of reactive strategy will include the checking of any sub-pair in the relation according to the attribute association of the query. The associated ones should be marked as accessed by storing a status column in the table in Appendix or by using an extra relation. If the query forms all rules of a chain, then it should be rejected, and all status changes should be roll backed. If this condition has not been reached, then, the changes should be committed in order to remember which chain parts have been generated in query history.

The above defined most efficient reactive control method has been implemented and all experiments are carried out by using this implementation. During the experiments, MySQL Server 5.6 is used with originally populated TPC-H data as described in [46]. The experiments are carried out on a system using Microsoft Windows 7 with12 GB RAM and Intel i7 2630QM CPU.

7.1.3 Experiments on TPC-H Schema

First set of experiment are on single table queries, and for these queries the running times with related overheads on TPC-H schema, is given in Table-1. In this table, NA means there is no relation with the specified number of attributes and the number of rows in the schema, therefore the experiment cannot be conducted. In addition to actual running times, the percentage ratios of overhead times to the query processing times are also given in Table-2. The dominancy of the overhead can be seen especially in less dense relations. Remarkable effect of overhead can be easily seen from the experiment. Notice that during the experiments, views are not used even for the most crowded tables. The query running times longer than 1 second is usually not tolerable in typical applications. Thus, usually developers are generally using materialized

Table 7.1: Single Table Query and Overhead Duration(sec) #Entries in Table vs #Attributes in Query

	2 Attributes	**4 Attributes**	**8 Attributes**
25 rows	0,0043	0,0044	NA
10K rows	0,0158	0,0223	NA
150K rows	0,153	0,2346	0,4397
200K rows	0,2137	0,3025	0,6062
800K rows	0,5583	1,0338	NA
1.5M rows	1,4836	2,6557	NA
6M rows	17,935	20,224	24,278
Overhead	0,0106	0,0197	0,0297

views in order to improve the performance. In such cases, since the query execution times will drop, the percentage of overhead rations will grow even for crowded tables.

Next set of experiment are carried out by performing equijoin on tables in order to determine the overheads. In TPC-H schema, the following equijoins can be performed (the number of rows of the relations are also given):

NATION(25) $\bowtie$ REGION(5)

NATION(25) $\bowtie$ SUPPLIER(10K)

NATION(25) $\bowtie$ CUSTOMER(150K)

SUPPLIER(10K) $\bowtie$ PARTSUPP (800K)

PART(200K) $\bowtie$ PARTSUPP(800K)

Table 7.2: Overhead Percentage w.r.t. Single Table Query Processing Duration(%) #Entries in Table vs #Attributes in Query

	2 Attributes	**4 Attributes**	**8 Attributes**
25 rows	246,51%	447,73%	NA
10K rows	67,09%	88,34%	NA
150K rows	6,93%	8,40%	6,75%
200K rows	4,96%	6,51%	4,90%
800K rows	1,90%	1,91%	NA
1.5M rows	0,71%	0,74%	NA
6M rows	0,06%	0,10%	0,12%

Few joins are not used during experiments because of joined table sizes, as they need a better system configuration in order to achieve realistic results.

It can be easily seen that equijoins are slightly faster than single table queries with comparable sizes. This occurs since specially during returning results, equijoins eliminate many rows while processing.

7.1.4 Experiments on Generated Schema

TPC-H schema contains some large relations, but, it is moderately small schema in terms of the number of attributes. In order to be able to measure the effect of larger schemas (in terms of the number of attributes) we have generated artificial schema by using the characteristics of TPC-H. The original TPC-H schema contains 61 attributes only. We have generated larger schemas consisting of 100 to 6400 attributes. For the number of security dependency sets and chains, we used the statistics in TPC-H schema.

Table 7.3: Overhead Percentage w.r.t. Equijoin Query Processing Duration(%) #Entries in Joined Tables vs #Attributes in Query

	2 Attributes	**4 Attributes**	**8 Attributes**
25⋈5	87,60%	162,81%	245,45%
25⋈10K	23,82%	44,27%	66,74%
25⋈150K	1,48%	2,75%	4,14%
10K⋈800K	0,67%	1,25%	1,88%
200K⋈800K	0,25%	0,47%	0,71%

As expected, due to the increase in the security dependency sets and the identifier chains, the overhead of the reactive algorithm has increased considerably in order to be able to verify the applicability of queries.

As it is observed from this set of experiments; the overhead duration increases linearly with the increase in the number of attributes. Obviously this behavior results a linear increase in overhead percentage ratio to the query timings as well since the execution times of queries are not directly affected from the database total attribute count.

7.1.5 Experiment on TPC-H Queries

Our final set of experiments is conducted by using TPC-H queries in [46]. There are 22 queries; however some of them cannot be applied due to the security dependencies defined in the beginning of this section. On the other hand, reactive strategy will always spend some time for checking the applicability of these queries.

The overhead durations and query processing times for both reactive and proactive strategy are listed on Table-5.

It is seen that reactive strategy always spends some time for checking the query

Table 7.4: The Effect of Attribute Count on Overhead Duration(sec) #Attributes in Schema vs #Attributes in Query

	2 Att.s	**4 Att.s**	**8 Att.s**	**16 Att.s**
100 att.s	0,0109	0,0205	0,0304	0,039
200 att.s	0,0111	0,0214	0,0319	0,0392
400 att.s	0,0116	0,0228	0,0342	0,0456
800 att.s	0,0171	0,0342	0,0512	0,0684
1600 att.s	0,0348	0,0655	0,0959	0,1258
3200 att.s	0,0705	0,1411	0,2124	0,2848
6400 att.s	0,2518	0,5178	0,7917	1,0759

whereas this duration is zero for proactive strategy. Moreover, reactive overhead of the prevented queries is more than the performed ones in general as they deal with more number of attributes. Additionally, the decomposed tables may cause a slight decrease on query duration especially because of having less number of attributes, compared to the original relations.

During these experiments, we have used materialized views for the proactive decomposed relations. This choice increases the storage usage but obviously improves the query performance. Unmaterialized view usage makes no sense while defining different external layers to each security role with sufficient performance. In addition to that, reactive overhead percentages may seem negligible at first for some queries, but it is important to note that TPC-H queries are produced for benchmarking for query execution and they are much more complex than a common standard interactive application queries. An interactive application, which spends more than 1 second for a query is not acceptable. If such queries are needed, usually materialized views are used to overcome this performance problem.

Table 7.5: Timing Analysis of TPC-H Queries(sec)

Query No	Reactive Overhead	Reactive Query Dur.	Proactive Query Dur.	Reactive Ov. Perc.
1	0,066	NA	NA	NA
2	0,174	NA	NA	NA
3	0,102	NA	NA	NA
4	0,055	NA	NA	NA
5	0,147	NA	NA	NA
6	0,036	NA	NA	NA
7	0,139	NA	NA	NA
8	0,165	NA	NA	NA
9	0,145	NA	NA	NA
10	0,150	NA	NA	NA
11	0,065	18,29	18,29	0,36%
12	0,066	20,39	15,99	0,32%
13	0,046	NA	NA	NA
14	0,055	NA	NA	NA
15	0,075	NA	NA	NA
16	0,065	1,358	1,299	4,78%
17	0,056	0,481	0,378	11,6%
18	0,084	NA	NA	NA
19	0,093	14,37	6,855	0,64%
20	0,139	NA	NA	NA
21	0,095	38,37	35,66	0,24%
22	0,035	NA	NA	NA

7.2 Experiments for the Relaxed-Cut Approach with Required Sets

The next step is to extend the experiemants for Relaxed-Cut approach. According to its nature, finding functional dependency graph of a given schema is a straightforward algorithm and its time complexity is $O(|F^+|)$ which can be ignored for such a proac-

tive process. Additionally, the time complexity of finding join chains stage for given "Required" or "Forbidden" sets of the process, should be investigated by depending to the base algorithm "all simple paths" [48]. Hence the overall algorithm is related to time complexity of DFS which is negligible for a proactive solution. The main step of the process is the Algorithm-4 as it is the most exhaustive stage. The algorithm can be implemented in both directions:

- Checking each "Allowed" set according to a brute-force selection on "Inhibited" set (Implementation Strategy-I).
- Checking each "Inhibited" set according to a brute-force selection on "Allowed" set (Implementation Strategy-II).

Many heuristics can be added to the algorithm, but these cannot be proven to optimize a particular portion of the input space. The problem is NP-Complete, but the examples are based on logical volumes of databases. It is important to note that, any logical database can be divided into sub-schemas in terms of join chains, so the algorithm can be repeated at each sub-schema easily. The timings are gathered on a i7, 16 GB RAM machine and heuristics are used during implementation for a better result, such as checking the forbidden against allowed, or vice versa up to their counts.

Table-2 refers to the timings of the algorithm when each "Allowed" set is checked according to a brute-force selection on "Inhibited" set and respectively Table-3 for implementation strategy-II.

These experiments have been conducted on TPC-H schema and all results are gathered as lower than 1 ms, as the schema size (in terms of number of edges in FDG) is so small.

These benchmarks show that the algorithm is scalable for widely used database volumes. The algorithm is exponential for both strategies but the more optimal strategy can be chosen according to the brute-force selection set size.

Table 7.6: Timings for Implementation Strategy-I

Criteria	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}
#edges in FDG	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
#edges in Forbidden Join Chains	10	10	10	10	10	10	20	30	40	50
#Forbidden Join Chains	20	40	60	80	100	100	100	100	100	100
#Required Attribute Sets	50	50	50	50	50	50	50	50	50	50
#Join Chains per Required Attribute Set	10	10	10	10	10	10	10	10	10	10
#edge per Join Chain for Required Attribute Set	10	10	10	10	10	10	10	10	10	10
#Duration	2 ms	2 ms	3 ms	3 ms	4 ms	3 ms	3 ms	4 ms	4 ms	6 ms

Table 7.7: Timings for Implementation Strategy-II

Criteria	E_{11}	E_{12}	E_{13}	E_{14}	E_{15}	E_{16}	E_{17}	E_{18}	E_{19}	E_{20}
#edges in FDG	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
#edges in Forbidden Join Chains	10	10	10	10	10	10	10	10	10	10
#Forbidden Join Chains	100	80	60	40	20	100	100	100	100	100
#Required Attribute Sets	50	50	50	50	50	50	50	50	50	50
#Join Chains per Required Attribute Set	10	10	10	10	10	10	10	10	10	10
#edge per Join Chain for Required Attribute Set	10	10	10	10	10	10	20	30	40	50
#Duration	2 ms	2 ms	1 ms	1 ms	1 ms	1 ms	1 ms	1 ms	1 ms	1 ms

CHAPTER 8

CONCLUSION AND FUTURE WORK

The approach given in this document assures a proactive cross-control of the required and inhibited attribute sets and a secure decomposition with maximal availability with minimal loss of dependencies on external schema. The verification of privacy for the data should be checked during design phase of an application as all user roles and according use-cases are stated. This work introduces a proactive approach for this problem and the main aim to prevent security leaks prior to production. These leaks generally have their root causes owing to the design of the system. The main point to check the security schema of the system is database and its usages. There is a clear distinction between the traditional approach and the mechanism given in this thesis.

Traditional design approach concentrates on the requirements of the system and user roles. Each user role is assigned to the intended requirement set and external database layer for this user role is defined accordingly. This approach can be stated as a ”whitelist” approach. Major design problems for security and privacy leaks have their root cause of contradictive whitelist. A simple example can be given as a user role, who should not view identification number of customers, but allowed the reprint the invoices of them. This is actually a granularity problem, since the allowed action contains the forbidden one. As the applications may have many different graphical user interface platforms (as web, mobile, etc.), database-centric policy enforcement and verification with more granular attribute-based control becomes a critical solution to this problem.

The design approach introduced in this thesis, offers a proactive control and decomposition according to both ”whitelist” and ”blacklist”. By this way, blacklisted actions should be defined upon attributes of data, therefore a complete verification mecha-

nism is satisfied. This approach can also be combined with existing normalization procedures of database schema, which only checks for database integrity. As a result, many supplementary seucrity mechanisms need not to be applied during production.

A designer may claim that all user roles should be authorized by only permitted set of queries, therefore there is no need for a decomposition. This idea does not contradict with the approach given in this thesis, as there should be a control mechanism for the permitted set and forbidden set. Decomposition is based on building external layer, which can be built by this way as it does not cancel the fact that it should be verified upon blacklisted actions on data.

As a future work, this optimization should be improved according to the query statistics of user and applicable reactive controls may be integrated. Even the method in this approach is proactive, experiments show that the timings are acceptable for a reactive-like behavior in future. We have experienced the benefits of this approach as given in real-life example section. Additionally, reactive meachnims can be integrated to the mechanism in order to prevent attacks based on unexpected key-like behaviour according to data distribution. To sum up, for the application usage of databases, the algorithms given in this thesis should be used for a proactive control and the given mathematical representation approach may lead to easy-to-build and integrate mechanisms for fine-grained database privacy problem.

REFERENCES

[1] E. Ferrari and B. Thuraisingham, "Security and privacy for web databases and services," in *Advances in Database Technology - EDBT 2004*, vol. 2992 of *Lecture Notes in Computer Science*, pp. 17–28, Springer Berlin Heidelberg, 2004.

[2] J. Biskup, D. W. Embley, and J.-H. Lochner, "Reducing inference control to access control for normalized database schemas," *Inf. Process. Lett.*, vol. 106, pp. 8–12, Mar. 2008.

[3] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, pp. 557–570, Oct. 2002.

[4] M. Stonebraker and E. Wong, "Access control in a relational data base management system by query modification," in *Proceedings of the 1974 Annual Conference - Volume 1*, ACM '74, (New York, NY, USA), pp. 180–186, ACM, 1974.

[5] A. Motro, "An access authorization model for relational databases based on algebraic manipulation of view definitions," in *Proceedings of the Fifth International Conference on Data Engineering*, (Washington, DC, USA), pp. 339–347, IEEE Computer Society, 1989.

[6] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Hippocratic databases," in *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pp. 143–154, VLDB Endowment, 2002.

[7] O. Cooperation, "Oracle database: Security guide." b14266.pdf, July 2012.

[8] J. Shi, H. Zhu, G. Fu, and T. Jiang, "On the soundness property for sql queries of fine-grained access control in dbmss," in *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*, pp. 469–474, June 2009.

[9] C. Dwork, *Differential Privacy: A Survey of Results*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[10] F. McSherry and K. Talwar, "Mechanism design via differential privacy," pp. 94 – 103, 2007.

[11] G. Smith, "The semantic data model for security: representing the security semantics of an application," in *Data Engineering, 1990. Proceedings. Sixth International Conference on*, pp. 322–329, Feb 1990.

[12] Y. Chen and W. Chu, "Protection of database security via collaborative inference detection," in *Intelligence and Security Informatics* (H. Chen and C. Yang, eds.), vol. 135 of *Studies in Computational Intelligence*, pp. 275–303, Springer Berlin Heidelberg, 2008.

[13] T. H. Hinke, "Inference aggregation detection in database management systems," in *Proceedings of the 1988 IEEE Conference on Security and Privacy*, SP'88, (Washington, DC, USA), pp. 96–106, IEEE Computer Society, 1988.

[14] A. Brodsky, C. Farkas, and S. Jajodia, "Secure databases: constraints, inference channels, and monitoring disclosures," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 12, pp. 900–919, Nov 2000.

[15] J.-W. Byun and N. Li, "Purpose based access control for privacy protection in relational database systems," *The VLDB Journal*, vol. 17, pp. 603–619, July 2008.

[16] J. Park, X. Zhang, and R. S, "Attribute mutability in usage control," in *In Proceedings of the Proceedings of 18th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pp. 15–29, Kluwer, 2004.

[17] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy, "Extending query rewriting techniques for fine-grained access control," in *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, (New York, NY, USA), pp. 551–562, ACM, 2004.

[18] E. Bertino, J.-W. Byun, and N. Li, "Foundations of security analysis and design iii," ch. Privacy-Preserving Database Systems, pp. 178–206, Berlin, Heidelberg: Springer-Verlag, 2005.

[19] D. Brewer and M. Nash, "The chinese wall security policy," in *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*, pp. 206–214, May 1989.

[20] E. Bertino, S. Jajodia, and P. Samarati, "Database security: Research and practice," *Information Systems*, vol. 20, no. 7, pp. 537 – 556, 1995.

[21] A. Kumar, N. Karnik, and G. Chafle, "Context sensitivity in role-based access control," *SIGOPS Oper. Syst. Rev.*, vol. 36, pp. 53–66, July 2002.

[22] K. Muralidhar, R. Parsa, and R. Sarathy, "A general additive data perturbation method for database security," *Management Science*, vol. 45, no. 10, pp. pp. 1399–1415, 1999.

[23] S. Jajodia, P. Samarati, M. L. Sapino, and V. S. Subrahmanian, "Flexible support for multiple access control policies," *ACM Trans. Database Syst.*, vol. 26, pp. 214–260, June 2001.

[24] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim, "Optimizing queries with materialized views," in *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pp. 190–200, Mar 1995.

[25] X. Jin, R. Sandhu, and R. Krishnan, "Rabac: Role-centric attribute-based access control," in *Computer Network Security* (I. Kotenko and V. Skormin, eds.), vol. 7531 of *Lecture Notes in Computer Science*, pp. 84–96, Springer Berlin Heidelberg, 2012.

[26] L. J. Buczkowski and E. Perry, "Database inference controller.," in *DBSec*, pp. 311–322, 1989.

[27] J. A. Goguen and J. Meseguer, "Unwinding and inference control," in *2012 IEEE Symposium on Security and Privacy*, pp. 75–75, IEEE Computer Society, 1984.

[28] S. Dawson, S. De Capitani di Vimercati, P. Lincoln, and P. Samarati, "Minimal data upgrading to prevent inference and association attacks," in *Proceedings of the Eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '99, (New York, NY, USA), pp. 114–125, 1999.

[29] S. Dawson, S. De Capitani Di Vimercati, and P. Samarati, "Specification and enforcement of classification and inference constraints," in *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pp. 181–195, 1999.

[30] D. E. Denning, "Commutative filters for reducing inference threats in multilevel database systems," in *Security and Privacy, 1985 IEEE Symposium on*, pp. 134–134, April 1985.

[31] D. G. Marks, "Inference in mls database systems," *IEEE Trans. on Knowl. and Data Eng.*, vol. 8, pp. 46–55, Feb. 1996.

[32] C. Meadows, "Extending the brewer-nash model to a multilevel context," in *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pp. 95–102, May 1990.

[33] M. Morgenstern, "Controlling logical inference in multilevel database systems," in *Security and Privacy, 1988. Proceedings., 1988 IEEE Symposium on*, pp. 245–255, Apr 1988.

[34] P. Stachour and B. Thuraisingham, "Design of ldv: a multilevel secure relational database management system," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 2, pp. 190–209, Jun 1990.

[35] T.-A. Su and G. Ozsoyoglu, "Data dependencies and inference control in multilevel relational database systems," in *2012 IEEE Symposium on Security and Privacy*, pp. 202–202, IEEE Computer Society, 1987.

[36] M. Thuraisingham, "Security checking in relational database management systems augmented with inference engines," *Computers & Security*, vol. 6, no. 6, pp. 479–492, 1987.

[37] G. Aggarwal, M. Bawa, P. Ganesan, H. Garcia-molina, K. Kenthapadi, R. Motwani, U. Srivastava, D. Thomas, and Y. Xu, "Two can keep a secret: A distributed architecture for secure database services," in *In Proc. CIDR*, 2005.

[38] W. Luo, Q. Xie, and U. Hengartner, "Facecloak: An architecture for user privacy on social networking sites," in *2009 International Conference on Computational Science and Engineering*, vol. 3, pp. 26–33, Aug 2009.

[39] S. D. C. di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati, "Fragmentation in presence of data dependencies," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, pp. 510–523, Nov 2014.

[40] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Fragments and loose associations: Respecting privacy in data publishing," *Proc. VLDB Endow.*, vol. 3, pp. 1370–1381, Sept. 2010.

[41] J. Lee and C. Clifton, "Differential identifiability," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, (New York, NY, USA), pp. 1041–1049, ACM, 2012.

[42] W. W. Armstrong and C. Delobel, "Decomposition and functional dependencies in relations," *ACM Trans. Database Syst.*, vol. 5, no. 4, pp. 404–430, 1980.

[43] E. Lawler, J. Lenstra, and A. Rinnooy Kan, "Generating all maximal independent sets: Np-hardness and polynomial-time algorithms," *SIAM Journal on Computing*, vol. 9, no. 3, pp. 558–565, 1980.

[44] J. Bailey and P. J. Stuckey, "Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization," in *Practical Aspects of Declarative Languages* (M. V. Hermenegildo and D. Cabeza, eds.), (Berlin, Heidelberg), pp. 174–186, Springer Berlin Heidelberg, 2005.

[45] A. Gainer-Dewar and P. Vera-Licona, "The minimal hitting set generation problem: algorithms and computation," *CoRR*, vol. abs/1601.02939, 2016.

[46] "Tpc benchmark h (decision support) standard specification revision 2.17.1." Accessed: 2014-12-15.

[47] T. Toland, C. Farkas, and C. Eastman, "Dynamic disclosure monitor (d2mon): An improved query processing solution," in *Secure Data Management* (W. Jonker and M. Petković, eds.), vol. 3674 of *Lecture Notes in Computer Science*, pp. 124–142, Springer Berlin Heidelberg, 2005.

[48] F. Rubin, "Enumerating all simple paths in a graph," *IEEE Transactions on Circuits and Systems*, vol. 25, pp. 641–642, August 1978.

APPENDIX A

PROOFS AND SCHEMAS

1) Proof For Theorem-1

Proof. Step by step proof is given below with a brief description of each step.

1. Assuming $\mathcal{L}$ as a secure logical schema, the formula given in (8) should be satisfied.

   $\forall \mathcal{S}_i((\mathcal{S}_i \in \mathcal{S}) \Rightarrow \forall \mathcal{R}_j(\mathcal{R}_j \in \mathcal{R}^+ \Rightarrow \mathcal{S}_i \not\subseteq \mathcal{R}_j))$

2. Let $\mathcal{S}_i$ be a security dependent set for $\mathcal{L}$.

   $\mathcal{S}_i \in \mathcal{S}$

3. Line (1) can be instantiated by using $\mathcal{S}_i$.

   $(\mathcal{S}_i \in \mathcal{S}) \Rightarrow \forall \mathcal{R}_j(\mathcal{R}_j \in \mathcal{R}^+ \Rightarrow \mathcal{S}_i \not\subseteq \mathcal{R}_j)$

4. When modus ponens is applied using lines (2) and (3).

   $\forall \mathcal{R}_j(\mathcal{R}_j \in \mathcal{R}^+ \Rightarrow \mathcal{S}_i \not\subseteq \mathcal{R}_j)$

5. Bu using the formula (1) of the property of $\mathcal{R}^+$, $\mathcal{S}_i$ should not be element of any existing relation $\mathcal{R}$ and there should not exist any attibute set to which $\mathcal{S}_i$ is functionally dependent since according to line (4), $\mathcal{S}_i$ is not a part of any relation in $\mathcal{R}^+$ so any new relation composing $\mathcal{S}_i$ should not be produced by meaningful joins.

$\forall \mathcal{R}_k(\mathcal{R}_k \in \mathcal{R} \Rightarrow \mathcal{S}_i \not\subseteq \mathcal{R}_k) \wedge \forall \mathcal{A}_l(\mathcal{A}_l \subseteq \mathcal{U}_{\mathcal{R}} \Rightarrow (\mathcal{A}_l \rightarrow \mathcal{S}_i) \notin \mathcal{F}^+)$

6. Assume that the inference of association among the attributes in $\mathcal{S}_i$ can be done. This assumption is the negation of the theorem 1, so proof by contradiction starts here.

   $\mathcal{X}_{\mathcal{L}}(\mathcal{S}_i)$

7. According to line (6), the formula (7) states that $\mathcal{S}_i$ should be inferable or subset of any existing relation in $\mathcal{R}$.

   $\exists \mathcal{A}_n(\mathcal{A}_n \overset{\mathcal{F}}{\rightrightarrows} \mathcal{S}_i \wedge \mathcal{A}_n \subseteq \mathcal{U}_{\mathcal{R}}) \vee \exists \mathcal{R}_o(\mathcal{R}_o \in \mathcal{R} \wedge \mathcal{S}_i \subseteq \mathcal{R}_o)$

8. In order to contradict the $\vee$ expression in line (7), both sides of $\vee$ should be contradicted. Accordingly, the first assumption is given below as $\mathcal{S}_i$ should be inferable.

   $\exists \mathcal{A}_n(\mathcal{A}_n \overset{\mathcal{F}}{\rightrightarrows} \mathcal{S}_i \wedge \mathcal{A}_n \subseteq \mathcal{U}_{\mathcal{R}})$

9. Let the expression in line (8) be instantiated using bound variable $\mathcal{A}_n$ denoting a attribute set which infers $\mathcal{S}_i$.

   $\mathcal{A}_n \overset{\mathcal{F}}{\rightrightarrows} \mathcal{S}_i \wedge \mathcal{A}_n \subseteq \mathcal{U}_{\mathcal{R}}$

10. First $\wedge$ instantiation using line (9).

    $\mathcal{A}_n \overset{\mathcal{F}}{\rightrightarrows} \mathcal{S}_i$

11. Second $\wedge$ instantiation using line (5).

    $\forall \mathcal{A}_l(\mathcal{A}_l \subseteq \mathcal{U}_{\mathcal{R}} \Rightarrow (\mathcal{A}_l \rightarrow \mathcal{S}_i) \notin \mathcal{F}^+)$

12. The universal quantifier in line (11) is instantiated using $\mathcal{A}_n$

    $\mathcal{A}_n \subseteq \mathcal{U}_{\mathcal{R}} \Rightarrow (\mathcal{A}_n \rightarrow \mathcal{S}_i) \notin \mathcal{F}^+$

13. Second $\wedge$ instantiation using line (9).

    $\mathcal{A}_n \subseteq \mathcal{U}_{\mathcal{R}}$

14. When modus ponens is applied using lines (12) and (13).

    $(\mathcal{A}_n \rightarrow \mathcal{S}_i) \notin \mathcal{F}^+$

15. Line (14) can be used to perform modus ponens to the contrapositive of formula (6).

    $\neg(\mathcal{A}_n \overset{\mathcal{F}}{\rightrightarrows} \mathcal{S}_i)$, (14) using formula 6

16. Lines (10) and (15) are leading to a contradiction.

    $\perp$

17. First assumption of $\vee$ expression in line (7) in line (8) has been contradicted. Next, the second assumption is given below as $\mathcal{S}_i$ should be a part of an existing relation.

    $\exists \mathcal{R}_o(\mathcal{R}_o \in \mathcal{R} \wedge \mathcal{S}_i \subseteq \mathcal{R}_o)$

18. First $\wedge$ instantiation using line (5).

    $\forall \mathcal{R}_k(\mathcal{R}_k \in \mathcal{R} \Rightarrow \mathcal{S}_i \nsubseteq \mathcal{R}_k)$

19. The existential quantifier in line (17) is instantiated using bounded variable $\mathcal{R}_o$.

    $\mathcal{R}_o \in \mathcal{R} \wedge \mathcal{S}_i \subseteq \mathcal{R}_o$, let $\mathcal{R}_o$ be a bound variable for $\mathcal{R}_o$

20. Line (18) can be instatiated again by using $\mathcal{R}_o$.

    $\mathcal{R}_o \in \mathcal{R} \Rightarrow \mathcal{S}_i \nsubseteq \mathcal{R}_o$

21. First $\wedge$ instantiation using line (19).

    $\mathcal{R}_o \in \mathcal{R}$

22. Second $\wedge$ instantiation using line (19).

    $\mathcal{S}_i \subseteq \mathcal{R}_o$

23. When modus ponens is applied using lines (20) and (21).

    $\mathcal{S}_i \not\subseteq \mathcal{R}_o$, (20, 21)

24. Lines (22) and (23) are leading to a contradiction for line (17).

    $\perp$

25. Lines (16) and (24) are leading to a contradiction for line (7) which means that it is impossible to make an inference of association among the attributes in $\mathcal{S}_i$.

    $\perp$

26. End of proof by contradiction is reached, hence the theorem holds.

    $\neg\mathcal{X}_{\mathcal{L}}(\mathcal{S}_i)$

□

2) Proof For Theorem-2

Proof. Step by step proof is given below with a brief description of each step.

1. Assume that $\mathcal{L}'$ is a secure logical schema, then it should satisfy the following property given in formula (9).

   $\forall\mathcal{S}_i(\mathcal{S}_i \in \mathcal{S}) \Rightarrow \neg\mathcal{X}_{\mathcal{L}'}(\mathcal{S}_i)$

2. Let $\mathcal{S}_i$ be a security dependent set for $\mathcal{L}'$.

   $\mathcal{S}_i \in \mathcal{S}$

3. Line (1) can be instantiated by using $\mathcal{S}_i$.

   $(\mathcal{S}_i \in \mathcal{S}) \Rightarrow \neg\mathcal{X}_{\mathcal{L}'}(\mathcal{S}_i)$

4. When modus ponens is applied using lines (2) and (3).

   $\neg\mathcal{X}_{\mathcal{L}'}(\mathcal{S}_i)$

5. Proof by contradiction begins by assuming negation of line (4) as if $\mathcal{L}'$ is not a secure logical schema, then inference of association among the attributes of a security dependent set $(\mathcal{S}_i)$ should be possible according to theorem (1).

   $\mathcal{X}_{\mathcal{L}'}(\mathcal{S}_i)$

6. According to line (5), the formula (7) states that $\mathcal{S}_i$ should be inferable or subset of any existing relation in $\mathcal{R}$.

   $\exists\mathcal{A}_n(\mathcal{A}_n \overset{\mathcal{F}'}{\rightrightarrows} \mathcal{S}_i \wedge \mathcal{A}_n \subseteq \mathcal{U}_{\mathcal{R}'}) \vee \exists\mathcal{R}_o(\mathcal{R}_o \in \mathcal{R}' \wedge \mathcal{S}_i \subseteq \mathcal{R}_o)$

7. Let the expression in line (6) be instantiated using bound variable $\mathcal{A}_n$ denoting a attribute set which infers $\mathcal{S}_i$ and $\mathcal{R}_o$ denoting the relation which contains $\mathcal{S}_i$.

   $(\mathcal{A}_n \overset{\mathcal{F}'}{\rightrightarrows} \mathcal{S}_i \wedge \mathcal{A}_n \subseteq \mathcal{U}_{\mathcal{R}'}) \vee (\mathcal{R}_o \in \mathcal{R}' \wedge \mathcal{S}_i \subseteq \mathcal{R}_o)$

8. In order to contradict the $\vee$ expression in line (7), both sides of $\vee$ should be contradicted. Accordingly, the first assumption is given below.

   $(\mathcal{A}_n \overset{\mathcal{F}'}{\rightrightarrows} \mathcal{S}_i \wedge \mathcal{A}_n \subseteq \mathcal{U}_{\mathcal{R}'})$

9. First $\wedge$ instantiation using line (8).

   $\mathcal{A}_n \overset{\mathcal{F}'}{\rightrightarrows} \mathcal{S}_i$, (8)

10. Formula (6) states that $\mathcal{S}_i$ should be functionally dependent to an attribute set when the statement in line (9) exists.

    $(\mathcal{A}_n \rightarrow \mathcal{S}_i) \in \mathcal{F}'^{+}$

11. Using formula (11), the dependency in line (10) can be transformed as below.

    $(\mathcal{A}_n \rightarrow \mathcal{S}_i) \in \mathcal{F}^{+}$

12. $\mathcal{S}_i$ in line (11) cannot be contained by $\mathcal{A}_n$ as partial functional dependencies are excluded in definition of $\mathcal{F}^+$ in Definition (3).

    $\mathcal{A}_n \not\supseteq \mathcal{S}_i$

13. Using lines (11) and (12), it can be stated that $\mathcal{A}_n$ is an identifier set for $\mathcal{S}_i$ according to formula (3).

    $\mathcal{A}_n \in \mathcal{I}^{\mathcal{F}}_{\mathcal{S}_i}$

14. If $\mathcal{L}'$ is a secure logical schema, then formula (13) should be satisfied.

    $\forall \mathcal{S}_i \in \mathcal{S}, \forall \mathcal{R}_i \in \mathcal{R}', \forall \sigma \in \mathcal{S}_i, \nexists \tau \in i^{\mathcal{F}}_{\sigma}((\{\sigma\} \cup \tau) \subseteq \mathcal{R}_i)$

15. There should not be any identifier for $\mathcal{S}_i$ according to the contrpositive of Identifiable Property's formula (5) and line (14) since it is prevented for any attribute in $\mathcal{S}_i$ to be in the same relation with an identifier.

    $\mathcal{I}^{\mathcal{F}}_{\mathcal{S}_i} = \emptyset$

16. $\mathcal{A}_n$ cannot be an identifier according to line (15).

    $\mathcal{A}_n \notin \mathcal{I}^{\mathcal{F}}_{\mathcal{S}_i}$

17. Lines (13) and (16) are leading to a contradiction.

    $\perp$

18. First assumption of $\vee$ expression in line (7) in line (8) has been contradicted. Next, the second assumption is given below as $\mathcal{S}_i$ should be a part of an existing relation.

    $(\mathcal{R}_o \in \mathcal{R}' \wedge \mathcal{S}_i \subseteq \mathcal{R}_o)$

19. Second $\wedge$ instantiation using line (18).

    $\mathcal{S}_i \subseteq \mathcal{R}_o$

20. First $\wedge$ instantiation using line (18).

    $\mathcal{R}_o \in \mathcal{R}'$

21. $\mathcal{S}_i$ cannot be part of any relation according to formula (12).

    $\mathcal{S}_i \not\subseteq \mathcal{R}_o$

22. Lines (19) and (21) are leading to a contradiction.

    $\perp$

23. Lines (17) and (22) are leading to a contradiction for line (5) which means that it is impossible to make an inference of association among the attributes in $\mathcal{S}_i$.

    $\perp$

24. End of proof by contradiction is reached, hence the theorem holds.

    $\neg\mathcal{X}_{\mathcal{L}'}(\mathcal{S}_i)$

□

3) Original TPC-H 2.17.1 Database Schema

As defined in version 2.17.1:

```
PART = (P_PARTKEY, P_NAME, P_MFGR, P_BRAND, P_TYPE,
       P_SIZE, P_CONTAINER, P_RETAILPRICE, P_COMMENT)

SUPPLIER = (S_SUPPKEY, S_NAME, S_ADDRESS,
       S_NATIONKEY(N_NATIONKEY), S_PHONE, S_ACCTBAL,
       S_COMMENT)
```

REGION = (*R_REGIONKEY*, R_NAME, R_COMMENT)

NATION = (*N_NATIONKEY*, N_NAME, N_COMMENT,

N_REGIONKEY(R_REGIONKEY))

CUSTOMER = (*C_CUSTKEY*, C_NAME, C_ADDRESS,

C_NATIONKEY(N_NATIONKEY), C_PHONE, C_ACCTBAL,

C_COMMENT, C_MKTSEGMENT)

PARTSUPP = (*PS_PARTKEY(P_PARTKEY)*, *PS_SUPPKEY(S_SUPPKEY)*,

PS_AVAILQTY, PS_SUPPLYCOST, PS_COMMENT)

LINEITEM = (*L_ORDERKEY(O_ORDERKEY)*, *L_LINENUMBER*, L_QUANTITY,

L_TAX, L_PARTKEY(PS_PARTKEY), L_COMMENT,

L_SUPPKEY(PS_SUPPKEY), L_DISCOUNT, L_EXTENDEDPRICE,

L_RETURNFLAG, L_LINESTATUS, L_SHIPDATE, L_COMMITDATE,

L_RECEIPTDATE, L_SHIPINSTRUCT, L_SHIPMODE)

ORDERS = (*O_ORDERKEY*, O_ORDERSTATUS, O_CLERK,

O_CUSTKEY(C_CUSTKEY), O_TOTALPRICE, O_ORDERDATE,

O_ORDERPRIORITY, O_SHIPPRIORITY, O_COMMENT)

4) Proactive Secure Decomposition of TPC-H Schema

The output of the Secure Decomposition Algorithm:

PART = (*P_PARTKEY*, P_NAME, P_MFGR, P_BRAND, P_TYPE, P_SIZE,

P_CONTAINER, P_RETAILPRICE, P_COMMENT)

SUPPLIER = (*S_SUPPKEY*, S_NAME, S_ADDRESS, S_NATIONKEY,
S_PHONE, S_ACCTBAL, S_COMMENT)

REGION = (*R_REGIONKEY*, R_NAME, R_COMMENT)

NATION = (*N_NATIONKEY*, N_NAME, N_COMMENT, N_REGIONKEY)

$CUSTOMER_1$ = (*C_CUSTKEY*, C_NATIONKEY, C_MKTSEGMENT)

$CUSTOMER_2$ = (C_ACCTBAL, C_COMMENT, C_NATIONKEY, C_MKTSEGMENT)

$CUSTOMER_3$ = (C_NAME, C_ADDRESS, C_PHONE, C_NATIONKEY,
C_MKTSEGMENT)

PARTSUPP = (*PS_PARTKEY*, *PS_SUPPKEY*, PS_AVAILQTY,
PS_SUPPLYCOST, PS_COMMENT)

$LINEITEM_1$ = (*L_ORDERKEY*,*L_LINENUMBER*, L_PARTKEY, L_SUPPKEY,
L_RETURNFLAG, L_LINESTATUS, L_SHIPINSTRUCT,
L_SHIPMODE)

$LINEITEM_2$ = (L_ORDERKEY, L_RECEIPTDATE, L_COMMITDATE,
L_PARTKEY, L_SUPPKEY, L_RETURNFLAG, L_LINESTATUS,
L_SHIPINSTRUCT, L_SHIPMODE, L_SHIPDATE, L_COMMENT)

$LINEITEM_3$ = (L_LINENUMBER, L_RECEIPTDATE, L_COMMITDATE,
L_SUPPKEY, L_RETURNFLAG, L_LINESTATUS, L_SHIPINSTRUCT,
L_COMMENT, L_SHIPMODE, L_PARTKEY, L_SHIPDATE)

$LINEITEM_4$ = (L_ORDERKEY, L_EXTENDEDPRICE, L_DISCOUNT,
L_PARTKEY, L_SUPPKEY, L_TAX, L_RETURNFLAG,

L_LINESTATUS, L_QUANTITY, L_SHIPINSTRUCT, L_SHIPMODE)

$LINEITEM_5$ = (L_LINENUMBER, L_EXTENDEDPRICE, L_DISCOUNT,
L_PARTKEY, L_SUPPKEY, L_TAX, L_RETURNFLAG, L_LINESTATUS,
L_QUANTITY, L_SHIPINSTRUCT, L_SHIPMODE)

$ORDERS_1$ = (*O_ORDERKEY*, O_ORDERSTATUS,
O_ORDERPRIORITY, O_SHIPPRIORITY, O_CLERK, O_CUSTKEY)

$ORDERS_2$ = (O_COMMENT, O_ORDERDATE, O_ORDERPRIORITY,
O_SHIPPRIORITY, O_CLERK, O_ORDERSTATUS, O_CUSTKEY)

$ORDERS_3$ = (O_TOTALPRICE, O_ORDERSTATUS, O_ORDERPRIORITY,
O_SHIPPRIORITY, O_CLERK, O_CUSTKEY)

Table A.1: Sample Entries for the Set of Security Dependent Sets in TPC-H Schema

Dependency Id	**Attribute**
1	C_NAME
1	L_EXTENDEDPRICE
2	C_ADDRESS
2	L_EXTENDEDPRICE
3	C_PHONE
3	L_EXTENDEDPRICE
...	...

5) Security Dependent Sets and Precomputed Chains for TPC-H 2.17.1 Database Schema

Table A.2: Sample Entries for the Precomputed Chains

Dep. Id	**Chain Id**	**Rule Id**	**Attribute-1**	**Attribute-2**
1	1	1	C_NAME	C_CUSTKEY
1	1	2	C_CUSTKEY	O_CUSTKEY
1	1	3	O_CUSTKEY	O_ORDERKEY
1	1	4	O_ORDERKEY	L_ORDERKEY
1	1	5	L_ORDERKEY	L_LINENUM.
...	...	...	...	...

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Turan,Uğur
Nationality: Turkish (TC)
Date and Place of Birth:07.06.1984, Ankara-Turkey
Marital Status: Married (Pelin Turan, 2016)
Phone: 0 312 2357036

EDUCATION

Degree	Institution	Year of Graduation
M.S.	Department of Computer Engineering, METU	2009
B.S.	Department of Computer Engineering, METU	2006

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2005-2009	TURKTRUST Corp.	Project Engineer/Manager
2009-2010	INVICTA Ltd.	Research Engineer/Manager
2010-2011	STIGMA Ltd.	Research Manager
2011-2014	TILYA Ltd.	CTO
2014-...	TETA-TEK Electronics Corp.	CTO

PUBLICATIONS

International Conference Publications

Uğur Turan, İsmail Hakkı Toroslu, Murat Kantarcıoğlu, Secure logical schema and decomposition algorithm for proactive context dependent attribute based inference control, Data & Knowledge Engineering, Volume 111, 2017, Pages 1-21, ISSN 0169-023X