Analysis and Implementation of a Method Resistant to Functional Dependency Attacks on Databases with Sensitive Records

Gnokam Fotso Flavien

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
January 2020
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Alexander Chefranov
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Zeki Bayram
2. Assoc. Prof. Dr. Alexander Chefranov
3. Asst. Prof. Dr. Öykü Akaydın

ABSTRACT

The technology evolution has helped to develop large database management systems. Certain information due to its importance is qualified as sensitive with the help of security constraints (SC). Basic encryption method (BEM) encrypts sensitive cells in respective sensitive records. But It does not guarantee security because of possible data dependencies between attributes that may be used for functional dependency attack (FDA) with the help of evidence records having the same values of left-hand side attributes of functional dependencies defining right-hand side sensitive cells. Partial encryption method (PEM) in addition to sensitive cells encrypts also some attributes of functional dependences to resist FDA. These methods are investigated in the thesis, and some problems of PEM are revealed (double encryption, absence of ordering of FDs after finding minimal attribute cover (MAC)). Its modification, PEM-M, eliminating double encryption and ordering FDs according to MAC is proposed. Methods PEM and PEM-M are implemented using Windev 17 platform, where a user can load a database with any scheme automatically recognized, define its security constraints and functional dependencies. Then, the methods transform the database to a form resistant to FDA. Implementation was tested on a number of examples. Efficiency of the methods was studied on a benchmark Adults database used originally for testing PEM by their authors. PEM-M was tested in the same way but using Test database. Some experiments were done using 100 and 32K records of Test database in PEM and PEM-M in order to compare efficiency and accuracy to see which method performs better. It appear that in term of execution time, PEM can performs better with scores of 0.311 and 859.13 seconds for 100 and 32K records respectively comparing to 0,345 and 952.55 seconds for PEM-M. But, in term of accuracy, PEM-M performs better with 0% of risk of double encryption which is not the case for PEM.

Keywords: Database management system, Security constraint, Sensitive cell, Sensitive record, Basic encryption method, Functional dependency, Functional dependency attack, Evidence record, Partial encryption method.

ÖZ

Teknoloji geliştirme, büyük veritabanı yönetim sistemlerinin geliştirilmesine yardımcı olmuştur. Önemi nedeniyle, bazı bilgiler güvenlik kısıtlamaları (SC) yardımıyla hassas kabul edilmektedir. Temel şifreleme yöntemi (BEM), hassas hücreleri ilgili hassas kayıtlarda şifreler. Bununla birlikte, sağ tarafa duyarlı hücreleri tanımlayan fonksiyonel bağımlılıkların sol taraf özelliklerinin aynı değerlerine sahip bazı kanıt kayıtlarının yardımıyla, fonksiyonel bağımlılık için mevcut özellikler arasındaki olası veri bağımlılıkları nedeniyle güvenliği garanti etmez saldırı (FDA). Kısmi şifreleme yöntemi (PEM), hassas hücrelere ek olarak, FDA'ya karşı koymak için fonksiyonel bağımlılıkların bazı özelliklerini de şifreler. Bu yöntemler tezde incelenir ve bazı PEM problemleri ortaya çıkar (çift şifreleme, minimal bir öznitelik kapağı (MAC) bulduktan sonra FD'ler sıralanmaz). Modifikasyonu olan PEM-M, çift şifrelemeyi ortadan kaldırır ve FD'leri MAC ile sıralar. Yöntemler PEM ve PEM-M, kullanıcının otomatik olarak tanınan herhangi bir şema ile bir veritabanı yükleyebileceği, güvenlik kısıtlarını ve işlevsel bağımlılıkları tanımlayabileceği Windev 17 platformu kullanılarak uygulanır. Yöntemler daha sonra veritabanını FDA'ya dayanıklı bir forma dönüştürür. Uygulama birkaç örnek üzerinde test edilmiştir. Yöntemlerin etkinliği, başlangıçta PEM'yi yazarları tarafından test etmek için kullanılan bir karşılaştırmalı Yetişkin veritabanında incelenmiştir. PEM-M de Test veritabanı kullanılarak test edilmiştir. Hangi yöntemin daha iyi performans gösterdiğini görmek için verimliliği ve doğruluğu karşılaştırmak amacıyla Test veritabanının 100 ve 32K kayıtları kullanılarak PEM ve PEM-M'de bazı deneyler yapıldı. Yürütme süresi açısından, PEM, 100 ve 32K kayıtları için sırasıyla 0.311 ve 859.13 saniye puanlarıyla PEM-M için 0.345 ve 952.55 saniyeden daha iyi performans gösterebilir. Bununla birlikte, doğruluk açısından, PEM-M, PEM için uygulanamayan% 0 çift şifreleme riski ile daha iyi performans gösterir.

Anahtar Kelimeler: Veritabanı yönetim sistemi, Güvenlik kısıtı, Hassas hücre, Hassas kayıt, Temel şifreleme yöntemi, Fonksiyonel bağımlılık, Fonksiyonel bağımlılık saldırısı, Kanıt kaydı, Kısmi şifreleme yöntemi.

.

DEDICATION

Dedicated this report to my family for their support

ACKNOWLEDGMENT

I would first like to thank God for his benefits in my life, and to have accompanied me throughout this training.

I would also like to thank all the EMU's staff and in particular Assoc. Prof. Alexander Chefranov; for guiding and helping me to drive this project to completion.

Finally, I would like to thank my family who has given me unconditional support so that I can live these unforgettable moments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

FD	Functional Dependency
FDA	Functional Dependency Attacks
MAC	Minimum Attribute Cover
MLS	Multi-level Security
PEM	Partial Encryption Method
PEM-M	Partial Encryption Method Modified
SC	Security Constraint

Chapter 1

INTRODUCTION

1.1 General Overview

One of the most important aspects of technological advances is the secure management of databases Cryptographic techniques tries preventing potential attacks on databases [1] [2]. Partial encryption method is developed to allow different users to have access according to the rights they have [1].

Access to information in databases is usually regulated by different levels of security [2] [3]. Thus, for a user A who has higher security level than user B, it can get access to information present in the level of B but the inverse is not possible [4]. However, functional dependency (FD), showing relationship between attributes, in the form of A→B (where A and B are subsets of a database attributes, meaning that if any two tuples in the database have the same values of attribute from A, they also have the same values of their attributes B), can lead to inferring from the level of B, the information contained in the level of A. It is therefore said that database can be attacked [1] using functional dependency. A general solution to solve that issue was to encrypt all data in the database but, in the case of Data Base as A Service (DAS) where there is a large volume of data [1]; it's a pretty heavy process. The appropriate solution is therefore a partial encryption based on security constraints (SC).

Example 1: Original database is shown in Table 1 and basic encryption in Table 2 [1]

Table 1: Original Table

NM	SEX	AGE	DC	DS
Alice	F	53	CPD5	HIV
Carol	F	30	VPI8	Breast Cancer
Ela	F	24	VPI8	Breast Cancer

Table 2: Basic Encryption

NM	SEX	AGE	DC	DS
Alice	F	53	CPD5	HIV
Carol	F	30	VPI8	α
Ela	F	24	VPI8	Breast Cancer

Example 1 shows a database with three patients who are registered with their name (NM), sex (SEX), age(AGE), disease code (DC), and disease (DS) in the original database, Table 1. Alice, Carole and Ela all have illnessyi. Since all patients have access to this database, some of them would not want their disease to be publicly available; this request for confidentiality is represented by a security constraint (SC). As it is shown, after encrypting DS attribute for Carol in Table 2 (Breast Cancer is encrypted by α) which represents here a sensitive cell [1], it is still possible to infer her DS based on the FD between DC and DS (DC→DS), since evidence record for Ela has the same DC as Carol has, and thus, basic encryption of the database does not resist FD attack (FDA). In fact, FD existence can represent a way by which confidential information can be attacked throughout evidence record [1] which is the record that shows the same disease code but non encrypted disease label (DS). Thus, somebody who knows FD: DC→DS can easily infer that Carol's disease is Breast Cancer because Carol and Ela tuples have the same DC values (VP18) and by virtue of the FD their DS attributes shall be also equal.

The idea here is to develop a system that can transform an input database to an output database resistant against FDAs with a minimal number of encrypted information.

1.2 Database Security Concerns

Database security has become one of the most important issues in database management. With the creation of Client/Server techniques, it is now possible to detect vulnerabilities in a system and attack its databases [4] [2].Various database protection techniques have been developed.

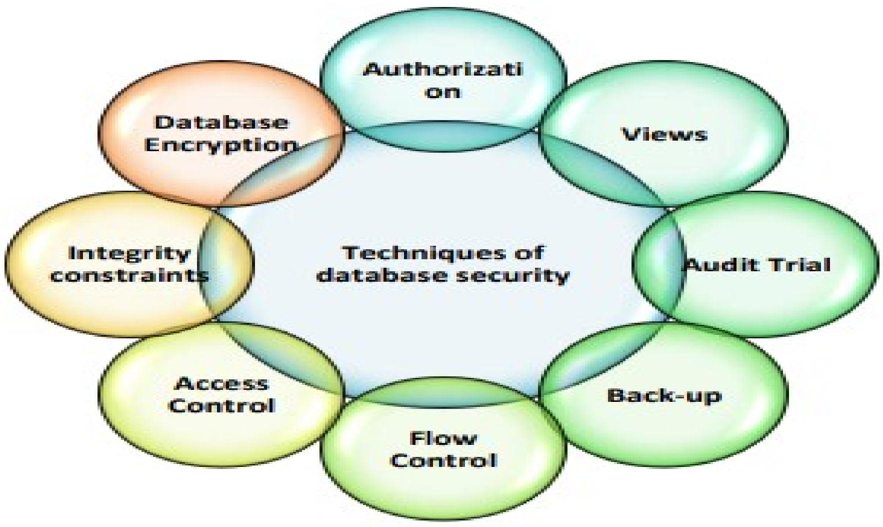


Figure 1: Various techniques for database security [4]

From the techniques shown in Figure 1, Database encryption and integrity constraints first of all will be described, and will be used to build a partial encryption system with the purpose of making database resistant to FD attacks [1].

Chapter 2

RELATED WORK AND PROBLEM DEFINITION

In this part, will be defined and explained the key concepts of the work done and discuss about existing problem.

The partial encryption method (PEM) is used in [1] with the goal of resisting functional dependency attacks. It aims reducing the number of encryptions. PEM uses concepts of database, functional dependency, security constraint, and some other. Experiments were conducted with PEM [1] implemented in Java using two datasets (Adults [5] and Order) and obtained results show estimates of execution time with different number of records. GMM algorithm created for attacks problems and optimal solution were used in the implementation approach and it appears that using both datasets, different number of functional dependencies and different number of security constraints, results presented in Table 3 were obtained.

Table 3: Obtained Results in PEM [1]

Data	n	m	f	t	w
Adult_32K	32,000	15	78	3877.4	1.016
Adult_64K	64,000	15	78	3989.7	2.03
Adult_128K	128,000	15	78	3990.4	4.059
Adult_256K	256,000	15	78	3988.9	8.129
Orders_0.3M	300,000	9	10	125.8	126.82
Orders_0.6M	600,000	9	10	191.1	306.51
Orders_0.9M	900,000	9	10	229.7	383.12
Orders_1.2M	1,200,000	9	10	259.6	459.99
Orders_1.5M	1,500,000	9	10	288.7	508.37

To understand how this method works, concepts used will be described and analyzed in the following sections.

2.1 Database Concepts

2.1.1 Definition

Nowadays, database can be defined as an organized collection of data, generally stored and accessed electronically from a computer system [6] but, before the creation of computers, data storage facilities already existed.

2.1.2 Brief History of Databases

The main idea about a database system is to store data. In early computer era, information storage was already observed in hospitals, administrative offices and some enterprises. In 1960s, with the technological improvement the first electronic database was developed [7]. This technology evolved and in the 1970s the first relational databases were created, and many improvements made have resulted in the existence of several types of databases to date.

2.1.3 Structure of Database

The goal of a database is to allow users to manipulate data quickly and reliably. A database must therefore be well structured for this purpose. There are several types of database, but relational databases will be used for this work. A relational database is defined as follows:

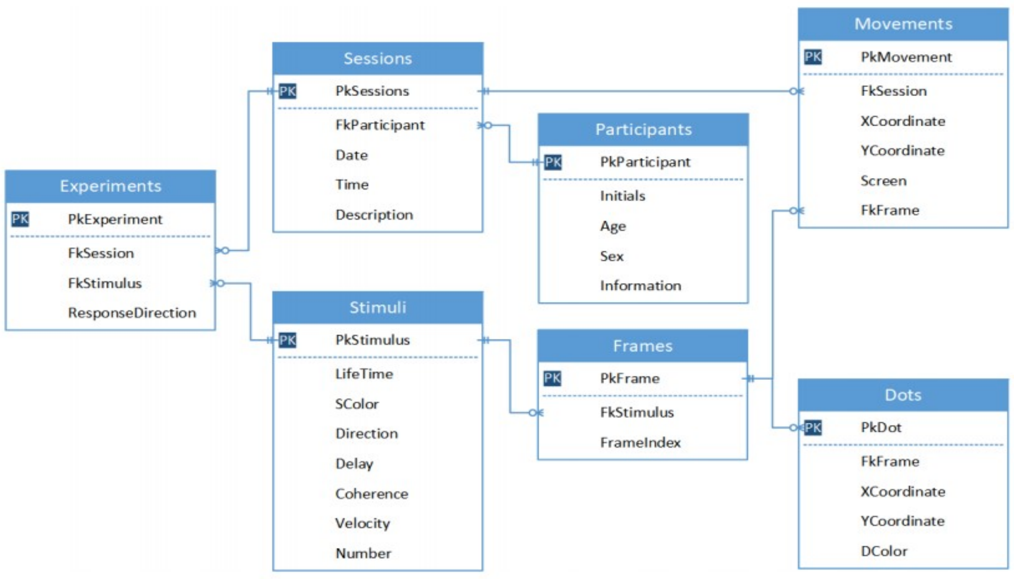


Figure 2: Structure of a Relational Database: " Modeling of voluntary saccadic eye movement during Decision Making" (Mvsemdm) [8]

Figure 2 shows the Mvsemdem relational database where tables can be easily seen, represented by boxes with names shown in their headers, attributes shown in the boxes, and relationship among tables shown as links between the boxes. Consider for example table Participants, where it is shown five attributes (**PKParticipant**, Initials, Age, Sex, Information). PKParticipant is a primary key (specific choice of a minimal set of attributes that uniquely specify a tuple in a table [8] (it is shown in bold and underlined). Thus, using PKParticipant, other tuple having a particular value of the attribute in the table can be uniquely determined. A primary key is also used for creating relationship between two tables [8] [9]. In that case, cardinality (In the context of databases, cardinality refers to the uniqueness of data values contained in a column. High cardinality means that column contains a large percentage of totally unique values. Low cardinality means that column contains a lot of "repeats" in its data range.), will allow one of the two primary keys to move to the second table and be a foreign key there. Consider for example a relationship between table Stimuli (**PKStimulus**, Lifetime, Scolor, Direction, Delay, Coherence, Velocity, Number) and table Frame (**PKFrame**, FKStimulus*, Frameindex). It is noticed that PKStimulus moved to table Frame, and there, instead of PKStimulus, FKStimulus is used but as a foreign key.

2.2 Encryption

2.2.1 Definition

Encryption can be defined as a process of hiding information so that to access it, a user needs special knowledge [10]. In fact, there are two techniques used to hide information using encryption, symmetric and asymmetric techniques. In this paper a symmetric technique named Vigenere will be used for encryption.

2.2.2 Vigenere Encryption Method

Vigenere method is defined as an alphabet encrypting method which uses a series of interwoven Caesar cipher [11][12]. Algebraically, going from A to 9 in the alphabet, a number will be attribute for each character starting from 0 to 35, for example A≘0, B≘1,...,9≘35, and since numbers go up to 35, addition will be perform with this. Then, if Tex considered as plaintext and K as key, Vigenere of Tex names cipher text Ct will be $Ct=Tex_k(Nt)=(Nt+Kt)Mod37$, where Nt is the number attributed to the character in the plaintext and Kt the number attributed to the character in the key.

Example 2: Let consider Tex=SAME and key=KEY, if starting from A to 9, S≘18, A≘0, M≘12, and E≘4. For the key, K≘10, E≘4 and Y≘24. Thus,

C(S) = (18+10) mod 36 = 28 => 2

C(A) = (0+4) mod 36 = 4 => E

C(M) = (12+24) mod 36 = 0 => A

C(E) = (4+10) mod 36 = 14 => O

So Vigenere cipher encryption of “SAME” is “2EAO” using key=”KEY”

2.3 Multi-Level Security

A system with multi-level security is a system with different levels of access [10].Thus, if A and B are two attributes, SL(A) security level of A and SL(B) security level of B, SL(A) > SL(B) means security level of A is higher than security level of B and consequently a user in a level of B cannot get access to information to the level of A. Therefore due the fact that databases are relational, and FDs among attributes interact, it could be attacked by FDA [12]. To prevent potential FD attacks, the approach is to avoid presence of compromise FDs. It has been shown that a FD can yield derivatives, and it is important to make sure that the derived FDs are also safe. So, Lemma 3.1 has shown that when FD set is safe, its derivatives, FD set closure, is also safe [13], If F is a set of functional dependencies FD, F^+ denotes the set of derivatives called a closure. Lemma 3.1 is as follows:

Lemma 3.1: For the set of functional dependencies, F= $\{FD_1, FD_{2,...}, FD_n\}$, defined on the database scheme R, if all FD=A→B $\in$ F, if SL is a security level and $SL(A_i) \geq SL(B_i)$ with $A_i \in A$ and $B_i \in B$ then, there does not exist an $\boldsymbol{FD} \in \boldsymbol{F}^+$ compromissing the database scheme R [13].

Technically, for ML security, two rules, “No-read up” and “No-write down”, shall be provided, meaning that a subject with the lower security level (SL) cannot read a document with the higher SL and a subject with the higher SL cannot write into a document with the lower SL respectively. Therefore, data inference can be responsible of FD compromise. To clearly understand what is FD compromise and how to fix it, let us consider Example 3.

Example 3: Compromise relationship and fixing method

Let the set of attributes R = {A, B, C, D}, SL is a security level, and SL(A) > SL(B) > SL(C) > SL(D). A FD compromises the database with the scheme R, when attribute with the bigger security level represents the right hand side of the relation. So, D→A is a compromising FD, but A→B is not. To fix the compromising issue, security level of D should be increased to be at the same level with A.

That was the idea about Multi-level security and how it works.

2.4 Basic Encryption Method

Basic encryption method (BEM) [1] is used for encryption of sensitive cells. Its goal is to hide sensitive information defined by security constraints.

2.4.1 Security Constraints

The security constraints represent conditions used to restrict the level of access to the data. It is then said constraints make it possible to further restrict a domain of an attribute [8]. To be clearer and show the rule of constraint in the process of basic encryption, consider Table 4 and Table 5.

Table 4: Original Table of D

TID	A	B	C
1	a_1	b_1	c_1
⋮	⋮	⋮	⋮
999	a_1	b_1	c_2
1000	a_1	b_1	c_2
1001	a_1	b_1	c_3
1002	a_2	b_2	c_3
⋮	⋮	⋮	⋮
2000	a_2	b_2	c_3

FD: A→B, SC$_1$: $\Pi_B \sigma_{C=c_1}$,

SC$_2$: $\Pi_B \sigma_{C=c_2}$

Table 5: Basic Encryption $\overline{D}$ of D

TID	A	B	C
1	a_1	$\boldsymbol{\beta_1}$	c_1
⋮	⋮	⋮	⋮
999	a_1	$\boldsymbol{\beta_1}$	c_2
1000	a_1	$\boldsymbol{\beta_1}$	c_2
1001	a_1	b_1	c_3
1002	a_2	b_2	c_3
⋮	⋮	⋮	⋮
2000	a_2	b_2	c_3

Given database with a scheme having a set of three attributes {A, B, C}where B is functionally depending on A (FD: $A \rightarrow B$), Table 4 is the original table and there are two thousand rows conditioned by two security constraints (SCs). The first one SC_1: $\Pi_B \sigma_{C=c1}$ is a projection on attribute B and selection of attribute C with value c1requests, which means necessity of encryption of attribute B in the tuples where $C=c_1$, and the second one SC_2:$\Pi_B \sigma_{C=c2}$ is the same meaning but the tuples shall be with $C=c_2$.

2.4.2 Basic Encryption, Sensitive Cells and Sensitive Records

If Table 4 and Table 5 are considered again, it is noticed in Table 5 that values of r[B] are encrypted in the rows where $C=c_1$ and $C=c_2$ respectively. To be clearer, it is easily seen that $\bar{\boldsymbol{D}}$ is obtained after applying security constrains SC_1 and SC_2, and in which values of B are encrypted in rows 1, 999 and 1000. The process is called basic encryption and respective cells to be encrypted are called sensitive cells. Sensitive records are then all the rows containing sensitive cells [1]. So in general, given a set of data D with attributes A, B and C, security constraint $\Pi_B \sigma_{C=c}$, sensitive cells are all cells r[B] where r[C]=c and sensitive records are all records where r[C]=c.

2.5 Functional dependency

Functional dependency (FD) denotes constraint between attributes. As explained in Section 2.2.3, relationship can exist between attributes and in the context of functional dependency, if given functional dependency FD: $A \rightarrow B$ with A and B two sets of attributes, B is functionally depending on A, which means A can uniquely determine B [1]. A is left hand side (LHS) attribute and B is right hand side attribute (RHS). Sensitive data defined by security constraints can be disclosed using FD: if LHS are the same then it can be inferred that RHS are also equal [12].

2.6 Evidence Records

Evidence here represents the flaw by which system can be attacked [1]. For a dataset D with attributes A and B, FD: A $\rightarrow$B, L = {SC$_1$, SC$_2$...SC$_n$} list of constraints and $\bar{\boldsymbol{D}}$ dataset with basic encryption, it appears that $\bar{\boldsymbol{D}}$ has an evidence record if for a row r with r[B] cipher text and r[A] plaintext, there is any record r' $\in \bar{\boldsymbol{D}}$ where r'[A] = r[A] and r'[B] is not encrypted [1]. To be more explicit, take into account tables in Example 4.

Example 4: Example of detection of evidence record [1] (Table 6, 7)

Table 6: Original Table of D

TID	A	B	C
r_1	a_1	b_1	c_1
r_2	a_1	b_1	c_2
r_3	a_1	b_1	c_3
r_4	a_2	b_2	c_3

Table 7: Basic Encryption of $\bar{D}$ of D

TID	A	B	C
r_1	a_1	$\boldsymbol{\beta_1}$	c_1
r_2	a_1	b_1	c_2
r_3	a_1	b_1	c_3
r_4	a_2	b_2	c_3

As evidence record was described in the definition, Example 4 show Table 6 and Table 7 which are respectively original table and basic encryption. In Table 7, it is noticed that r_1[B] is encrypted after applying security constraint SC$=\Pi_B\sigma_{C=c1}$.

Now, in application of what was described before, considering $\bar{\boldsymbol{D}}$, functional dependency FD: A$\rightarrow$B, and r1[B]=β1 which is an encrypted value. It appears that r_1[A]=a_1 is not encrypted. By checking into others rows, it is also noticed in rows r_2 and r_3 that r_2[A]=r_3[A]=r_1[A]=a_1 (Not encrypted value), and r_2[B]=r_3[B]=b_1 (Not encrypted value) thus, based on r_2, r_3 and FD β1=b1 can be easily inferred and consequently r_2 and r_3 are called evidence records.

2.7 Functional Dependency Attacks

Since sensitive and evidence records were defined, it’s easy to understand the principle of FD attack. In fact, a system that leaves the gaps is not robust and can be attacked. Attacks are made in a dataset which from one or more properties can reveal certain secret information. It means for a dataset D and his basic encryption$\overline{\boldsymbol{D}}$, D can be attacked if it exists any evidence record in $\overline{\boldsymbol{D}}$. Then, some steps should be followed to check if a system can be attack or not:

- Check if it exists any sensitive record: As sensitive record was defined, this step aims to detect in $\overline{\boldsymbol{D}}$, if it exists any record r'_1 where $r'_1[A]$ is a plaintext and $r'_1[B]$ is a cipher text.

- Check if it exists any evidence record: since sensitive record is detected, the next step is to verify if it exists any other record r'_2 where $r'_2[A]$ and $r'_2[B]$ are plaintext, and $r'_1[A]=r'_2[A]$

If these situations are found, conclusion is that $\overline{\boldsymbol{D}}$ is not robust and can be attacked. There is a way to fight against functional dependency attacks by making system robust to potential attacks.

2.8 Partial Encryption Method (PEM)

The idea here is to check existence of gaps in the system and prevent it for being attacked. As discussed before, a dataset which has evidence records in basic encryption is not robust and needs to be secure in order to be robust for attacks. To perform it, some steps need to be followed as shown in the flowchart.

2.8.1 Flowchart of PEM

The flowchart will represent steps to perform PEM. It means he shall start after basic encryption and the whole process of the method combining basic encryption and partial encryption will be represent in Section 4.

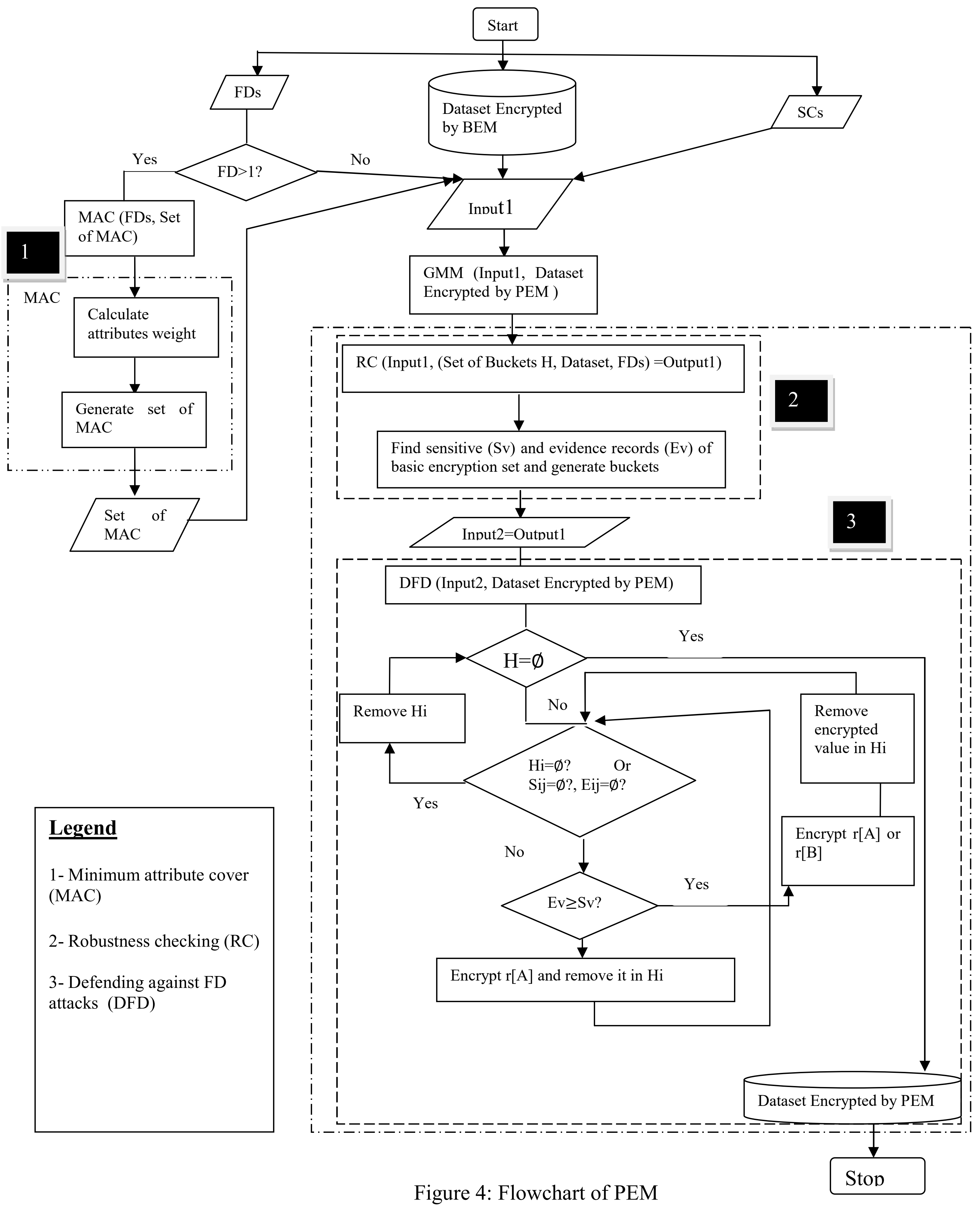

Start
FDs
Dataset Encrypted by BEM
SCs
Yes
FD>1?
No
MAC (FDs, Set of MAC)
Input1
1
MAC
Calculate attributes weight
Generate set of MAC
Set of MAC
GMM (Input1, Dataset Encrypted by PEM)
RC (Input1, (Set of Buckets H, Dataset, FDs) =Output1)
2
Find sensitive (Sv) and evidence records (Ev) of basic encryption set and generate buckets
3
Input2=Output1
DFD (Input2, Dataset Encrypted by PEM)
Yes
H=∅
Remove Hi
No
Remove encrypted value in Hi
Hi=∅? Or Sij=∅?, Eij=∅?
Yes
Encrypt r[A] or r[B]
No
Yes
Ev≥Sv?
Encrypt r[A] and remove it in Hi
Dataset Encrypted by PEM
Stop
Legend
1- Minimum attribute cover (MAC)
2- Robustness checking (RC)
3- Defending against FD attacks (DFD)

Figure 4: Flowchart of PEM

2.8.2 Robustness Checking

In the easier way, robustness checking means check if there does not exist any evidence record when sensitive records exist like précised in (2) in the flowchart. Thus, for a dataset D and its basic encryption $\overline{\boldsymbol{D}}$, one security constraint such that B is sensitive, and one FD: A→B, to verify whether the system is robust, two conditions related to Lemma 4.1 [1] should be considered, that shall hold for each record r where r[B] is sensitive.

Lemma 4.1 conditions:

- Condition 1: There exists at least one attribute $\boldsymbol{Z} \in \boldsymbol{A}$ such that $\boldsymbol{r}[\boldsymbol{Z}]$ is encrypted.

- Condition 2: If r[A] is not encrypted, and there does not exist a record r' where r'[A] =r[A] and r'[B] is not encrypted.

If those conditions are respected, consequently system is robust. Consider the following example to prove the previous conditions of Lemma 4.1 [1].

Example 5: Example of robustness checking after basic encryption [1].

Table 8: Original Table of D

TID	A	B	C	D
r_1	a_1	b_1	c_1	d_1
r_2	a_1	b_1	c_1	d_2
r_3	a_1	b_2	c_2	d_2
r_4	a_2	b_1	c_1	d_3
r_5	a_2	b_1	c_1	d_1

Table 9: Basic Encryption$\overline{D}$ of D

TID	A	B	C	D
r_1	a_1	b_1	γ_1	d_1
r_2	a_1	b_1	γ_1	d_2
r_3	a_1	b_2	c_2	d_2
r_4	a_2	b_1	γ_1	d_3
r_5	a_2	b_1	γ_1	d_1

This example is used to illustrate the conditions mentioned above., For a dataset D represented by Tables 8, which contains four attributes {A, B, C, D}, one functional dependency FD: {A, B}$\rightarrow$ C and one security constraint SC: $\Pi_C \sigma_{B=b1}$. The result after basic encryption, based on security constraint is represents in Table 9. It appears that rows r_1, r_2, r_4 and r_5 were selected where b=b_1, and projection on cells r_1[C], r_2[C], r_3[C] and r5[C] was done in order to get them encrypted as it is seen in Table 9. Since $\overline{\boldsymbol{D}}$ is table of basic encryption and represents by Table 9, those conditions of Lemma 4.1 should be applied on it to verify if system is robust or not. For the first condition, it exists at least one row in $\overline{\boldsymbol{D}}$ where r[X] is not encrypted and r[Y] is encrypted. They can be verified in rows r_1, r_2, r_4 and r_5 where none of r[A] or r[B] is encrypted but r[C] is encrypted. For the second condition, if rows r[C]s with encrypted values are considered, the next step is to check if there exist any other row r[X] represents by r[A] and r[B] is not encrypted, and r[Y] represents by r[C] is also not encrypted. Going through Table 9, there is not such kind of row which means $\overline{\boldsymbol{D}}$ is robust.

2.8.3 Defending Against Functional Dependency Attacks

A system can defend itself against FD attacks when he is robust. Thus, the idea behind defending against FD attacks is to make the system robust by encrypting the non-sensitive cells which can represent the way throughout the system can be attack [1] [2]. The process to make a system robust depends on some parameters such as number of functional dependency, and number of security constraints. The process with one security constraint will be the first case of the description.

2.8.3.1 Case of One Security Constraint

As it was said before, it is not possible to talk about sensitive records if it does not exist security constraint, and don't talk about attacks if there is no evidence records. Thus, the process on how to check the existence of evidence records and fix the gaps for a case of single security constraint will be explained. Since security constraint is a selection query, consider SC: $\Pi_B \sigma_C$ where B is the right hand side (RHS) of the functional dependency FD: A$\rightarrow$B. The idea here is to get based on FD and SC all the attributes which are concerning in the process of encryption. Then, security constraint will be transformed so that left hand side (LHS) and (RHS) will be getting in the same selection, by applying security constraint. Then, SC=$\Pi_{(A,B)} \sigma_C$ which means select (A,B) where condition C is respected will be the transformation result and after that, the next step is to divide it in bucket in order to get a unique sensitive cell for each bucket and it matching A value. The process is then followed by applying two selections. One selection Ss: $\Pi_{TID} \sigma_{(A=a,B=b) \cup C}$ with C as selection condition to get sensitive records, and another selection Se: $\Pi_{TID} \sigma_{(A=a,B=b)}$ to get all the records which have the same attribute A with the ones considering as sensitive records. As soon as sensitive records are obtained after basic encryption, there are two cases:

- First case : There are no evidence records

  In this case, conclusion is that dataset is robust and cannot be attacked.

- Second case : There is at least one evidence record

In this case, the problem should be fixed by encryption, and to do that, two possibilities (Local and Global solutions) are envisaged [1].

-1st Possibility (Local Solution): For each sensitive record r, pick A ∈ LHS and encrypt r[A] in the case that there is only one attribute in the LHS, or randomly pick A ∈ LHS and encrypt r[A] in the case of many attributes in LHS.

-2nd Possibility (Global Solution): For each evidence records r', which is representing by Se(D)-Ss(D) and means the attributes which are present in Se(D) but not in Ss(D), select A or B and encrypt r'[A] or r'[B]. Take into account Example 6 to be clearer.

Example 6: Example of solving the problem of evidence records with one constraint.

Table 10: Original Table D

TID	A	B	C
r_1	a_1	b_1	c_1
r_2	a_2	b_2	c_2
r_3	a_1	b_1	c_3
r_4	a_3	b_3	c_4
r_5	a_4	b_4	c_5

Table 11: Basic Encryption $\overline{D}$ of D

TID	A	B	C
r_1	a_1	$\boldsymbol{\beta_1}$	c_1
r_2	a_2	b_2	c_2
r_3	a_1	b_1	c_3
r_4	a_3	b_3	c_4
r_5	a_4	b_4	c_5

Let FD: A→B, and SC: $\Pi_B\sigma_{C=c1}$. Table 10 is representing original table and Table 11 Basic encryption. As explained before, the first thing to do is to check if Table 11 with basic encryption is robust or not. Based on the result, it is shown that $\boldsymbol{\overline{D}}$ is not robust due to the fact that sensitive record r_1 has an evidence record r_3 and need to be fixed. To fix it, those steps need to be followed:

- Transformation of $\Pi_B \sigma_{C=c1}$: Rewrite the query so that all attributes concerning by the FD during the selection. Then rewriting result is $\Pi_{(A,B)} \sigma_{C=c1}$.

- The next step is to get our buckets, for the first one, apply the following query $\Pi_{TID} \sigma_{(A=a,B=b)Uc1}$ to have bucket $B_1(a_1,b_1,c_1)$ which will generate sensitive record Ss(D)={r_1}. For the second query $\Pi_{TID} \sigma_{(A, B)}$, bucket $B_2(a_1, b_1)$ is obtained and it contains Se(D)={r_1, r_3}. Solving possibilities can now be applying to fix the problem.

- Solving with first possibility (Local Solution) : If r[A] in Ss(D) is encrypted, the obtained result is like in Table 12.

Table 12: Result after Applying Local Solution (Robust)

TID	A	B	C
r_1	$\boldsymbol{\alpha_1}$	$\boldsymbol{\beta_1}$	c_1
r_2	a_2	b_2	c_2
r_3	a_1	b_1	c_3
r_4	a_3	b_3	c_4
r_5	a_4	b_4	c_5

- Solving with the second possibility (Global Solution): If Se(D)-Ss(D) is applied, {r_3} is obtained so, r[A] or r[B] will be encrypted in r_3 to get result in Table 13.

Table 13: Result after Applying Global solution (Robust)

TID	A	B	C
r_1	a_1	β_1	c_1
r_2	a_2	b_2	c_2
r_3	a_1	β_1	c_3
r_4	a_3	b_3	c_4
r_5	a_4	b_4	c_5

It is noticed in Table 12 and Table 13 after applying both cases that tables are robust, and in this example Ss(D) and Se(D)-Ss(D) have the same number of encryption overhead which is one. Note that encryption overhead means total number of encryption after basic encryption. Therefore, it might happen than one of the two solutions needs more encryptions than other, therefore the best solution is the one which has a minimum number of encryption. To decide on which solution has a less number of encryption, the number of records where cell has to be encrypted should be considered thus, if Nev is a number of evidence records in the first solution and Ns=Se(D)-Ss(D) the number of records to encrypt in the second solution, to select the best solution to apply, Nev and Ns should be compared in order to apply the solution with the minimal number of record to encrypt after basic encryption. In other words, it means, get Min (Ns, Nev) and apply the result corresponding to the minimal value.

After getting how to fix the problem of robustness with one security constraint, let see how to solve with multiple security constraints (MSCs)

2.8.3.2 Case of Multiple Security Constraints (MSCs)

MSCs mean existence of more than one security constraint [3]. In such kind of case, the solving process that is used for one security constraint can be used in each security constraint [1] to solve the problem. In others words, if

$\mathcal{L}$ = {SC_1, SC_2,...SC_n}is the set of security constraints, for each security constraint from 1 to n, the same solving process used to get solution with one security constraint will be followed. Therefore, to optimize the solution, the less number of overhead encryption should be considered.

Example 7: Solution with MSCs (Proposed example).

Let D a set of data and $\overline{\boldsymbol{D}}$ basic encryption representing by the following tables

Table 14: Original Table of D

TID	A	B	C
r1	a_1	b_1	c_1
r2	a_1	b_1	c_1
r3	a_1	b_1	c_3
r4	a_1	b_1	c_2
r5	a_1	b_1	c_4

Table 15: Basic Encryption $\overline{D}$ of D

TID	A	B	C
r1	a_1	$\boldsymbol{\beta_1}$	c_1
r2	a_1	$\boldsymbol{\beta_1}$	c_1
r3	a_1	b_1	c_3
r4	a_1	$\boldsymbol{\beta_1}$	c_2
r5	a_1	b_1	c_4

With one FD: A→B, two security constraints $SC_1=\Pi_B\sigma_{C=c1}$ and $SC_2=\Pi_B\sigma_{C=c2}$, Table 14 as original table and Table 15 as basic encryption, for SC_1, r_1 and r_2 are sensitive records because of $r_1[B]$ and $r_2[B]$ which are encrypted, and for SC_2 r_4 is sensitive record because of $r_4[B]$ which is encrypted also. If local optimal solution is applied, for SC_1, r_3 and r_5 are evidence records, and total of sensitive record=total of evidence records=2, so r[A] can be encrypted to fix the problem, which will give a total of 2 encryptions $r_1[A]$ and $r_2[A]$ as it is seen in Table 13 below. For SC_2, r_3 and r_5 are also evidence records, but since total sensitive record<total evidence record which is 1<2, r[A] will also be encrypted to fix the problem, which will give one encryption. So, for optimal encryption, a total of 3 encryptions are obtained. If consider global solution now, total of sensitive records is 3 (r_1, r_2, r_4) and total of evidence records is 2 (r_3, r_5) then, to fix the problem, r[B] will be encrypted for each evidence record and system will get robust. Total encryption overhead is 2 (r_3[B], r_5[B]) as in Table 17.

Table 16: Result Using Local Solution (Robust with overhead=3)

TID	A	B	C
r1	**α1**	β1	c1
r2	**α1**	β1	c1
r3	a1	b1	c3
r4	**α1**	β1	c2
r5	a1	b1	c4

Table 17: Result Using Global Solution (Robust with overhead=2)

TID	A	B	C
r1	a1	β1	c1
r2	a1	β1	c1
r3	a1	**β1**	c3
r4	a1	β1	c2
r5	a1	β1	c4

In conclusion, global encryption is the best solution to solve the problem because solution is obtained (Table 17) with a minimal number of encryption. Based on what was described till now, a process of fixing FD attacks can be summarized by algorithm [1].

This algorithm is called GMM [1] and is used to go through security constraints and functional dependencies to detect and fix a database attacks problems. Its uses:

- $\mathcal{L}$ ={SC_1,SC_2,..., SCn}as a list of SCs
- FD: X→Y represents functional dependency
- V_{ij} as a sensitive cell
- Sv as a sensitive record
- Ev as an evidence record
- H_i as a bucket which contains a set in the form of (V_{ij}, Sv, Ev) for each SC

Algorithm 1 of fixing FDs attacks: GMM ($\mathcal{L} = \{S_1, S_2,...,Sn\}, X \rightarrow Y$) [1]

```
1.  For all $SC_i$ € $\mathcal{L}$ do
2.    Checking of sensitive cell { $V_{ij}$} set
3.    For all $V_{ij}$ do
4.      Find $S_{ij}$ and $E_{ij}$, sensitive and evidence records respectively
5.    end for//$V_{ij}$
6.    Let $H_i=\{(V_{ij}, S_{ij}, E_{ij})\}$
7.  End for
8.  While $Hi \neq \Phi$ for all i € 1,…,k do
9.     Let $Min_c = Min_{forallVij€Hi}Min(|S_{ij}|, |V_{ij}|)$
10.    Let $Min_v$ be the sensitive cell that deliver $Min_c$
11.    Let $S_v$ and $E_v$, sensitive and evidence records of $Min_v$
12.    If $|S_v|<= |E_v|$ then
13.      $Min_{se}= S_v$
14.      Pick randomly an attribute A € X, and encrypt A in all records $\in$ Sv
15.    Else
16.      $Min_{se} = Ev$
17.      Pick randomly attribute A € X U Y , and encrypt A in all records $\in$ ev
18.    End if
19.  For all $H_i$ do
20.    For all $(V_{ij}, S_{ij}, E_{ij})$ € $H_i$ do
21.        If Vij = $Min_v$ then
22.           $S_{ij} = S_{ij}-Min_{se}$
23.           $e_{ij} = E_{ij}-Min_{se}$
24.             If $S_{ij} = \Phi$ or Eij = $\Phi$ then
25.               Remove($V_{ij}, S_{ij}, E_{ij}$) from $H_i$
26.             End if
27.        End if
28.    End for
29.  End for//*Hi*
30.  End while
```

Consider Example 8 to see exactly how Algorithm1 works.

Example 8: Example of application of Algorithm1 [1]

Table 18: Original Table of D

TID	Name	Sex	Age	DC	DS
r_1	Joe	M	28	CPD5	HIV
r_2	Alice	F	24	CPD5	HIV
r_3	Maggy	F	33	CPD5	HIV
r_4	Phil	M	43	CPD5	HIV
r_5	Peter	M	39	CPD5	HIV
r_6	Rey	M	52	CPD5	HIV
r_7	Steve	M	31	CPD5	HIV

Table 19: Basic Encryption $\overline{D}$ of D

TID	Name	Sex	Age	DC	DS
r_1	Joe	M	28	CPD5	α
r_2	Alice	F	24	CPD5	α
r_3	Maggy	F	33	CPD5	α
r_4	Phil	M	43	CPD5	HIV
r_5	Peter	M	39	CPD5	HIV
r_6	Rey	M	52	CPD5	HIV
r_7	Steve	M	31	CPD5	HIV

For this example, consider original Table 18, FD: DC→DS and two security constraints SC_1: $\Pi_{DS}\sigma_{Age<30}$ and SC_2: $\Pi_{DS}\sigma_{Sex=\text{"F"}}$. After applying SC_1 and SC_2, basic encryption is represented by Table 19. Since at least on sensitive record exists, GMM can be applied in order to make system robust.

For SC_1, there are two sensitive records r_1 and r_2 and four evidence records r_4, r_5, r_6 and r_7, thus H_1 will be:

H_1 (Sc_1, DC→DS) = ({{r_1[DS], r_2[DS]},{r_1, r_2},{r_4, r_5, r_6, r_7}}). In the same logic, For SC_2, H_2 (Sc_2, DC→DS) = ({{r_2[DS], r_3[DS]},{r_2, r_3},{r_4, r_5, r_6, r_7}}). So, H = {H_1, H_2} is set of H evoked in step 8 of Algorithm1.

For H_1, Min (|Sv|, |Ev|) =2 which is the number of sensitive records then, based on FD, r_1[DC] and r_2[DC] should be encrypted to get the results presented in Table 20.

Table 20: Result after 1st Encryptions Using H_1 [1]

TID	Name	Sex	Age	DC	DS
r_1	Joe	M	28	β	α
r_2	Alice	F	24	β	α
r_3	Maggy	F	33	CPD5	α
r_4	Phil	M	43	CPD5	HIV
r_5	Peter	M	39	CPD5	HIV
r_6	Rey	M	52	CPD5	HIV
r_7	Steve	M	31	CPD5	HIV

$H_1=\phi$ and due to the fact that r_2 is already encrypted, it will be removed in H_2 and the new value will be H_2 (Sc_2, DS→DC) = ({{r_2[DC], r_3[DC]},{r_3},{r_4, r_5, r_6, r_7}}).

For H_2, Min (|Sv|, |Ev|) =1 which is the number of sensitive record, and r_3[DC] has to be encrypted. The next table will be:

Table 21: Result After 2nd Encryptions Using H_2 [1]

TID	Name	Sex	Age	DC	DS
r_1	Joe	M	28	β	α
r_2	Alice	F	24	β	α
r_3	Maggy	F	33	β	α
r_4	Phil	M	43	CPD5	HIV
r_5	Peter	M	39	CPD5	HIV
r_6	Rey	M	52	CPD5	HIV
r_7	Steve	M	31	CPD5	HIV

As shown in Table 21, system is robust after 2 iterations and cannot be attacked. It was shown how to process to get solution with more than one security constraint let see now how to process when more than one functional dependency are given.

2.8.3.3 Multiple Functional Dependencies

Multiple functional dependencies mean existence of more than one functional dependency. As the goal is to prevent FD for attacks, each FD must be checked to verify if the security is guaranty [1] [12]. Before explaining how the robustness can be achieved, let talk about security level with encryption which is totally different from security level with multi level security. In fact, in multi level security, all attribute in the same line have the same level security [12] but in the case of encryption, only encryption can put to attributes in the same level. Therefore, when a system has to be prevented from FD attacks, as previously sensitive and evidence information must be encrypted by going through each FD dependency. The only problem now is how to manage those FD to easily get the minimal encryption during solving the problem. To solve this, Minimal Attribute cover (MAC) is used to classify FD in order to encrypt the most frequent attributes first, and the rest after. One second algorithm is then proposed for minimum attribute cover [1]. This algorithm takes as input set of functional dependencies and outputs set of minimum attributes cover. Consider:

- $\mathcal{F}$ ={F1, F2,..., Fn}as set of functional dependencies
- $\mathcal{A}$ as set of Minimal Attribute Cover
- RHS as right hand side attribute in the FD
- LHS as left hand side attribute in the FD
- R as set of attributes
- W as weight

For this algorithm, input is $\mathcal{F}$ = {F1, F2,..., Fn} and output is $\mathcal{A}$.

Algorithm 2 [1]: Find Minimum Attributes Cover ($\mathcal{F}$)

```
1. $\mathcal{A} = \emptyset$
2. For all $A \in R$ do
3.     $A.w = 0$
4. End For
5. For all $F \in \mathcal{F}'$ do
```

```
6.     For all A ∈ LHS(F) do
7.       A.w++
8.     End For
9.     RHS(F).w++
10. End For
11. While $\mathcal{F}' \neq \emptyset$ do
12.    Select A with the largest W in R
13.    For all F ∈ $\mathcal{F}'$ with A ∈ LHS(F) OR A ∈ RHS(F) do
14.      For all A' ∈ F do
15.         A'.w - -
16.      End For//A
17.      $\mathcal{F}' = \mathcal{F}'$-F
18.     $\mathcal{A}$.Add(A)
19.   End For//F
20. End While
21. Return
```

For well understanding, consider Example 9 to see how this algorithm works.

Example 9: Application of Algorithm 2 for MAC (Proposed example).

Let F_1= A, B→C; F_2=A→D; F_3=C→D, F= {F_1, F_2, F_3}, R= {A, B, C, D}, $\mathcal{A}=\emptyset$, A.weight=0; B.weight=0; C.weight=0 and D.weight=0**.** After applying loop on F, A.weight=2; B.weight=1; C.weight=2, D.weight=2.

Let A.weight=2 be the bigger weight because of the rank in the set, we are now in line 11 of our Algorithm and for the second loop, we have $\mathcal{F}'$= {F_1, F_2, F_3}.

For F_1:A.weingt=1; B.weight=0; C.weight=1 and D.weight=2, $\mathcal{F}'$= {F_2, F_3}, and $\mathcal{A}$ = {A}.

For F_2, the bigger weight is for D and A.weight=0, B.weight=0; C.weight=1; D.weight=1, $\mathcal{A}$= {A, D} and $\mathcal{F}$'= {F3}.

For F3 the bigger weight is for C and A.weight=0, B.weight=0, C.weight=0, D.weight=0, $\mathcal{F}$'= {∅} and $\mathcal{A}$= {A, D, C}.

So minimal cover is $\mathcal{A}$= {A, D, C}.

Previous sections presented what have been done to prevent and secure dataset for functional dependency attacks and method used can now be analyzed to highlight the shortcomings in order to improve the way to secure a system by partial encryption.

2.9 Problem Definition

Since the beginning of this work, key concepts were defined and analyzed in order to understand how PEM works. For more understanding and especially the concern to improve what have been done, the next work will focus first on analyzing PEM problems, secondly will focus on proposition of modification of PEM for fixing problems and get PEM-M, will thirdly focus on implementation of PEM and PEM-M so that in the fourth point both methods will be tested and forward in the last point on conduction of experiments on PEM and PEM-M similar to those conducted on PEM, and compare their efficiency.

Chapter 3

PROBLEMS OF PEM

The goal of this section is to analyze and describe PEM in order to detect existing problems. In fact, as mentioned previously, the purpose is to secure efficiently the set of data with the smallest number of encryption. Therefore, the method proposed in [1] is showing some problems that have to be solved to improve the way of securing dataset. This part of work will first consist in analyzing PEM problems; secondly consist in proposition of modification of PEM to get PEM-M for fixing attacks problems.

3.1 Problem 1: GMM can result in double encryption

As said before, GMM is an algorithm to fix FD attacks problems. There are some cases where it is not satisfied because of double encryption. Example 10 will clearly show one case when double encryption is possible.

Example 10: Illustration of a case of double encryption using algorithm GMM with multiple FDs (Proposed example)

Table 22: Original Table of D

TID	A	B	C	D	E
r_1	a_1	b_1	c_1	d_1	e_1
r_2	a_2	b_2	c_2	d_2	e_2
r_3	a_3	b_1	c_1	d_1	e_1
r_4	a_4	b_3	c_3	d_3	e_3
r_5	a_1	b_2	c_2	d_2	e_1
r_6	a_6	b_1	c_1	d_1	e_1
r_7	a_7	b_2	c_2	d_2	e_2

Table 23: Basic Encryption $\overline{D}$ of D

TID	A	B	C	D	E
r_1	a_1	$\boldsymbol{\beta_1}$	$\boldsymbol{\alpha_1}$	d_1	e_1
r_2	a_2	b_2	c_2	d_2	e_2
r_3	a_3	$\boldsymbol{\beta_1}$	c_1	d_1	e_1
r_4	a_4	b_3	c_3	d_3	e_3
r_5	a_1	$\boldsymbol{\beta_2}$	$\boldsymbol{\alpha_2}$	d_2	e_1
r_6	a_6	$\boldsymbol{\beta_1}$	c_1	d_1	e_1
r_7	a_7	b_2	c_2	d_2	e_2

FD_1: D→C, FD_2: D→B, FD_3: B→C

$\mathcal{L}$ – {SC1: $\Pi_B \sigma_{E=e_1}$; SC_2: $\Pi_C \sigma_{A=a_1}$}, Table 22 original table and Table 23 Basic encryption. In Table 23, it is noticed in r_1 (B→C) that attributes B and C are both already encrypted. Thus, to determine evidence record, β1 should be consider as LHS attribute. In that way, the next step will show one sensitive record (sv) and two evidence records (ev) and {|sv|=1<=|ev|=2 =>Minse=sv= {r_1} which means the total number of sensitive record is less than the total number of evidence records and sensitive records should be kept for encryption. B will then have to be encrypted twice in r_1 which will give two encryptions in the same attribute. In conclusion it can be say that GMM applying with MFD is not always satisfied.

3.2 Problem 2: Basic encryption scheme and ML security are two different models

Multi-Level security is defined as the application of a computer system to process information with incompatible classifications (i.e., at different security levels), permitting access for users with different security clearances, and prevent users from obtaining access to information for which they lack authorization [3].

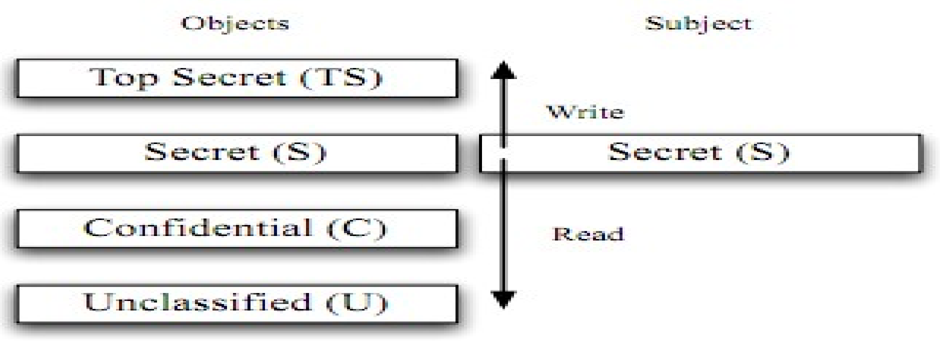


Figure 5: Example of Multilevel Security [12]

Figure 5 is showing many levels of security represented by each line. It can be noticed that four levels of security are representing in descending order. In that order, the level above can get access to the ones below. So, TS>S>C>U which means TS has the highest security level and can access to the information to the level below, and U has the lowest security level and can be accessed by the others level. In the Basic encryption scheme, such levels are not considered, it is just expected that sensitive cells are encrypted whereas non-sensitive cells are not encrypted. As said in section 2.1.3 ML security works with two rules, "No-read up" and "No-write down" with different level of security but basic encryption scheme effectively has just two security levels: encrypted and not encrypted. A holder of a secret key can access the both, a subject not having the key can access just non-encrypted data. Thus, "no-read up" is supported, but a holder of the key can write to non-encrypted data as well, and so, "no-write down" rule is not supported. Thus, Basic encryption scheme differs from ML security model.

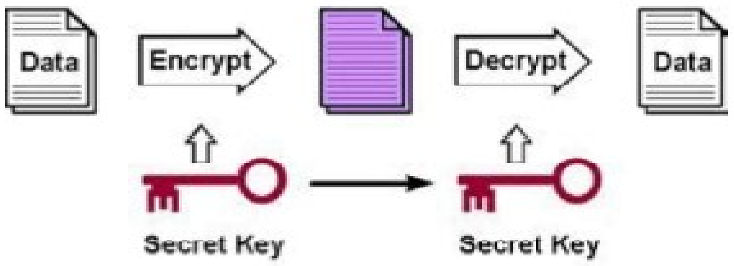


Figure 6: Illustration of Encryption Model [12]

With Figure 6, it is noticed that the only condition to get access of information is to have key then, you can be in the same level of security but if you don't have key you will not read the encrypted information.

That was prove that Multilevel Security model is not the same as Basic encryption scheme model.

3.3 Problem 3: Lemma 4.1 [1] for ML security is not true for the use of basic encryption scheme

Lemma 4.1 is used to check if a system is robust or not. Section 2.1.9.1 showed that there are two conditions in Lemma 4.1. It refers to Lemma 3.1 [12] to prove that a system is robust. However, Lemma 3.1 [12] is used to avoid FD attack on databases with ML security [16] [6]. In others words, for a set of data D with A, B and C as attributes, let SL denotes security level. If SL(A) > SL(B) > SL(C) >SL(D), it is says that FD= B, C→A compromises A because SL(A) is greater than SL of B and C and consequently they can determine A. to solve the problem, attributes should be classified so that the left-hand side (LHS) of an FD has SL not less than SL of the right-hand side of the FD. This is totally different from Lemma 4.1 which uses encryption of sensitive cells to protect them instead of SL.

Proof:

Condition 1 of Lemma 4.1 states that if some attribute $A \in X$ is encrypted then the sensitive cell, $r[Y]$, cannot be compromised with $X \rightarrow Y$. Assume that $r[X]$is encrypted. If there exists other record, $r' \neq r$, such that $r^{'}[X]$ is encrypted and $r[X] = r^{'}[X]$, but $r^{'}[Y]$ is not encrypted then original content of $r[Y]$ can be revealed as $r^{'}[Y]$ from the functional dependency $X \rightarrow Y$, this assumption on existence of r' does not contradict Condition 2 of Lemma 4.1 since Condition 2 concerns not existence of the records with $r^{'}[X]$ being a plaintext. Thus, Lemma 4.1 is not true for the case of using basic encryption scheme.

3.4 Problem 4: Algorithm_2 outputs minimum attribute cover instead of functional dependency with a new order

As explained before, Algorithm 2 is used when there is more than one functional dependency in order to reorder them so that in Algorithm 1, system will start working with functional dependencies which have the most frequent attributes. Therefore, Algorithm 2 just output a set of minimum attribute cover instead of set of set of functional dependencies with new order while in Algorithm_1 system goes through each functional dependency to fix attacks problems.

That was the problems retained by analyzing the method proposed in [1].

Chapter 4

PARTIAL ENCRYPTION METHOD MODIFIED (PEM-M)

This Chapter aims to take into account problems detected in PEM and proposes solutions for the ones which affect accuracy of fixing attacks problems. Based on problems that were detected in the previous section, some solutions are proposed in order to improve accuracy of the method. . To evaluate performance and accuracy of PEM-M, implementation of PEM and PEM-M will be done in Section 4 and some experiments will be done in Section 5 using a TEST database, PEM and PEM-M and it will be shown that it works with 0% risk of double encryption but can perform less better than PEM in the term of execution time.

Section 4.1 will show a flowchart diagram to present a general idea of PEM-M, Section 4.2 will propose encryption of the concatenation of TID value and concerning cell to encrypt for fixing double encryption problem and section 4.3 will propose a modification of Algoritm_1 for MAC so that output will be a set of reordered functional dependencies.

4.1 Flowchart of PEM-M

Let precise that PEM-M flowchart is almost similar to PEM flowchart due to the fact that PEM was adjusted to get PEM-M. So, the part described in Section 2 will not be described in this section.

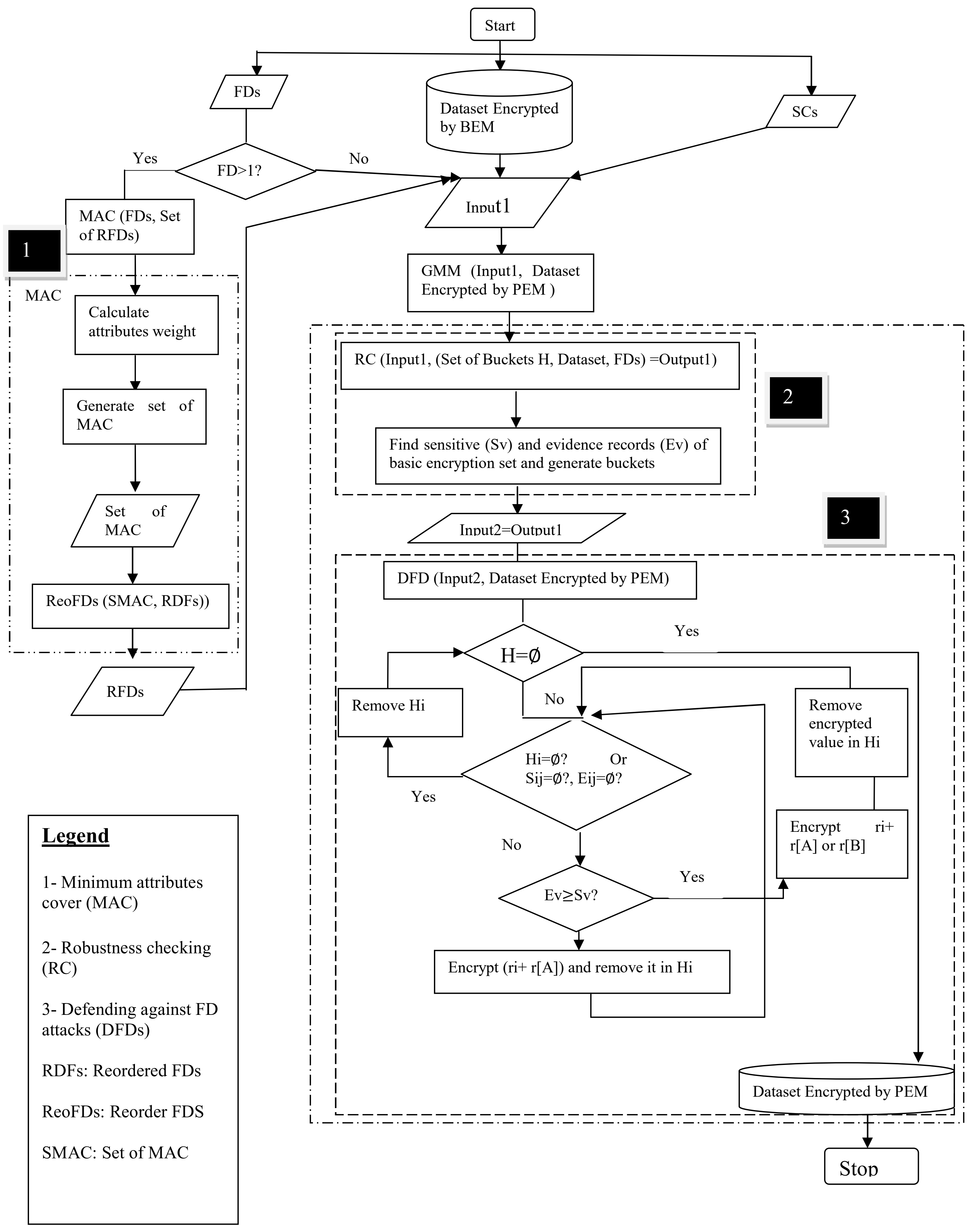

Start
FDs
Dataset Encrypted by BEM
SCs
FD>1?
Yes
No
MAC (FDs, Set of RFDs)
Input1
1
MAC
Calculate attributes weight
Generate set of MAC
Set of MAC
ReoFDs (SMAC, RDFs))
RFDs
GMM (Input1, Dataset Encrypted by PEM)
RC (Input1, (Set of Buckets H, Dataset, FDs) =Output1)
2
Find sensitive (Sv) and evidence records (Ev) of basic encryption set and generate buckets
Input2=Output1
3
DFD (Input2, Dataset Encrypted by PEM)
H=∅
Yes
No
Remove Hi
Remove encrypted value in Hi
Hi=∅? Or Sij=∅?, Eij=∅?
Yes
No
Encrypt ri+ r[A] or r[B]
Ev≥Sv?
Yes
Encrypt (ri+ r[A]) and remove it in Hi
Dataset Encrypted by PEM
Stop
Legend
1- Minimum attributes cover (MAC)
2- Robustness checking (RC)
3- Defending against FD attacks (DFDs)
RDFs: Reordered FDs
ReoFDs: Reorder FDS
SMAC: Set of MAC

Figure 7: Flowchart of PEM-M

4.2 Modification 1: Contribution to solve problem of double encryption

As explained in Problem 1 Section 3.1, there is a risk of double encryption in the same attribute when existing more than one functional dependency. To solve this problem and get 0% of risk of double encryption, a concatenation of TID value and concerning cell before encryption is proposed so that for the same value, cipher text can be different. It can be noticed in part 2 in the flowchart of PEM-M where row number ri is concatenated with concerning value before encryption. If consider again Example 10 Section 3.1, to solve the risk of double encryption before encrypting each r[B], the corresponding value is concatenated with the value corresponding to the row number. That will allow each value to have a unique encrypted value and prevent a system for double encryption in the same value.

Demonstration:

Table 24: Basic Encryption of D

TID	A	B	C	D	E
$\mathbf{r_1}$	a_1	$\mathbf{b_1}$	$\mathbf{c_1}$	d_1	e_1
r_2	a_2	b_2	c_2	d_2	e_2
$\mathbf{r_3}$	a_3	$\mathbf{b_1}$	c_1	d_1	e_1
r_4	a_4	b_3	c_3	d_3	e_3
$\mathbf{r_5}$	a_1	$\mathbf{b_2}$	$\mathbf{c_2}$	d_2	e_1
$\mathbf{r_6}$	a_6	$\mathbf{b_1}$	c_1	d_1	e_1
r_7	a_7	b_2	c_2	d_2	e_2

Table 25: Original Table $\overline{D}$ of D

TID	A	B	C	D	E
r_1	a_1	$\boldsymbol{\beta_1}$	$\boldsymbol{\alpha_1}$	d_1	e_1
r_2	a_2	b_2	c_2	d_2	e_2
r_3	a_3	**¥1**	c_1	d_1	e_1
r_4	a_4	b_3	c_3	d_3	e_3
r_5	a_1	$\boldsymbol{\beta_2}$	$\boldsymbol{\alpha_2}$	d_2	e_1
r_6	a_6	$\mathbf{\ddot{x}_1}$	c_1	d_1	e_1
r_7	a_7	b_2	c_2	d_2	e_2

FD_1: D→C, FD_2: D→B, FD_3: B→C and $\mathcal{L} - \{SC_1: \Pi_B \sigma_{E=e_1}; SC_2: \Pi_C \sigma_{A=a_1}\}$ are given. If r_1 and b_1 are concatenated before encryption, r_3 and b_1 are also concatenated before encryption, it means for the same value of b a unique cipher text is obtained.

As shown in Table 25, with different encrypted value α1 cannot be inferred using r_3 and r_6 and following algorithm_1 they cannot be encrypted twice as shown before in Example 10.

4.3 Modification 2: Contribution to solve problem of Algorithm_2

Algorithm 2 takes set of functional dependencies like input and output set of minimum attribute cover which is not used in Algorithm 1. In fact, Algorithm_2 should output set of functional dependencies reordered. In Part 3 of flowchart of PEM-M, it is noticeable that comparing to PEM, output is a set of reordered FDs. So this part will be added in Algorithm 2 so that output will be a set of functional dependencies reordered.

Algorithm 2-Modified: Finding reordered set of FDs using MAC ($\mathcal{F}$)

```
1.  $\mathcal{A} = \emptyset$
2.  For all A ∈ R do
3.     A.w = 0
4.  End For
5.  For all F ∈ $\mathcal{F}'$ do
6.     For all A ∈ LHS(F) do
7.        A.w++
8.     End For
9.     RHS(F).w++
10. End For
11. While $\mathcal{F}' \neq \emptyset$ do
12.    Select A with the largest W in R
13.    For all F ∈ $\mathcal{F}'$ with A ∈ LHS(F) OR A ∈ RHS(F) do
14.       For all A' ∈ F do
15.           A'.w - -
16.       End For//A
17.       $\mathcal{F}' = \mathcal{F}'$-F
18.       $\mathcal{A}$.Add(A)
19.    End For//F
20. End While
21. For all A ∈ $\mathcal{A}$.
```

```
22.   For all F $\in \mathcal{F}$
23.     If (A $\in$ F) And (F $\notin \mathcal{F}'$)
24.       $\mathcal{F}'$.ADD (F)
25.     End If
26.   End For
27. End For
28. Return ($\mathcal{F}'$)
```

If Example 9 in Section 2.1.8.3.3 is considered again, minimal cover is $\mathcal{A}$= {A, D, C} and FD order is F= {F_1, F_2, F_3}. Contrary to Algorithm_2, Algorithm_2-Modified outputs reordered set of FD due to modifications done from line 21 to line 28 in Algorithm 2.

Chapter 5

IMPLEMENTATION ENVIRONMENT

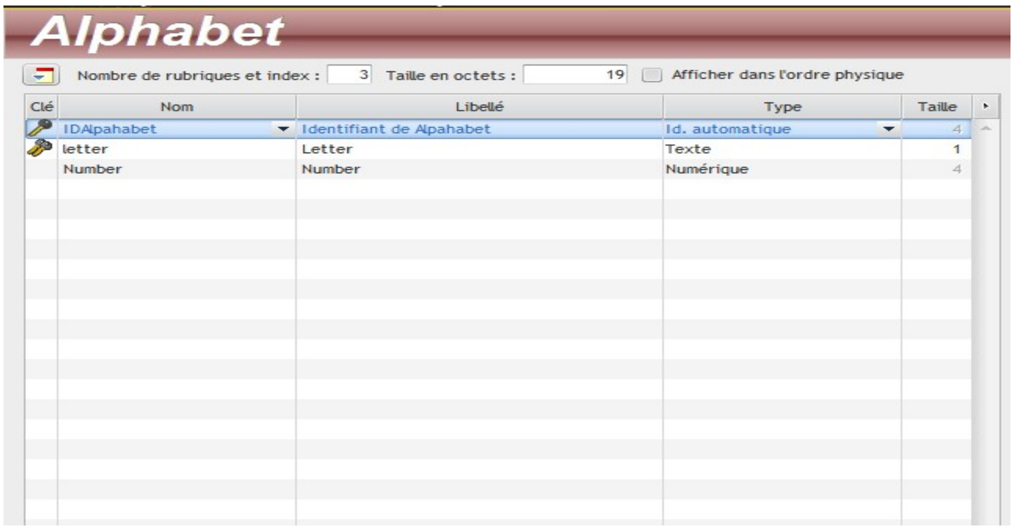
This section will described materials used for implementation.

5.1 WIndev17 Description

Windev can be defined as software of engineering workshop developed in France in 1993 by PC soft Company [18]. In addition to languages like Java, SQl, Visual Basic and others, this platform has his own language called WD language [19]. It is also possible to import a database from others sources or create directly a database in the platform. Started from version 1 to version 25 nowadays, the software is able to work on windows and Linux system. This platform mainly allows development of data oriented software, which can also work in windows and Linux. This software also offers a possibility to develop web application throughout Webdev and Mobil application throughout Windev Mobil. For this implementation, a database which helps to apply Vigenere encryption method is created. Before showing the implementation of the database, let see briefly how Vigenere encryption method works.

5.2 Database in Windev

For implantation, a database was created with one table which is going to contain alphabet characters and their representative numerical value. Two attributes Letter and Number were created, with Letter as primary key. Letter is going to store an alphabet character and Number is going to store a numerical corresponding value. Alphabet is going from A to 9 and will use modulo 36 for Vigenere encryption described in section 2.2.2. Database is called Pencrypt and contains table Alphabet (**Number**, Letter). Let see how the database looks in Windev:



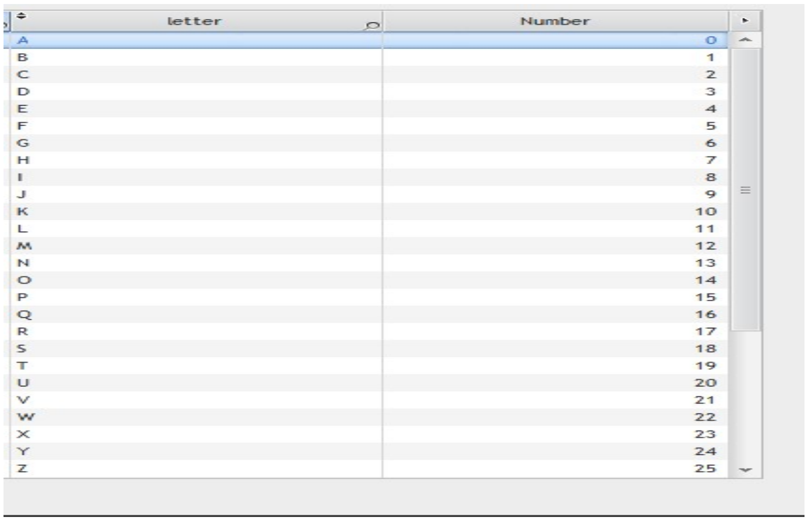
Alphabet

Nombre de rubriques et index : 3 Taille en octets : 19 Afficher dans l'ordre physique

Clé	Nom	Libellé	Type	Taille
🔑	IDAlpahabet	Identifiant de Alpahabet	Id. automatique	4
🔑	letter	Letter	Texte	1
	Number	Number	Numérique	4

Figure 8: Figure of table Alphabet

Since database and table are created, different occurrences of data like in Figure 7 can be seen.



letter	Number
A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8
J	9
K	10
L	11
M	12
N	13
O	14
P	15
Q	16
R	17
S	18
T	19
U	20
V	21
W	22
X	23
Y	24
Z	25

Figure 9: Figure Showing Values Inside Table Alphabet

Those values were introduced directly in the database without any writing code.

Chapter 6

IMPLEMENTATION OF PEM AND PEM-M

This chapter describes implementation environment and gives details about PEM and PEM-M implementation based on scheme described in Chapters 2 and 3. Problems detected in PEM will also be considered in implementation of PEM-M in order to improve performance of functional dependency resisting attacks. Section 5.1 will start giving flowchart of PEM and PEM-M as a whole, and the remaining sections of the chapter will present PEM and PEM-M implementation figures.

First of all, let’s mention that PEM and PEM-M have the same flowchart but some different part of algorithm due to modifications.

6.1 Flowchart of PEM and PEM-M

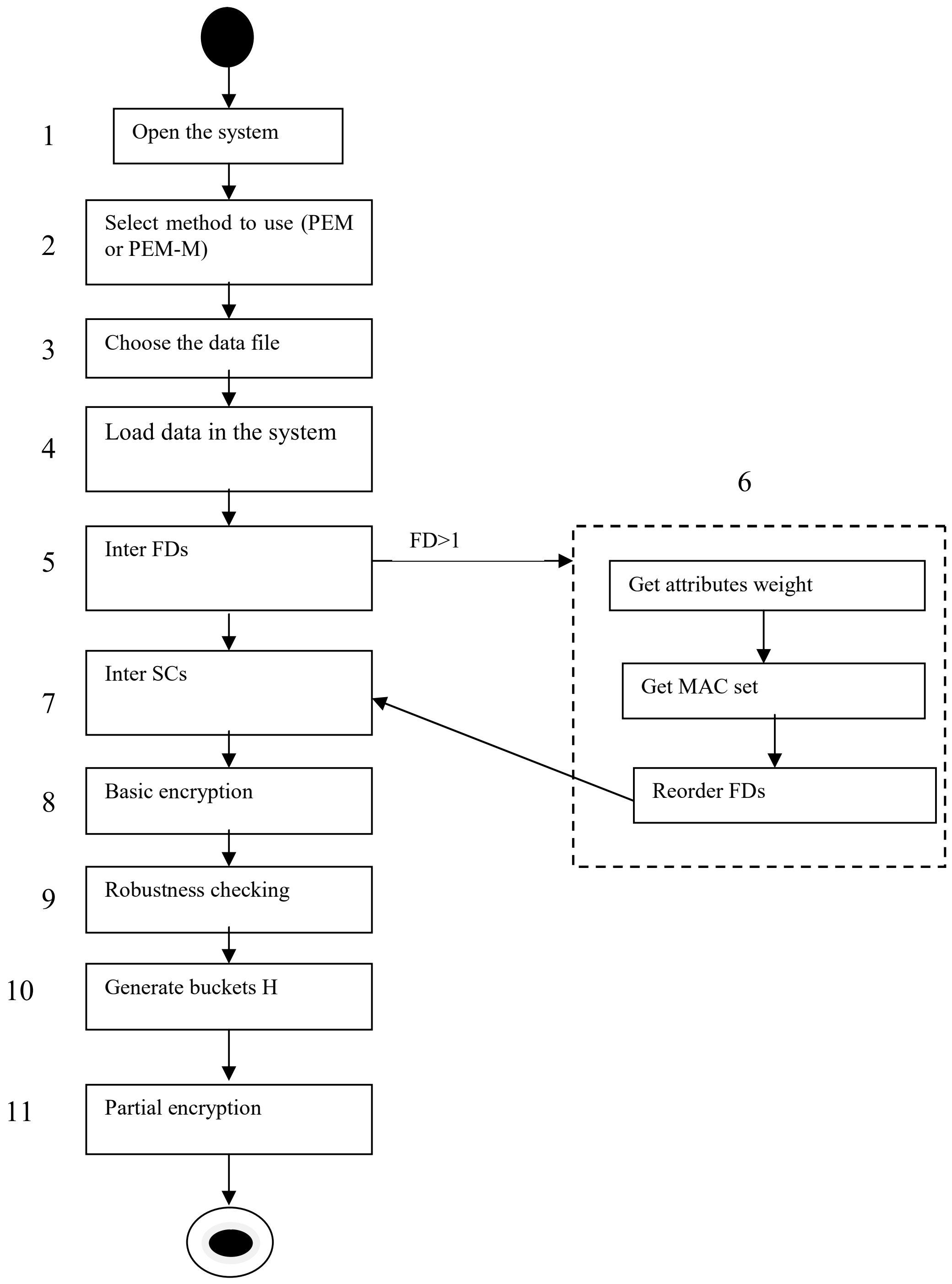


Figure 10: Flowchart of PEM and PEM-M

Figure 10 shows all steps that should be followed for fixing FDA using PEM or PEM-M. Both methods are almost similar. The difference between them was clearly described in the previous section and it was said that PEM-M will concatenate row value with main value before encryption unlike of PEM which encrypts only the main value. For others steps, process is same.

When user will open proposed software, he will have to choose the use of PEM or PEM-M like précised in 2 in Figure 10. After selecting what to use, the process of loading dataset, entering FDs and SCs, MAC and sensitive cells encryption are the same in both methods. Those steps represented from 2 to 9 were described in Section 2 including robustness checking described in Flowchart of PEM in the same section.

6.2 Open the system

The software has an executable file to run it. The file is called Pencrypt.exe and is in the software package.

6.3 Selection of Method PEM or PEM-M

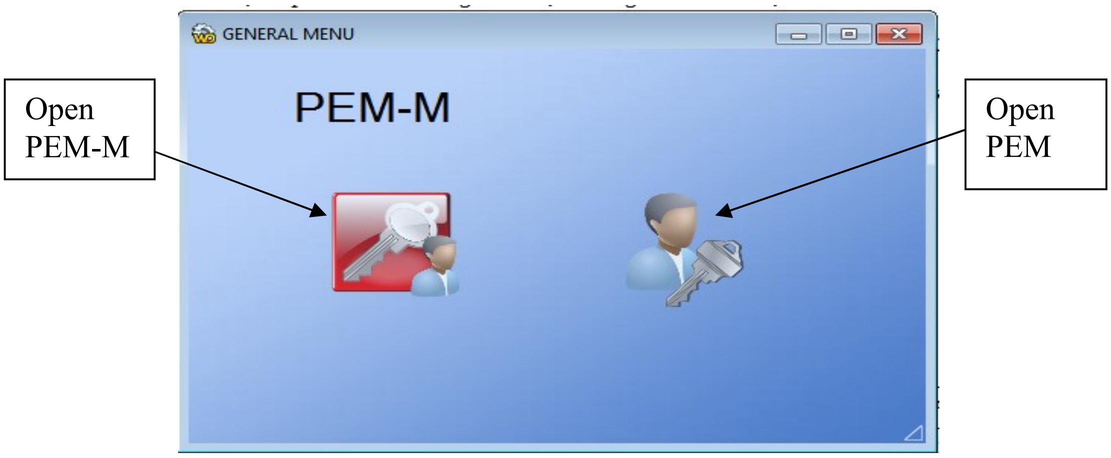
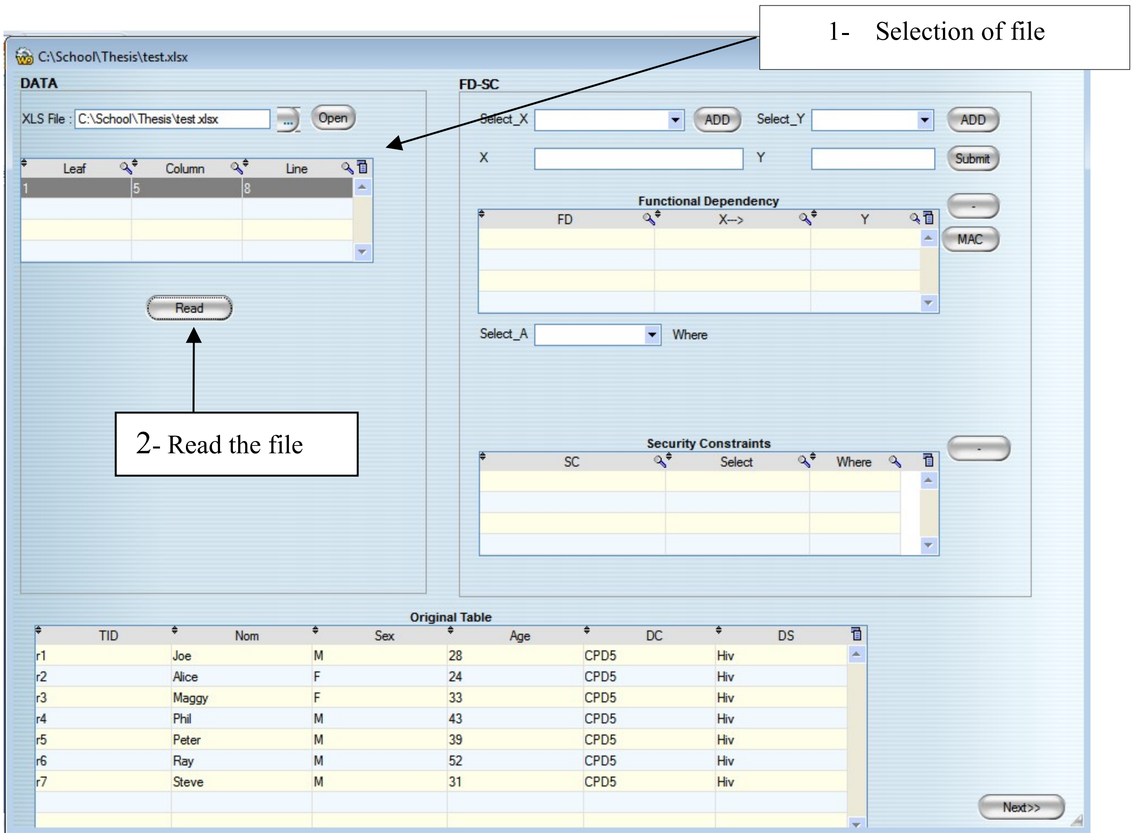


Figure 11: Selection of Method PEM or PEM-M

6.4 Dataset Loading of PEM or PEM



XLS File : C:\School\Thesis\test.xlsx

Leaf	Column	Line
1	5	8

Original Table

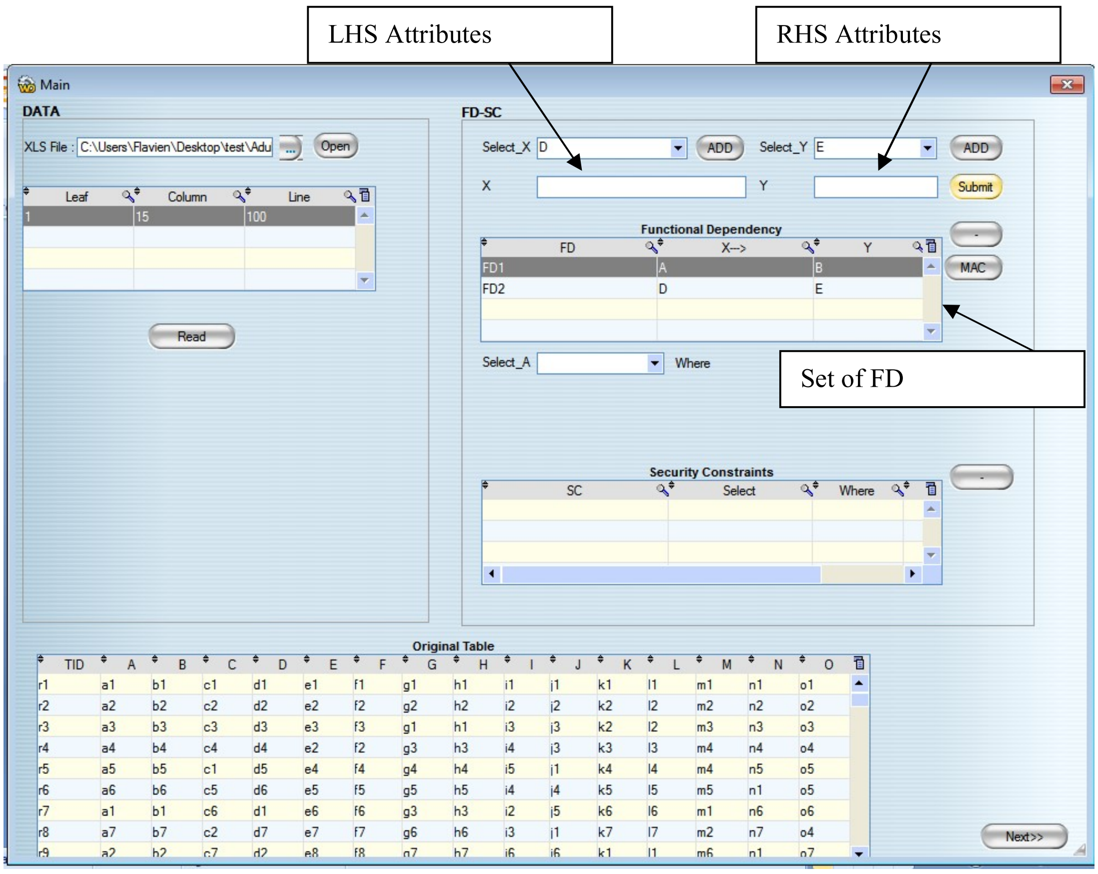
TID	Nom	Sex	Age	DC	DS
r1	Joe	M	28	CPD5	Hiv
r2	Alice	F	24	CPD5	Hiv
r3	Maggy	F	33	CPD5	Hiv
r4	Phil	M	43	CPD5	Hiv
r5	Peter	M	39	CPD5	Hiv
r6	Ray	M	52	CPD5	Hiv
r7	Steve	M	31	CPD5	Hiv

Figure 12: Loading of Dataset of PEM or PEM-M

Figure 12, shows that application allows user to select file (1). After selection he can click on OPEN to see in the small table below, number of leaf, number of line, and number of columns in the dataset. Since it is done, he clicks on READ (2) to load the data in the big table below as shown in Figure 12. For this figure related to Example 8, 1 leaf, 5 columns and 8 lines are shown. When data are loaded, user can now go to the next step. Those are the results obtained using codes described in Appendix A1 for opening the selected file from line 1 to line 17, and Appendix A2 for reading and display file content from line 18 to line.

6.5 Enter Functional Dependency

Since data are loaded, select attributes which are going to represent functional dependency and submit them to the set of functional dependency as in the figure bellow.



Leaf	Column	Line
1	15	100

Functional Dependency

FD	X-->	Y
FD1	A	B
FD2	D	E

Security Constraints

SC	Select	Where

Original Table

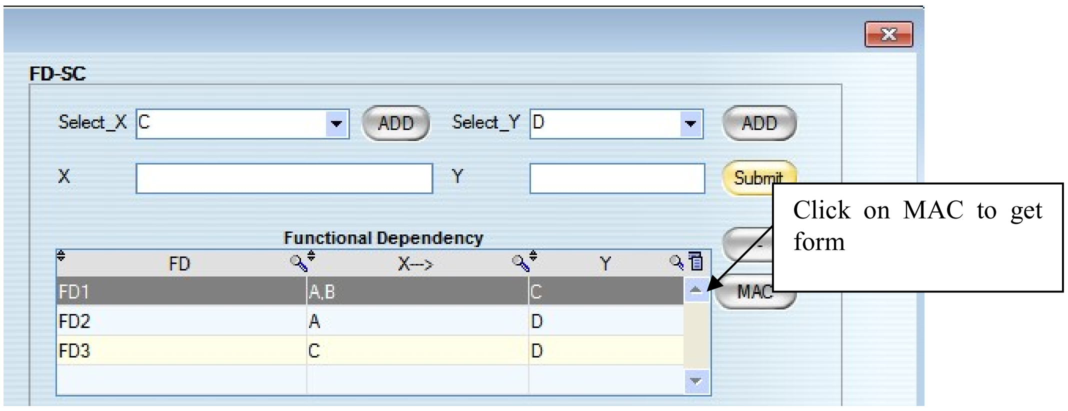
TID	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
r1	a1	b1	c1	d1	e1	f1	g1	h1	i1	j1	k1	l1	m1	n1	o1
r2	a2	b2	c2	d2	e2	f2	g2	h2	i2	j2	k2	l2	m2	n2	o2
r3	a3	b3	c3	d3	e3	f3	g1	h1	i3	j3	k2	l2	m3	n3	o3
r4	a4	b4	c4	d4	e2	f2	g3	h3	i4	j3	k3	l3	m4	n4	o4
r5	a5	b5	c1	d5	e4	f4	g4	h4	i5	j1	k4	l4	m4	n5	o5
r6	a6	b6	c5	d6	e5	f5	g5	h5	i4	j4	k5	l5	m5	n1	o5
r7	a1	b1	c6	d1	e6	f6	g3	h3	i2	j5	k6	l6	m1	n6	o6
r8	a7	b7	c2	d7	e7	f7	g6	h6	i3	j1	k7	l7	m2	n7	o4
r9	a2	b2	c7	d2	e8	f8	g7	h7	i6	j6	k1	l1	m6	n1	o7

Figure 13: Enter FDs

Figure 13 shows two FDs, FD_1=A→B and FD_2=D→E. representing in the table which is considered as set of functional dependencies. The code used to implement this is described in Appendix A.4 where FDs are inserted into the table from line 84 to line 89.

6.6 Get MAC set of PEM or PEM-M

For implementation of MAC, set of FDs was keep for both methods since it was not possible to get result using set on minimum attributes. When there is more than one functional dependency, MAC can be applied in order to use the best FDs order to make implementation fast and efficient [1]. This function was implemented in the basis of Algorithm_1 [1], and using Example 9 to see how it works in the application.



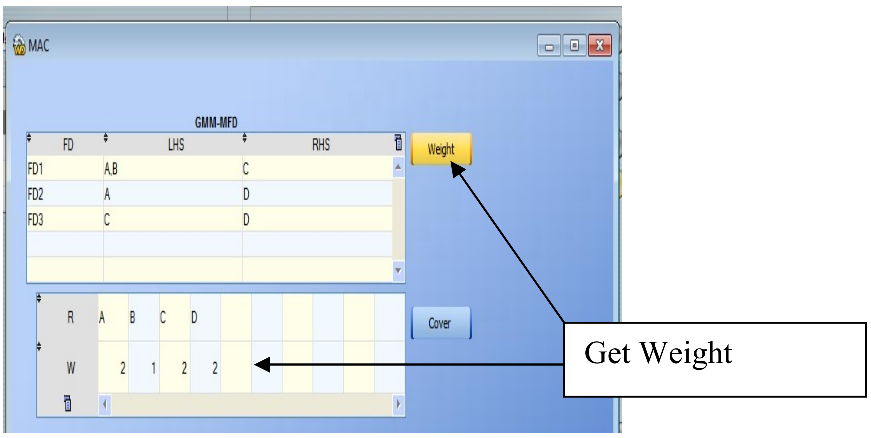
Select_X: C
Select_Y: D

Functional Dependency

FD	X-->	Y
FD1	A,B	C
FD2	A	D
FD3	C	D

Figure 14: Implementation of MAC for PEM or PEM-M (part 1)

In Figure 14, FD_1, FD_2 and FD_3 are shown exactly like in Example 9. After getting set of FDs, if number of FDs is greater than one you can click on MAC to apply Algorithm 1. Let see the result in Figure 15.



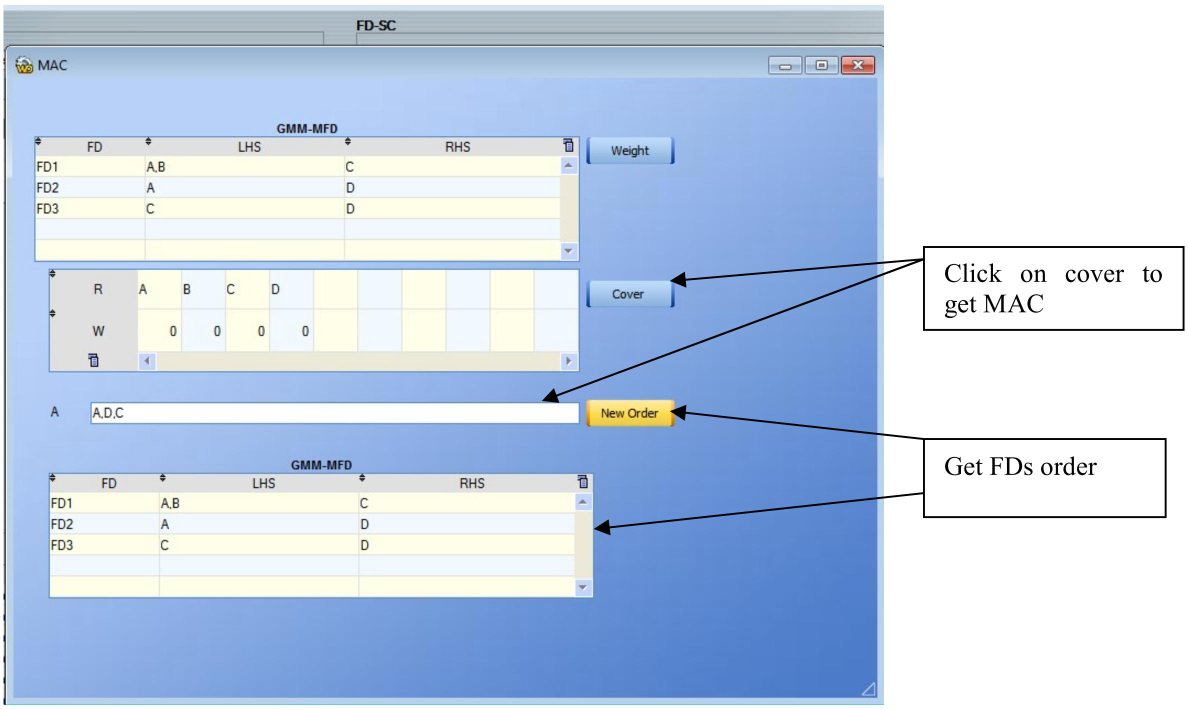
GMM-MFD

FD	LHS	RHS
FD1	A,B	C
FD2	A	D
FD3	C	D

R	A	B	C	D
W	2	1	2	2

Figure 15: Get Attributes Weight for PEM or PEM-M

The code using to get this example of result is described in Appendix A.5.1 where from line 90 to 142, system goes through each functional dependency and generates weight of each attribute.



GMM-MFD

FD	LHS	RHS
FD1	A,B	C
FD2	A	D
FD3	C	D

R	A	B	C	D
W	0	0	0	0

A A,D,C

GMM-MFD

FD	LHS	RHS
FD1	A,B	C
FD2	A	D
FD3	C	D

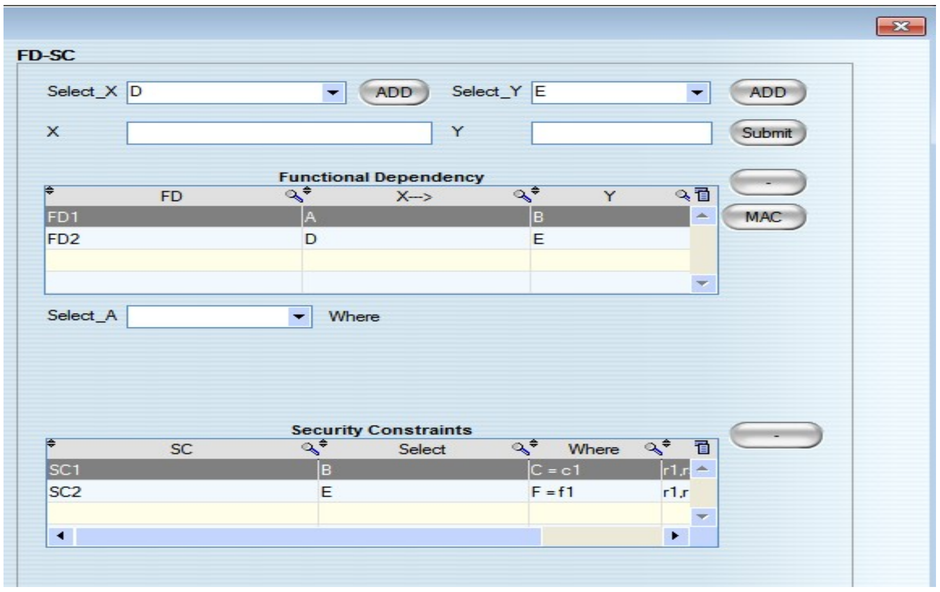
Figure 16: Get MAC and Reordered FDs for PEM or PEM-M (part 2)

In Figure 16, each attribute is weighted by clicking on button Weight. After getting weight, just click on Cover to get MAC as shown in Figure 16. As in example 9, $\mathcal{A}$= {A, D, C} and F= {FD_1, FD_2, FD_3} which means function is working perfectly.

To get this result, the code used is described in Appendix A.5.2 where, from line 143 to line 170, using previous result set of minimum attribute A is obtained. To get now the new order of FDs, the code used is Appendix A.5.3 and from line 171 to 214, loop FOR is used to go through set of attributes to generate the new order.

6.7 Enter Security Constraints

Security constraint is the last element to enter before getting Basic encryption. As FDs were entered, same process is used for security constraints but with different box. Let see the result in Figure 17.



Select_X: D
Select_Y: E

Functional Dependency

FD	X-->	Y
FD1	A	B
FD2	D	E

Security Constraints

SC	Select	Where	
SC1	B	C = c1	r1,r
SC2	E	F = f1	r1,r

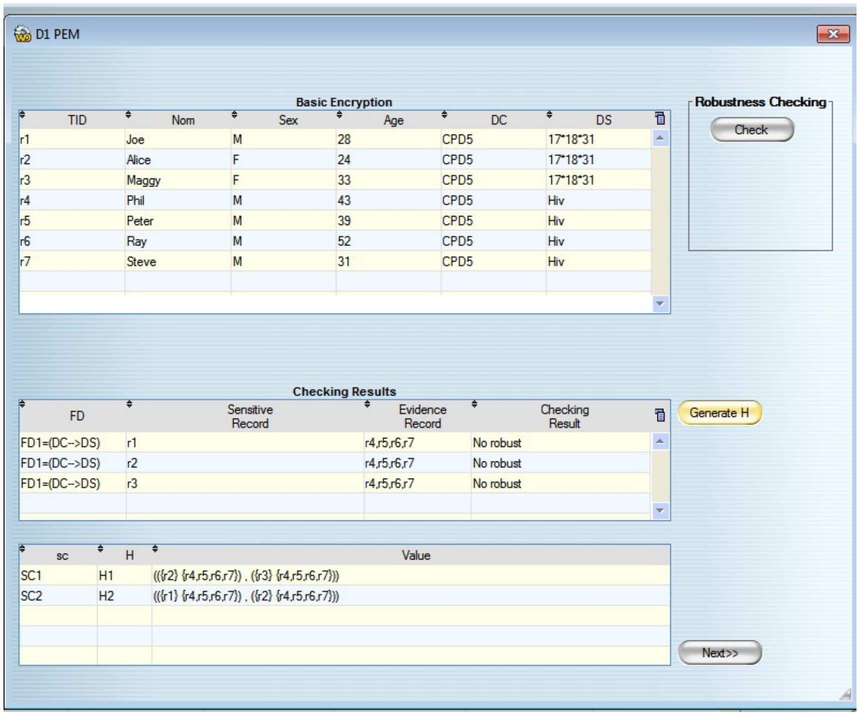
Figure 17: Enter SCs for PEM or PEM-M

Figure 17, shows in the table of security constraints two security constraints. This result was obtained applying some codes used in Appendix A.6.2, especially lines 256, 282, 306 and 331 where for each case SCs are inserted in the table. The next step is to get basic encryption, check if the system is robust, and generate Buckets H in the case that the system is not robust.

6.8 Basic Encryption Scheme, Robustness Checking and Generation of Buckets (H)

6.8.1 Basic encryption for PEM

As said before, for PEM only main value is concerns by encryption. Implementation result is like follow.



Basic Encryption

TID	Nom	Sex	Age	DC	DS
r1	Joe	M	28	CPD5	17*18*31
r2	Alice	F	24	CPD5	17*18*31
r3	Maggy	F	33	CPD5	17*18*31
r4	Phil	M	43	CPD5	Hiv
r5	Peter	M	39	CPD5	Hiv
r6	Ray	M	52	CPD5	Hiv
r7	Steve	M	31	CPD5	Hiv

Checking Results

FD	Sensitive Record	Evidence Record	Checking Result
FD1=(DC-->DS)	r1	r4,r5,r6,r7	No robust
FD1=(DC-->DS)	r2	r4,r5,r6,r7	No robust
FD1=(DC-->DS)	r3	r4,r5,r6,r7	No robust

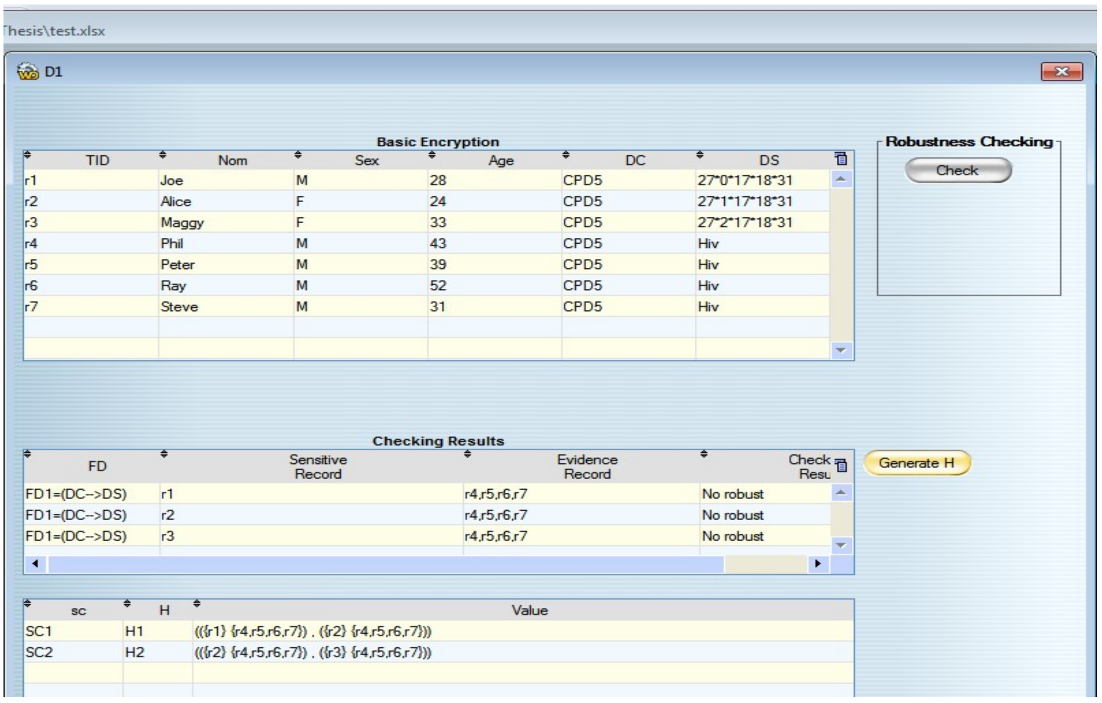
sc	H	Value
SC1	H1	(({r2} {r4,r5,r6,r7}) . ({r3} {r4,r5,r6,r7}))
SC2	H2	(({r1} {r4,r5,r6,r7}) . ({r2} {r4,r5,r6,r7}))

Figure 18: Basic Encryption of PEM

It is noticeable in the first table of Figure 18 that sensitive cells are encrypted using PEM method. The code used to get that result is in Appendix A.6.2 from line 352 to line 468. And in line 252 it is show how only the main cell is encrypted.

6.8.2 Basic Encryption of PEM-M

The process is the same but as said before PEM-M used concatenation of row and main value during encryption.



Basic Encryption

TID	Nom	Sex	Age	DC	DS
r1	Joe	M	28	CPD5	27*0*17*18*31
r2	Alice	F	24	CPD5	27*1*17*18*31
r3	Maggy	F	33	CPD5	27*2*17*18*31
r4	Phil	M	43	CPD5	Hiv
r5	Peter	M	39	CPD5	Hiv
r6	Ray	M	52	CPD5	Hiv
r7	Steve	M	31	CPD5	Hiv

Checking Results

FD	Sensitive Record	Evidence Record	Check Resu
FD1=(DC-->DS)	r1	r4,r5,r6,r7	No robust
FD1=(DC-->DS)	r2	r4,r5,r6,r7	No robust
FD1=(DC-->DS)	r3	r4,r5,r6,r7	No robust

sc	H	Value
SC1	H1	(({r1} {r4,r5,r6,r7}) . ({r2} {r4,r5,r6,r7}))
SC2	H2	(({r2} {r4,r5,r6,r7}) . ({r3} {r4,r5,r6,r7}))

Figure 19: Basic Encryption of PEM-M Robustness Checking and Buckets (H)

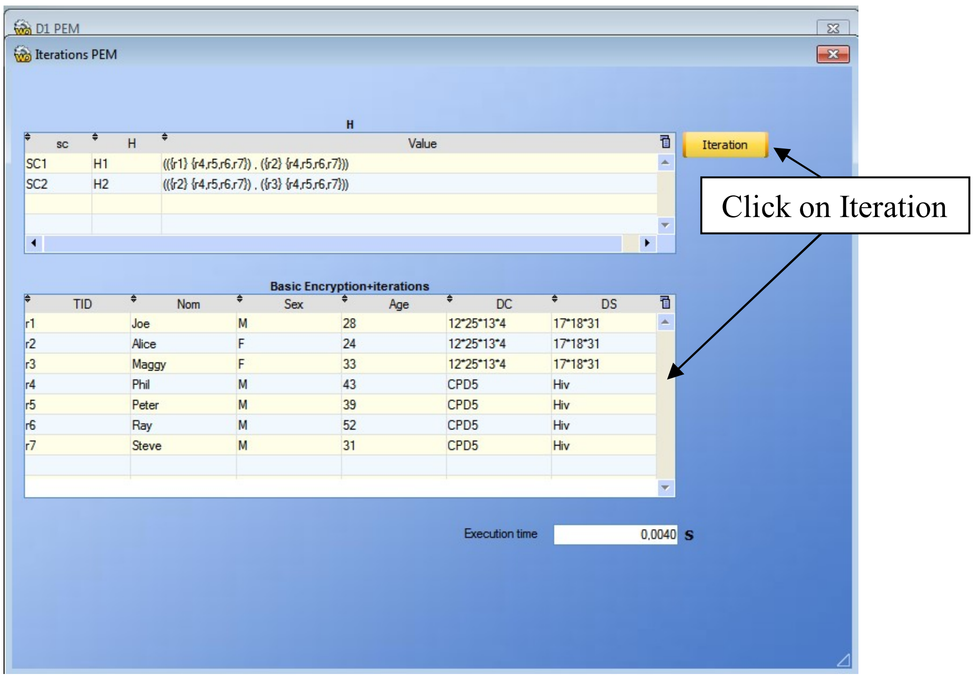
Figure 19 shows basic encryption of PEM-M, and it is noticeable that the values are different from the ones getting in PEM.

6.9 Robustness Checking and Generation of Buckets for PEM or PEM-M

For this implementation, Figure 18 or Figure 19 can be used since the process is similar in both methods. Consider SCs in the first table of Figure 19, Robustness checking in the second one, and set of Bucket H in the third table. Those results were obtained after applying codes used in Appendix A.6.2 or A.6.3 for PEM and PEM-M respectively and Basic encryption, Appendix B.1 from line 469 to line 505 for robustness checking and Appendix B.2 from line 505 to line 573. After getting all those results, step about fixing attacks problems can be implemented.

6.10 Partial Encryption

6.10.1 Partial Encryption for PEM



H

sc	H	Value
SC1	H1	(({r1} {r4,r5,r6,r7}) . ({r2} {r4,r5,r6,r7}))
SC2	H2	(({r2} {r4,r5,r6,r7}) . ({r3} {r4,r5,r6,r7}))

Basic Encryption+iterations

TID	Nom	Sex	Age	DC	DS
r1	Joe	M	28	12*25*13*4	17*18*31
r2	Alice	F	24	12*25*13*4	17*18*31
r3	Maggy	F	33	12*25*13*4	17*18*31
r4	Phil	M	43	CPD5	Hiv
r5	Peter	M	39	CPD5	Hiv
r6	Ray	M	52	CPD5	Hiv
r7	Steve	M	31	CPD5	Hiv

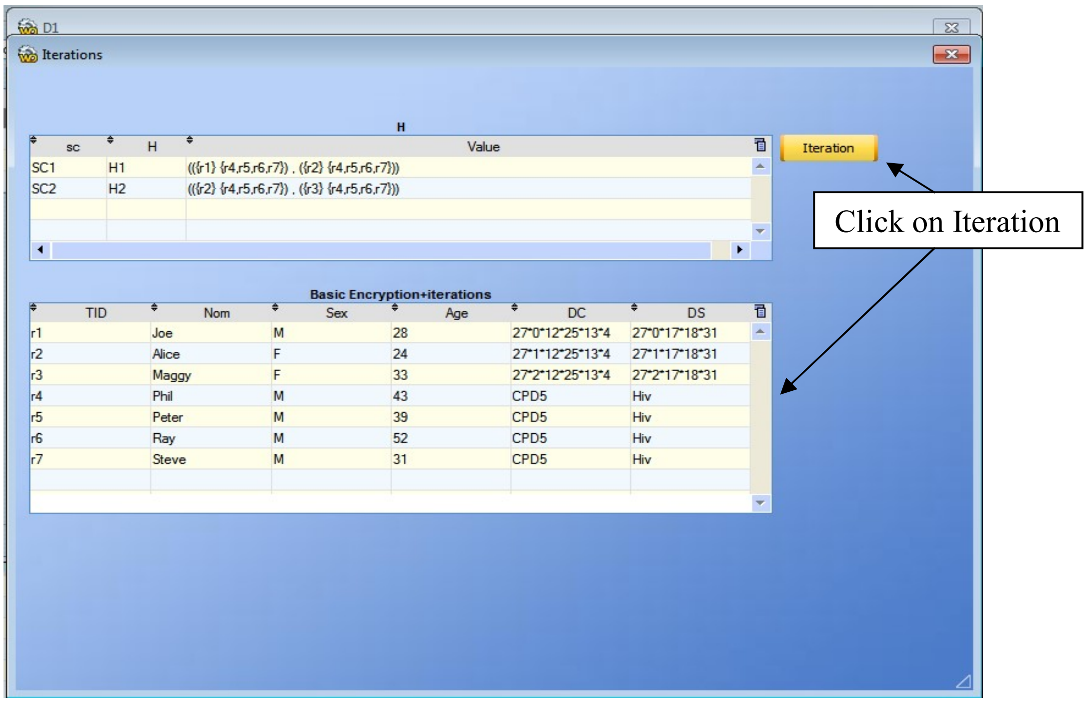
Execution time 0,0040 s

Figure 20: Partial Encryption for PEM

In Appendix C.1 partial encryption for PEM is applying from line 574 to line 633 and it is noticeable that only main value is encrypted.

6.10.2 Partial Encryption for PEM-M

The process is almost the same with the one used in PEM, only encryption makes difference. Figure 21 shows result with PEM-M and different encryption values can be noticed.



H

sc	H	Value
SC1	H1	(({r1} {r4,r5,r6,r7}) . ({r2} {r4,r5,r6,r7}))
SC2	H2	(({r2} {r4,r5,r6,r7}) . ({r3} {r4,r5,r6,r7}))

Basic Encryption+iterations

TID	Nom	Sex	Age	DC	DS
r1	Joe	M	28	27*0*12*25*13*4	27*0*17*18*31
r2	Alice	F	24	27*1*12*25*13*4	27*1*17*18*31
r3	Maggy	F	33	27*2*12*25*13*4	27*2*17*18*31
r4	Phil	M	43	CPD5	Hiv
r5	Peter	M	39	CPD5	Hiv
r6	Ray	M	52	CPD5	Hiv
r7	Steve	M	31	CPD5	Hiv

Figure 21: Partial Encryption for PEM-M

As shown in Figure 21, in this step, after clicking on Iteration and obtaining a robust system exactly as it was obtained in Example 10. This was made possible using code described in Appendix C from line 634 to line 693 where, it is shows how program goes through each record, check existence of evidence record, and solve the problem. Let mention that to prevent the system to encrypt the same attribute twice as mentioned in the previous chapter, concatenation value of TID and concerning value attribute is done in order to get a unique result of each encryption.

Chapter 7

EXPERIMENTS ON PEM AND PEM-M

In this chapter, some experiments are made in order to evaluate the performance and compare the results with the ones getting in [1]. Let describe experiment environment first.

7.1 Experimental Environment used

Experiment will be done using result obtained with Adult database in [1] and comparison will be done between PEM and PEM-M using TEST database which will be described in the following sections.

7.1.1 Adult Database

As introduced in Chapter 2, Adult database was used in the experiment done in [1]. Based on the difficulty to get 78 FDs as they said during their experiment, a TEST database will be created to perform experiment with PEM and PEM-M. Adult database has 15 attributes and can be downloaded in [5]. TEST database will have exactly 15 attributes in order to get something similar to Adult database.

7.1.2 TEST database

As said in the previous section, TEST database was created to perform experiments in this work.

7.1.2.1 Description of TEST Dataset

TEST is a dataset with 15 alphabet letters considered as attributes. From A to O 5 FDs were considered FD_1: $A \rightarrow B$, FD_2: $E \rightarrow F$, FD_3: $B \rightarrow D$, FD_4: $G \rightarrow H$, FD_5: $A \rightarrow B$

and FD_5: K→L. TEST was created in Excel 2007 and will be used with 100 rows up to 32K rows.

7.1.2.2 Structure of TEST Dataset

Since caption is not clear to present TEST structure, following table will be used to show how TEST dataset looks like.

Table 26: Structure of TEST Dataset

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
a_1	b_1	c_1	d_1	e_1	f_1	g_1	h_1	i_1	j_1	k_1	l_1	m_1	n_1	o_1
a_2	b_2	c_2	d_2	e_2	f_2	g_2	h_2	i_2	j_2	k_2	l_2	m_2	n_2	o_2
a_3	b_3	c_3	d_3	e_3	f_3	g_1	h_1	i_3	j_3	k_2	l_2	m_3	n_3	o_3
a_4	b_4	c_4	d_4	e_2	f_2	g_3	h_3	i_4	j_3	k_3	l_3	m_4	n_4	o_4
a_5	b_5	c_1	d_5	e_4	f_4	g_4	h_4	i_5	j_1	k_4	l_4	m_4	n_5	o_5
a_6	b_6	c_5	d_6	e_5	f_5	f_5	h_5	i_4	j_4	k_5	l_5	m_5	n_1	o_5
a_1	b_1	c_6	d_1	e_6	f_6	g_3	h_3	i_2	j_5	k_6	l_6	m_1	n_6	o_6
a_7	b_7	c_2	d_7	e_7	f_7	g_6	h_6	i_3	j_1	k_7	l_7	m_2	n_7	o_4
a_2	b_2	c_7	d_2	e_8	f_8	g_7	h_7	i_6	j_6	k_1	l_1	m_6	n_1	o_7

7.2 Materials

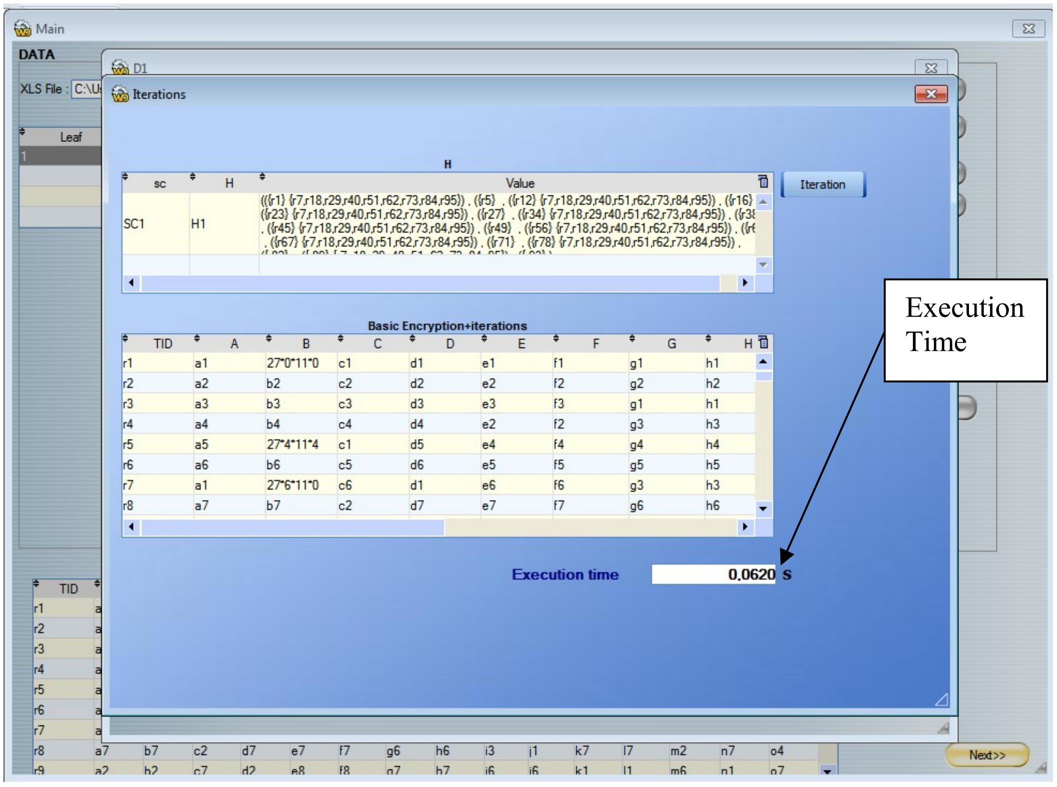
Those experiments will be done in a Windows 7 system 64 bits, with processor Intel (R) Core (TM) i5 CPU 2.6GHz and 4GB RAM.

7.3 Experiments Description

For the experiments, database TEST will be used with 100, 1000, 5000, 10000, 20000, 25000, and 32000 records. Execution time will be estimated using PEM-M combinations numbers of FDs and SCs from 1 to 4. After getting execution time in each case with PEM-M, the same experiment will be perform using PEM with 100 and 32K records applying with 4 FDs (FD_1=A→B, FD_2: F→E, FD_3: G→H and FD_4: K→L) and 4SCs (SC_1: $\Pi_B \sigma_{C=c1}$, SC_2: $\Pi_E \sigma_{I=i1}$, SC_3: $\Pi_H \sigma_{J=j1}$, SC$_4$: $\Pi_L \sigma_{O=o1}$). A performance comparison will be done and obtained results will be commented.

7.4 Experimental Results using PEM-M

This experiment will summarize in different tables all what was described before and analyze time execution based on each parameter using PEM-M. The following figures are showing results obtained during experiments and different execution times depending on different parameters. Result is shown in Table ITERATION and for the first case with FD= A→B and SC= $\Pi_B \sigma_{C=c1}$, Figure 23 will show example of result.



H

sc	H	Value
SC1	H1	(({r1} {r7,r18,r29,r40,r51,r62,r73,r84,r95}) , ({r5} , ({r12} {r7,r18,r29,r40,r51,r62,r73,r84,r95}) , ({r16} ({r23} {r7,r18,r29,r40,r51,r62,r73,r84,r95}) , ({r27} , ({r34} {r7,r18,r29,r40,r51,r62,r73,r84,r95}) , ({r3 , ({r45} {r7,r18,r29,r40,r51,r62,r73,r84,r95}) , ({r49} , ({r56} {r7,r18,r29,r40,r51,r62,r73,r84,r95}) , ({r , ({r67} {r7,r18,r29,r40,r51,r62,r73,r84,r95}) , ({r71} , ({r78} {r7,r18,r29,r40,r51,r62,r73,r84,r95}) ,

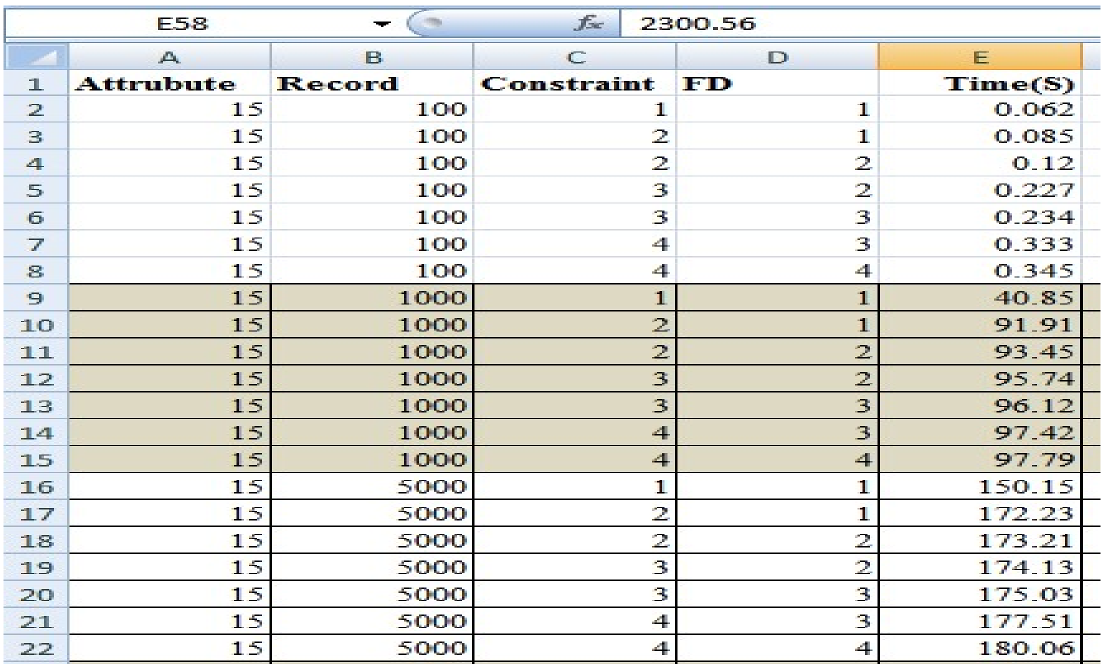
Basic Encryption+iterations

TID	A	B	C	D	E	F	G	H
r1	a1	27*0*11*0	c1	d1	e1	f1	g1	h1
r2	a2	b2	c2	d2	e2	f2	g2	h2
r3	a3	b3	c3	d3	e3	f3	g1	h1
r4	a4	b4	c4	d4	e2	f2	g3	h3
r5	a5	27*4*11*4	c1	d5	e4	f4	g4	h4
r6	a6	b6	c5	d6	e5	f5	g5	h5
r7	a1	27*6*11*0	c6	d1	e6	f6	g3	h3
r8	a7	b7	c2	d7	e7	f7	g6	h6

Execution time 0,0620 s

Figure 22: Obtained Result with 1 FD and 1 SC using PEM-M

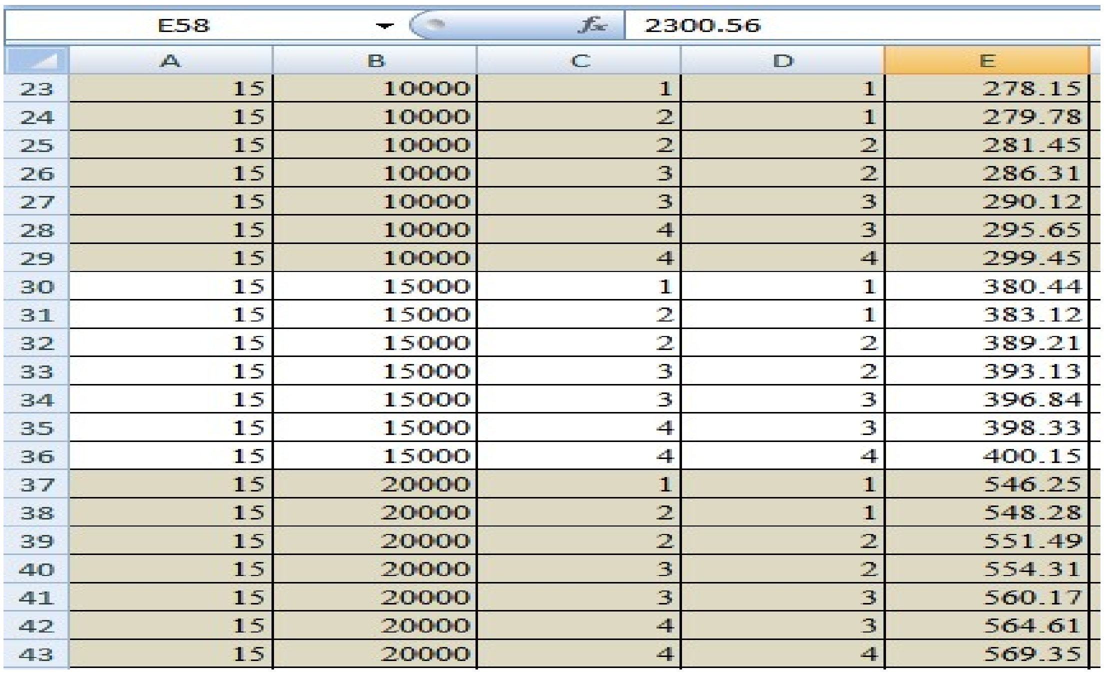
The same process will be used to evaluate the others performances. Time is evaluated with the function Chrono described in Appendix D



E58 | f_x 2300.56

	A	B	C	D	E
1	Attrubute	Record	Constraint	FD	Time(S)
2	15	100	1	1	0.062
3	15	100	2	1	0.085
4	15	100	2	2	0.12
5	15	100	3	2	0.227
6	15	100	3	3	0.234
7	15	100	4	3	0.333
8	15	100	4	4	0.345
9	15	1000	1	1	40.85
10	15	1000	2	1	91.91
11	15	1000	2	2	93.45
12	15	1000	3	2	95.74
13	15	1000	3	3	96.12
14	15	1000	4	3	97.42
15	15	1000	4	4	97.79
16	15	5000	1	1	150.15
17	15	5000	2	1	172.23
18	15	5000	2	2	173.21
19	15	5000	3	2	174.13
20	15	5000	3	3	175.03
21	15	5000	4	3	177.51
22	15	5000	4	4	180.06

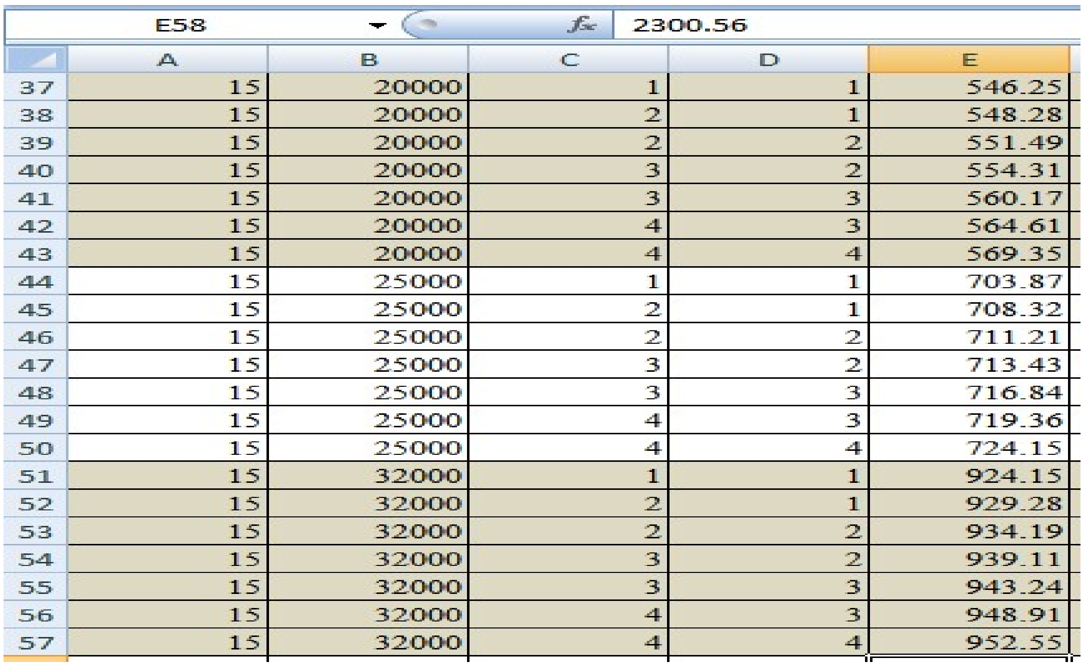
Figure 23: Execution Time for 100, 1000, and 5000 Records using PEM-M



E58 | f_x 2300.56

	A	B	C	D	E
23	15	10000	1	1	278.15
24	15	10000	2	1	279.78
25	15	10000	2	2	281.45
26	15	10000	3	2	286.31
27	15	10000	3	3	290.12
28	15	10000	4	3	295.65
29	15	10000	4	4	299.45
30	15	15000	1	1	380.44
31	15	15000	2	1	383.12
32	15	15000	2	2	389.21
33	15	15000	3	2	393.13
34	15	15000	3	3	396.84
35	15	15000	4	3	398.33
36	15	15000	4	4	400.15
37	15	20000	1	1	546.25
38	15	20000	2	1	548.28
39	15	20000	2	2	551.49
40	15	20000	3	2	554.31
41	15	20000	3	3	560.17
42	15	20000	4	3	564.61
43	15	20000	4	4	569.35

Figure 24: Execution Time for 10000, 15000 and 20000 Records using PEM-M



E58 | 2300.56

	A	B	C	D	E
37	15	20000	1	1	546.25
38	15	20000	2	1	548.28
39	15	20000	2	2	551.49
40	15	20000	3	2	554.31
41	15	20000	3	3	560.17
42	15	20000	4	3	564.61
43	15	20000	4	4	569.35
44	15	25000	1	1	703.87
45	15	25000	2	1	708.32
46	15	25000	2	2	711.21
47	15	25000	3	2	713.43
48	15	25000	3	3	716.84
49	15	25000	4	3	719.36
50	15	25000	4	4	724.15
51	15	32000	1	1	924.15
52	15	32000	2	1	929.28
53	15	32000	2	2	934.19
54	15	32000	3	2	939.11
55	15	32000	3	3	943.24
56	15	32000	4	3	948.91
57	15	32000	4	4	952.55

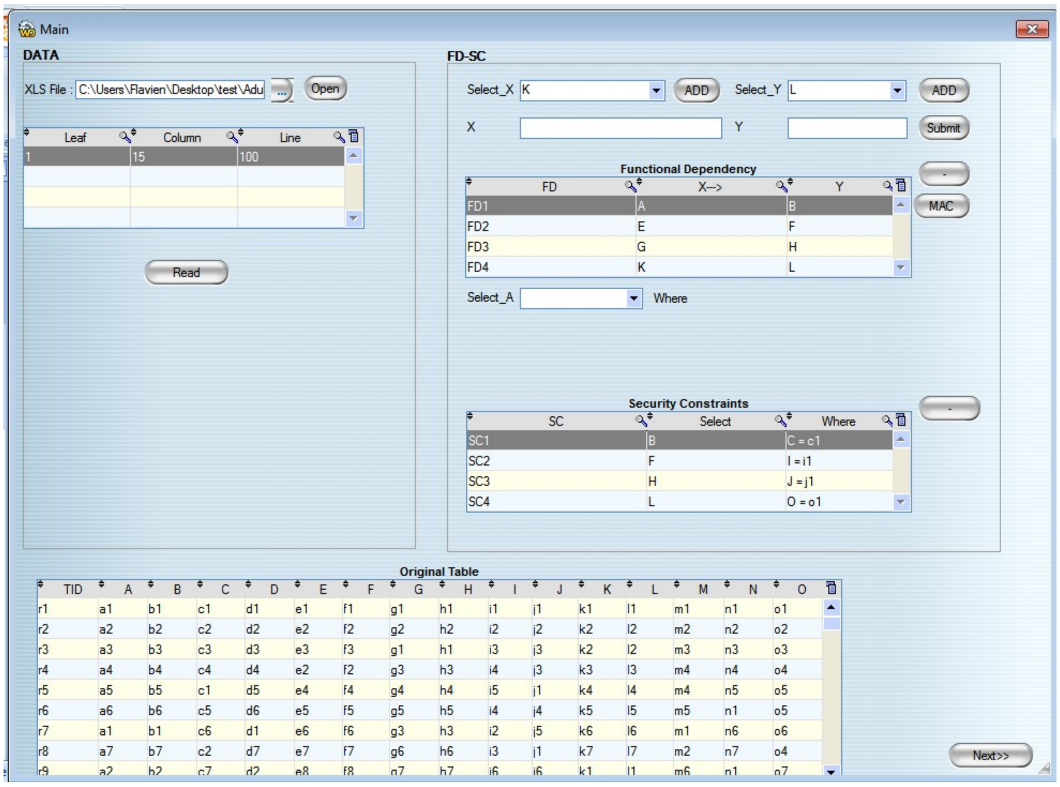
Figure 25: Execution Time for 20000, 25000 and 32000 Records using PEM-M

Based on the obtained results, it is noticeable that performance is better with less number of records. In Figure 25 with 100 records, execution time is 0.062s like minimum time and 0.345s like maximum time depending on different parameter. It is also noticed that when number of record increases, execution time also increases and the system can take a lot of time to fix functional dependency attacks problems.

Comparing to what have been done in [1], the first thing to noticed is that experiments in [1] were done using Adult and Orders databases as explained in section 2.1, and Figure 3 shows performance evaluation results. But, these results could not be used for comparison with the results obtained in the experiments done in PEM-M since different databases and parameters were used. Therefore, to obtain a reliable comparison results, Algorithm_1 used in PEM was run with TEST dataset using 100 records and 32k records, with 4 FDs and 4 SCs. Obtained results are as follow:

7.5 Results Obtained using PEM

The same process is used to evaluate execution time for PEM. For 100 records, FDs and SCs described in Section 3 were used as shown in Figure 26.



Leaf	Column	Line
1	15	100

Functional Dependency

FD	X-->	Y
FD1	A	B
FD2	E	F
FD3	G	H
FD4	K	L

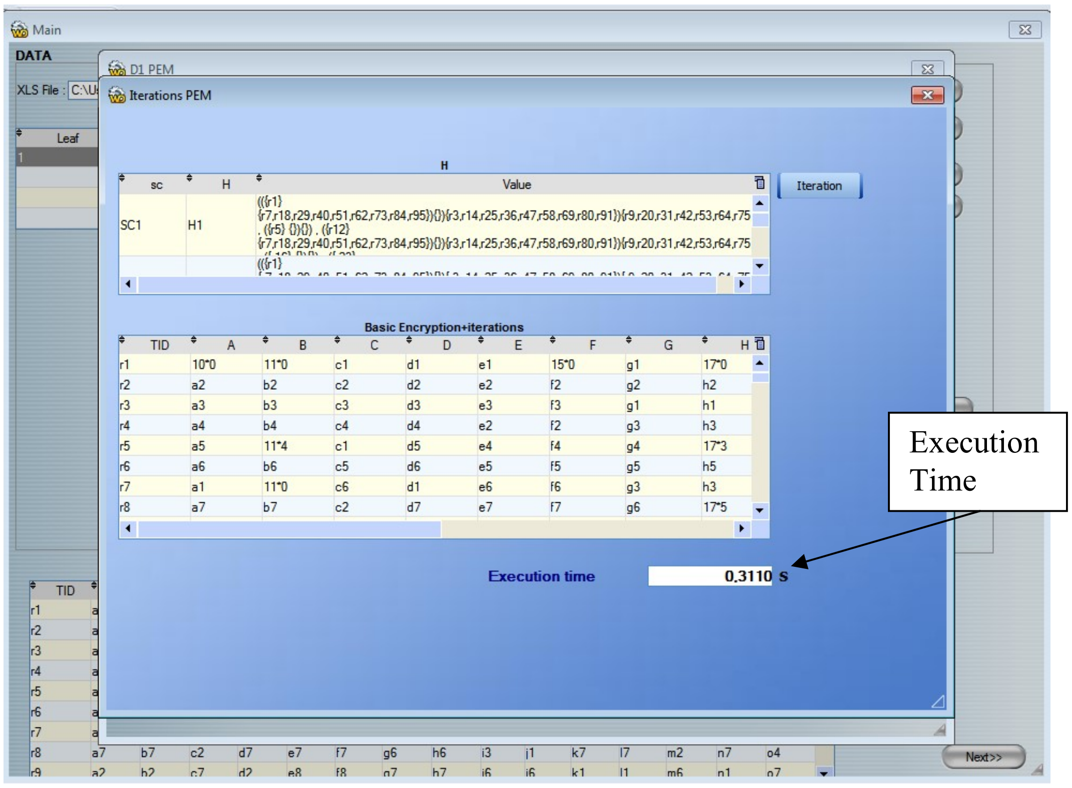
Security Constraints

SC	Select	Where
SC1	B	C = c1
SC2	F	I = i1
SC3	H	J = j1
SC4	L	O = o1

Original Table

TID	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
r1	a1	b1	c1	d1	e1	f1	g1	h1	i1	j1	k1	l1	m1	n1	o1
r2	a2	b2	c2	d2	e2	f2	g2	h2	i2	j2	k2	l2	m2	n2	o2
r3	a3	b3	c3	d3	e3	f3	g1	h1	i3	j3	k2	l2	m3	n3	o3
r4	a4	b4	c4	d4	e2	f2	g3	h3	i4	j3	k3	l3	m4	n4	o4
r5	a5	b5	c1	d5	e4	f4	g4	h4	i5	j1	k4	l4	m4	n5	o5
r6	a6	b6	c5	d6	e5	f5	g5	h5	i4	j4	k5	l5	m5	n1	o5
r7	a1	b1	c6	d1	e6	f6	g3	h3	i2	j5	k6	l6	m1	n6	o6
r8	a7	b7	c2	d7	e7	f7	g6	h6	i3	j1	k7	l7	m2	n7	o4
r9	a2	b2	c7	d2	e8	f8	g7	h7	i6	j6	k1	l1	m6	n1	o7

Figure 26: Parameters used for 100 Records for PEM



H

sc	H	Value
SC1	H1	(({r1} {r7,r18,r29,r40,r51,r62,r73,r84,r95})){})(r3,r14,r25,r36,r47,r58,r69,r80,r91})(r9,r20,r31,r42,r53,r64,r75 . ({r5} {}){}) . ({r12} {r7,r18,r29,r40,r51,r62,r73,r84,r95})){})(r3,r14,r25,r36,r47,r58,r69,r80,r91})(r9,r20,r31,r42,r53,r64,r75
		(({r1}

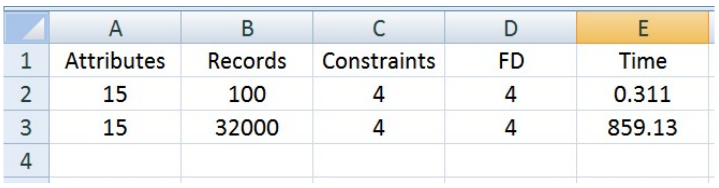
Basic Encryption+iterations

TID	A	B	C	D	E	F	G	H
r1	10*0	11*0	c1	d1	e1	15*0	g1	17*0
r2	a2	b2	c2	d2	e2	f2	g2	h2
r3	a3	b3	c3	d3	e3	f3	g1	h1
r4	a4	b4	c4	d4	e2	f2	g3	h3
r5	a5	11*4	c1	d5	e4	f4	g4	17*3
r6	a6	b6	c5	d6	e5	f5	g5	h5
r7	a1	11*0	c6	d1	e6	f6	g3	h3
r8	a7	b7	c2	d7	e7	f7	g6	17*5

Execution time 0,3110 s

Figure 27: Execution time using 100 Records 4 FDs and 4 SCs for PEM

The same parameters were used for 32K Records and the results obtained are summarized in Table 28.



	A	B	C	D	E
1	Attributes	Records	Constraints	FD	Time
2	15	100	4	4	0.311
3	15	32000	4	4	859.13
4					

Figure 28: Execution Time for 100 and 32K Records using PEM

From Figure 28, it is shown that using TEST dataset with parameters described above, PEM performs 0.311 with 100 records and 859.13 with 32K records.

7.6 Results Comparison

During experiments, PEM-M performs 0.345s using 100 records and 952.55s using 32K records. Table 27 can be used to compare PEM and PEM-M in the term of performance.

Table 27: Results Comparison

PEM			PEM-M		
SCs	FDs	Time(s)	SCs	FDs	Time(s)
4	4	0.311	4	4	0.345
4	4	859,13	4	4	952.55

Based on obtained results as presented in Table 27, it is shown in the term of execution time that PEM is better performing than PEM-M using 100 records and 32K records which are the minimum and the maximum number of records used for experiments. This can be explained by the fact that for encryption, concatenation was used to prevent double encryption as explained in Problem 1 Section 2. Therefore, that concatenation of value of column TID with the main value to encrypt contributed to improve accuracy and make the risk of double encryption equal to 0%.

Chapter 8

CONCLUSION AND FUTURE WORK

Based on what have been done in the main article and in this report, PEM is a good method to adopt if there is a need of defending system against FDs attacks. From chapter 2 to chapter 5, PEM was analyzed in order to better understand its technique and its functioning. Therefore, some problems were detected in PEM method proposed in [1], and to improve accuracy, problem_1 was adjusted so that Algorithm_1 [1] will consider the case when |Sv|=|Ev|. The way of encryption was also changed in order to prevent the case when system will face double encryption. And finally software which can fix functional dependency attacks problems by taking a dataset as input and generate a secure dataset in output was proposed. About experiments some tests were made in Test dataset and the comparison result was made between PEM proposed in [1] and PEM-M proposed in this paper using 100 records and 32k records. The proposed technique (PEM-M) improved the one proposed in [1] (PEM) in the term of accuracy because, as said before, the risk of double encryption is 0%, but in term of performance PEM-M still have to be improved because it was noticed that method proposed in [1] (PEM) is faster. About those problems which did not affect implementation of the technique on partial encryption, the future works will be done in order to optimize the way of preventing and fixing functional dependencies attacks.

REFERENCES

[1] B.Dong, H.W. Wang. (2018) "Secure partial encryption with adversarial functional dependency constraints in the database-as-a-service model", *Data and Knowledge Engineering„ Vol. 116*, pp.1-20

[2] Mamalis, T.Patel, "Database Security-Attacks and Control Methods". (2016) *International Journal of Information Science and Techniques, Vol. 6, No 1/2*, pp.175-183

[3] P.D. Stachour, B.Thuraisingham. (1990) "Design of LVD: multilevel Secure Relational Database Management System". *Transactions on Knowledge and Data Engineering, Vol 2, No 2*, pp.1-21

[4] A.H.Almutari, A.H.Airuwaili. (2012). "Security in Database System", *Global journal of Computer Sciences and Technology, Vol. 12, No 17*, pp.1-7.

[5] R.Kohavi, B,Becker. (1996). "Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid", *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* [Online] available https://datahub.io/machine-learning/adult [Accessed 17-02-2020]

[6] Wikipedia contributors. (2019). Database. In *Wikipedia, The Free Encyclopedia*. From https://en.wikipedia.org/w/index.php?title=Database&oldid=938672211, [Accessed 22-12-2019]

[7] J.V.Loon. (2008). “Data Security Concept and Approach”, *Seminar in database System, vol. 9, No 265*, pp.1-22

[8] R.StankovaKraleva, V.Kralev. (2018). “Design and Analysis of a Relational Database for Behavioral Experiments Data”, *International Journal of Online Engineering, Vol. 14, No 2*, pp.1-17

[9] A.Brodsky,C.Farkas, S.Jajodia. (2000). “Secure Database: Constraints, Inference Channels, and Monitoring Disclosures”, *IEEE Transaction on Knowledge and Data Engineering, 2000, Vol12, No 6*, pp.1-20

[10] Encryption. (2019). *Wikipedia, The Free Encyclopedia*. From https://simple.wikipedia.org/w/index.php?title=Encryption&oldid=6628741. [Accessed 13-02-2019]

[11] Q.-A.Kester. (2012) “A crypto system Based on Vigenere Cipher with Varying Key”, *Information Technology and Advanced Computing, Vol. 1, No 10*, pp.108-113

[12] Wikipedia contributors. (2019). “Vigenère cipher”. In *Wikipedia, The Free Encyclopedia*,
From https://en.wikipedia.org/w/index.php?title=Vigen%C3%A8re_cipher&oldid=942750659, [Accessed: 28-12-2019]

[13] T.-A.Su, G.Ozsoyoglu. (1991). “Controlling FD and MVD Inferences in Multilevel Relational Database Systems", *IEEE Transaction on Knowledge and Data Engineering, Vol 3, No 4*, pp.474-485

[14] K.J. Singh, R.Manimegalai. (2015). “Evolution of Encryption Techniques and Data Security Mechanism”, *World Apply Science journal, Vol. 33, No 10*, pp.1597-1613

[15] M.Morgenstern. (1988) “Controlling Logical Inference in Multilevel Database Systems”, *Computer science, Vol.5*, pp.245-255

[16] B.Dong, W.Wang, J.Yang. (2016) “Secure Data Outsourcing with Adversarial Data Dependency Constraints”, *IEEE International Conference on Intelligent Data and Security, Vol. 2*, pp.73-78

[17] M.B Thuraisingham. (1987) “Security Checking in Relational Database Management Systems Augmented with Inferences Engines”, *Computer and Security, Vol.6*, pp.479-492

[18] Wikipedia contributors. (2019). “Windev”. In *Wikipedia, The Free Encyclopedia libre*, From https://en.wikipedia.org/w/index.php?title=Vigen%C3%A8re_cipher&oldid=942750659, [Accessed 24-12-2019]

[19] PCsoft. (2019). ”Windev platform” [Online], Available: https://www.windev.com/windev/index.html, [Accessed 24-12-2019]

APPENDICES

Appendix A: Load input, Vigenere Encryption and MAC

Appendix A.1: Load database file

This code was used for uploading of file in the system by clicking on button **OPEN** in the main form

```
IF TableOccurrence(Table_descript)<>0Then
TableDeleteAll(Table_descript)
END
I is an integer

sFichier= SAI_FIC//allow the file link to the variable
fich=xlsOuvre(sFichier,xlsEcriture)//Open   file   and   allow
content to variable fich
IF ErrorDetected=False Then
   nbfeuille is an integer=xlsNblift(fich)
   FOR i=1 TO nblift PAS 1
      xlsFeuilleEnCours(fich,i)
      nbcolumn is an integer=xlsNbColumn(fich,False)
      nbline is an integer=xlsNbLine(fich,False)
      TableADDLine(Table_descript,nblift,nbcolumn,nbline)
      END
      ELSE
      Info("Error")
      END
```

Appendix A.2: Load file content in the table

This code was used to load file content in the main table by clicking on button **READ** in the main form

```
      i,j,nb,k,f are integer

      xlsCurrentLift(fich,Table_descript.Line)
      nblineis an integer=xlsNbLigne(fich,Faux)
      nbcol is an integer=xlsNbColumn(fich,Faux)
      IF (nbcol+1<TableTID..NumberColumn) Then
      nb=TableTID..NumberColumn
      FORi=nb TO (nbcol+1) PAS -1
      ChampSupprime(TableEnumèreColonne(TableTID, i))
      ChampSupprime(TableEnumèreColonne(Tablecopietid,   i
))

      END
      END
```

```
IF (nbcol+1>TableTID..NombreColonne) Then//Adjust column number
nb=TableTID..NumberColumn
WHILEnbcol+1<>nb
TableADDColumn(TableTID)
TableADDColumn(Tablecopietid)
END
END
TableDeleteAll(TableTID)
TableDeleteAll(Tablecopietid)

{TableTID..Nom + "." + TableEnumèreColonne(TableTID, 1), indChamp}..Titre ="TID"// change column title and put TID in the first column
{Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, 1), indChamp}..Titre ="TID"
k=2
FORi=1 A nbcol PAS 1// For i=1 to nbcol change others titles
{TableTID..Nom + "." + TableEnumèreColonne(TableTID, k), indChamp}..Titre =xlsDonnée(fich,1,i)
{Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, k), indChamp}..Titre =xlsDonnée(fich,1,i)
k=k+1
END
TableADjust(TableTID)
TableADjust(Tablecopietid)

// Upload attribute name in the application boxes

FORi=2 A TableTID..NombreColonne
ListeAjoute(Combo_X,{TableTID..Nom + "." + TableEnumèreColonne(TableTID, i), indChamp}..Titre)
END
FORi=2 A TableTID..NombreColonne
ListeAjoute(Combo_Y,{TableTID..Nom + "." + TableEnumèreColonne(TableTID, i), indChamp}..Titre)
END
FORi=2 A TableTID..NombreColonne
ListeAjoute(Combo_X1,{TableTID..Nom + "." + TableEnumèreColonne(TableTID, i), indChamp}..Titre)
END
FOR i=2 A TableTID..NombreColonne
ListeAjoute(Combo_X2,{TableTID..Nom + "." + TableEnumèreColonne(TableTID, i), indChamp}..Titre)
END

k=1
```

```
FOR i=2A  nbligne PAS 1
TableAjoute(TableTID,("r"+(TableOccurrence(Tablecopietid)+1)))
f=1;
TableAjoute(Tablecopietid,("r"+(TableOccurrence(Tablecopietid)+1)))
FOR j=2 A nbcol+1 PAS 1

TableTID[k,j]=xlsDonnée(fich,i,f,Faux)
Tablecopietid[k,j]=xlsDonnée(fich,i,f,Faux)
f=f+1
END
k=k+1

END
```

Appendix A.3: Vigenere Encryption

This code represents procedure used for Vigenere encryption

```
PROCEDURE Vigenere_Crypt(ch is a string)
i is an integer
ch1 is a string
ch1=""

FOR i=1 A Length(ch) PAS 1
IF HLitRecherchePremier(Alpahabet,Alpahabet.letter,ch[[i]],hIdentique)=True Then//Check letter in the data base
IF ch1="" Then
ch1=(Alpahabet.Number+10)modulo(37)
ELSE

ch1=ch1+"*"+(Alpahabet.Number+10)modulo(37)
END

END
END
RETURN ch1
```

Appendix A.4: Enter Functional Dependencies

This code is used to insert Functional Dependencies in the system. An example of result is shown in Figure 11 after selecting FD and clicking on button **SUBMIT.**

```
i est un entier
i=TableOccurrence(Table_fd)
```

```
TableAjouteLigne(Table_fd,"FD"+(i+1),Saisie_X,Saisie_Y)
TableAffiche(Table_fd)
Saisie_X=""
Saisie_Y=""
```

Appendix A.5: Minimum Attribute Cover (MAC)

This code is used for Minimum Attribute Cover, which helps of reordering of FDs. For better understanding, three parts are used

Appendix A.5.1: Weighted Attribute

In this part, the goal is to calculate the weight of each attribute. An example of result is shown in second table in Figure 13. Getting after clicking on button **WEIGHT**

```
i,j,t are integer
wrd, rep are string
wrd="";rep="no";

FOR  i=1  A  TableFD..Occurrence  PAS  1//Extract attribute name
IF ChaîneOccurrence(TableFD[i].LHS,",")=0 Then
IF TableW..Occurrence>0 Then
rep="no";
FOR t=1 A TableW..Occurrence
IF (TableFD[i].LHS=TableW[t].R) Then
rep="yes"
END
END
IFrep="no" Then
TableADDLine(TableW,TableFD[i].LHS)
END
ELSE
TableADDLine(TableW,TableFD[i].LHS)
END
ELSE

FOR j=1 A Length(TableFD[i].LHS) PAS 1
IFTableFD[i].LHS[[j]]<>"," Then
wrd=wrd+TableFD[i].LHS[[j]]
END
IF  (TableFD[i].LHS[[j]]=",")  OR (j=Length(TableFD[i].LHS)) Then
rep="no"
```

```
 FOR t=1 A TableW..Occurrence PAS 1
IF (wrd=TableW[t].R) Then
rep="yes"
END
END
IFrep="non" Then
TableADDLine(TableW,wrd)
wrd=""
END

END


END
END
rep="no"
FOR t=1 A TableW..Occurrence PAS 1
IF TableFD[i].RHS=TableW[t].R ALORS
rep="yes"
END
END
IF rep="no" Then
TableADDLine(TableW,TableFD[i].RHS)

END
wrd=""
END



FOR j=1 A TableW..Occurrence PAS 1

IF ChaîneOccurrence(Saisie1,TableW[j,1])>0 Then
TableW[j,2]=ChaîneOccurrence(Saisie1,TableW[j,1])
END

END
```

Appendix A.5.2: Minimum cover

In this second part of MAC, the following code is used to generate the set of minimum attribute cover. A result illustration can be seen in the box A in Figure 14, after clicking on button **MAC**

```
i,j,maxi are integer
mac,maxi1 are string

mac="";
FOR i=1 A TableFD..Occurrence PAS 1
maxi=0; maxi1=""
FOR j=1 A TableW..Occurrence PAS 1
IF TableW[j].W1>maxi Then
maxi=TableW[j].W1
maxi1=TableW[j].R
END
END

IF ChaîneOccurrence(TableFD[i].xy,maxi1)>0 Then
FOR j=1 A TableW..Occurrence PAS 1
IF    ChaîneOccurrence(TableFD[i].xy,TableW[j].R)>0 Then
TableW[j].W1=TableW[j].W1-1
END
END
TableDisplay(TableW)
END
IF mac="" Then
mac=maxi1
ELSE
IF ChaîneOccurrence(mac,maxi1)=0 Then
mac=mac+","+maxi1
END
END
END

Saisie2=mac//Display MAC in the form
```

Appendix A.5.3: Change FDs order

In this last part of MAC, the following code is used to the reordering of FDs using minimum cover obtained in the previous part. An example of result is shown in third table in Figure 14 after clicking on button **NEW ORDER**

```
i,j,k are integer
ch,rep are string
ch="";i=1 ; rep="no"
FOR i=1 A Length(Saisie2) PAS 1//Go through MAC
rep="no"
IF (Saisie2[[i]]<>",")  Then
ch=ch+Saisie2[[i]]
```

```
END
IF (Saisie2[[i]]=",") OU (i=Length(Saisie2))Then
FOR j=1 A TableFD..Occurrence PAS 1
IF ChaîneOccurrence(TableFD[j].xy,ch)>0 Then
IF TableFD1..Occurrence=0 Then
TableADDLine(TableFD1,TableFD[j].FD,TableFD[j].LHS,TableFD[j].RHS,TableFD[j].xy)
ELSE
FOR k=1 A TableFD1..Occurrence PAS 1
IF TableFD[j].FD=TableFD1[k].FD Then
rep="yes"
END
END
IF rep="no" Then
TableADDLine(TableFD1,TableFD[j].FD,TableFD[j].LHS,TableFD[j].RHS,TableFD[j].xy)
END
END
END

END
ch=""
END

END
FOR i=1 A TableFD..Occurrence PAS 1
rep="no"
FOR j=1 A TableFD1..Occurrence PAS 1
IF TableFD[i].FD=TableFD1[j].FD Then
rep="yes"
END
END
IF rep="non" END
TableADDLine(TableFD1,TableFD[i].FD,TableFD[i].LHS,TableFD[i].RHS,TableFD[j].xy)
END
END
TableDisplay(TableFD1)

TableDeleteAll(Main.Table_fd)//Delete FDs to add FDs with new order
FOR i=1 A TableFD1..Occurrence PAS 1
TableADDLine(Main.Table_fd,TableFD1[i].FD,TableFD1[i].LHS,TableFD1[i].RHS)//Add FDs with new order
END
```

Appendix A.6: Enter SCs and Basic encryption

These codes are used to get SCs and basic encryption table using FDs and security constraints. An example of obtained result is shows in the first table in Figure 18 for PEM and Figure 19 for PEM-M

Appendix A.6.1: Enter SCs for PEM of PEM-M

This process is includes in the code using for basic encryption so following Appendices will be referred for this part.

Appendix A.6.2: Basic Encryption for PEM

```
i,j,k are integer
colA,colB,sr are string

FOR i=1 A Tablecopietid..NumberColumn PAS 1

IF ({Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, i), indChamp}..Titre =Combo_X1) Then
colA={Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, i), indChamp}..Titre
j=i
END
IF ({Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, i), indChamp}..Titre=Combo_X2..ValeurAffichée) Then
colB={Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, i), indChamp}..Titre
k=i
END
END
IF (sai_j="")  Then
sai_j=j
ELSE
IF (sai_j<>"") ET (ChaîneOccurrence(sai_j,j)=0) ALORS
sai_j=sai_j+","+j
END
END
IF sai_k="" Then
sai_k=k
ELSE
IF (sai_k<>"") ET (ChaîneOccurrence(sai_k,k)=0) Then
sai_k=sai_k+","+k
END
```

```
END
i=TableOccurrence(Table_fd1)
IF (Combo_X3="Equal to") Then
sr=""
FOR t=1  ATableOccurrence(Tablecopietid) PAS 1
IF (Tablecopietid[t,k]=Saisie1) Then
IF sr="" Then
sr="r"+t
ELSE
sr=sr+",r"+t
END
IF (ChaîneOccurrence(Tablecopietid[t,j],"*")=0) Then
Tablecopietid[t,j]=Vigenere_Crypt(Tablecopietid[t,j])
END
END
END
TableADDLine(Table_fd1,"SC"+(i+1),Combo_X1,Combo_X2+" = "+Saisie1,sr)
TableDisplay(Table_fd1)
TableDisplay(Tablecopietid)
Combo_X1=""
Combo_X2=""
Combo_X3=""
Saisie1=""
Combo_X2..Visible=Faux
Combo_X3..Visible=Faux
Saisie1..Visible=Faux
BtnADD..Visible=Faux
END
IF (Combo_X3..ValeurAffichée="Less than") Then
sr=""
FOR t=1  ATableOccurrence(Tablecopietid) PAS 1
IF     (Tablecopietid[t,k]..ValeurAffichée<Saisie1) Then
IF sr="" Then
sr="r"+t
ELSE
sr=sr+",r"+t
END
IF (ChaîneOccurrence(Tablecopietid[t,j],"*")=0) Then
Tablecopietid[t,j]=Vigenere_Crypt(Tablecopietid[t,j])
```

```
END

END
END
TableADDLine(Table_fd1,"SC"+(i+1),Combo_X1,Combo_X2+"<"+Saisie1,sr)
TableDisplay(Table_fd1)
Combo_X1=""
Combo_X2=""
Combo_X3=""
Saisie1=""
Combo_X2..Visible=False
Combo_X3..Visible=False
Saisie1..Visible=False
BtnADD..Visible=False
END
IF (Combo_X3..ValeurAffichée="More than")  Then
sr=""
FORt=1  ATableOccurrence(Tablecopietid) PAS 1

IF     (Tablecopietid[t,k]..ValeurAffichée>Saisie1) Then
IF sr="" Then
sr="r"+t
ELSE
sr=sr+",r"+t
END

IF (ChaîneOccurrence(Tablecopietid[t,j],"*")=0) Then
Tablecopietid[t,j]=Vigenere_Crypt(Tablecopietid[t,j])

END

END
END
TableADDLine(Table_fd1,"SC"+(i+1),Combo_X1..ValeurAffichée,Combo_X2+">"+Saisie1,sr)
TableDisplay(Table_fd1)
Combo_X1=""
Combo_X2=""
Combo_X3=""
Saisie1=""
Combo_X2..Visible=False
Combo_X3..Visible=False
Saisie1..Visible=False
BntADD..Visible=False
END
IF (Combo_X3..ValeurAffichée="Different from") Then
sr=""
FOR t=1  ATableOccurrence(Tablecopietid) PAS 1
```

```
IF (Tablecopietid[t,k]<>Saisie1) Then

IF sr="" Then
sr="r"+t
ELSE
sr=sr+",r"+t
END
IF (ChaîneOccurrence(Tablecopietid[t,j],"*")=0) Then


Tablecopietid[t,j]=Vigenere_Crypt(Tablecopietid[t,j])

END

END
END
TableADDLine(Table_fd1,"SC"+(i+1),Combo_X1,Combo_X2+"<> "+Saisie1,sr)
TableDisplay(Table_fd1)
Combo_X1=""
Combo_X2=""
Combo_X3=""
Saisie1=""
Combo_X2..Visible=False
Combo_X3..Visible=False
Saisie1..Visible=False
BtnADD..Visible=False
END
```

Appendix A.6.3: Basic Encryption for PEM-M

```
i,j,k are integer
colA,colB,sr are string

FOR i=1 A Tablecopietid..NumberColumn PAS 1

IF ({Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, i), indChamp}..Titre =Combo_X1) Then
colA={Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, i), indChamp}..Titre
j=i
END
IF ({Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, i), indChamp}..Titre=Combo_X2..ValeurAffichée) Then
colB={Tablecopietid..Nom + "." + TableEnumèreColonne(Tablecopietid, i), indChamp}..Titre
k=i
END
```

```
END
IF (sai_j="")  Then
sai_j=j
ELSE
IF  (sai_j<>"")  ET  (ChaîneOccurrence(sai_j,j)=0)
ALORS
sai_j=sai_j+","+j
END
END
IF sai_k="" Then
sai_k=k
ELSE
IF (sai_k<>"") ET (ChaîneOccurrence(sai_k,k)=0) Then
sai_k=sai_k+","+k
END

END

i=TableOccurrence(Table_fd1)

IF (Combo_X3="Equal to") Then
sr=""
FOR t=1  ATableOccurrence(Tablecopietid) PAS 1

IF (Tablecopietid[t,k]=Saisie1) Then
IF sr="" Then
sr="r"+t
ELSE
sr=sr+",r"+t
END
IF (ChaîneOccurrence(Tablecopietid[t,j],"*")=0) Then
Tablecopietid[t,j]=Vigenere_Crypt(Tablecopietid[t,1
]+Tablecopietid[t,j])

END

END
END

TableADDLine(Table_fd1,"SC"+(i+1),Combo_X1,Combo_X2
+" = "+Saisie1,sr)
TableDisplay(Table_fd1)
TableDisplay(Tablecopietid)
Combo_X1=""
Combo_X2=""
Combo_X3=""
Saisie1=""
Combo_X2..Visible=Faux
Combo_X3..Visible=Faux
Saisie1..Visible=Faux
BtnADD..Visible=Faux
END
```

```
IF (Combo_X3..ValeurAffichée="Less than") Then
sr=""
FOR t=1  ATableOccurrence(Tablecopietid) PAS 1

IF     (Tablecopietid[t,k]..ValeurAffichée<Saisie1) Then
IF sr="" Then
sr="r"+t
ELSE
sr=sr+",r"+t
END
IF (ChaîneOccurrence(Tablecopietid[t,j],"*")=0) Then
Tablecopietid[t,j]=Vigenere_Crypt(Tablecopietid[t,1]+Tablecopietid[t,j])

END

END
END
TableADDLine(Table_fd1,"SC"+(i+1),Combo_X1,Combo_X2+" < "+Saisie1,sr)
TableDisplay(Table_fd1)
Combo_X1=""
Combo_X2=""
Combo_X3=""
Saisie1=""
Combo_X2..Visible=False
Combo_X3..Visible=False
Saisie1..Visible=False
BtnADD..Visible=False
END
IF (Combo_X3..ValeurAffichée="More than")  Then
sr=""
FORt=1  ATableOccurrence(Tablecopietid) PAS 1

IF     (Tablecopietid[t,k]..ValeurAffichée>Saisie1) Then
IF sr="" Then
sr="r"+t
ELSE
sr=sr+",r"+t
END

IF (ChaîneOccurrence(Tablecopietid[t,j],"*")=0) Then
Tablecopietid[t,j]=Vigenere_Crypt(Tablecopietid[t,1]+Tablecopietid[t,j])

END

END
END
```

```
TableADDLine(Table_fd1,"SC"+(i+1),Combo_X1..ValeurAffichée,Combo_X2+" > "+Saisie1,sr)
TableDisplay(Table_fd1)
Combo_X1=""
Combo_X2=""
Combo_X3=""
Saisie1=""
Combo_X2..Visible=False
Combo_X3..Visible=False
Saisie1..Visible=False
BntADD..Visible=False
END
IF (Combo_X3..ValeurAffichée="Different from") Then
sr=""
FOR t=1  ATableOccurrence(Tablecopietid) PAS 1

IF (Tablecopietid[t,k]<>Saisie1) Then

IF sr="" Then
sr="r"+t
ELSE
sr=sr+",r"+t
END
IF (ChaîneOccurrence(Tablecopietid[t,j],"*")=0) Then


Tablecopietid[t,j]=Vigenere_Crypt(Tablecopietid[t,1]+Tablecopietid[t,j])

END

END
END
TableADDLine(Table_fd1,"SC"+(i+1),Combo_X1,Combo_X2+"<> "+Saisie1,sr)
TableDisplay(Table_fd1)
Combo_X1=""
Combo_X2=""
Combo_X3=""
Saisie1=""
Combo_X2..Visible=False
Combo_X3..Visible=False
Saisie1..Visible=False
BtnADD..Visible=False
END
```

Appendix B: First part GMM

Appendix B.1: Robustness Checking

This code is used to check if there exists any evidence record in order to conclude whether system is robust or not. The result is obtaining by clicking on button **CHECK** in Figure 19. An example of result is shown in the second table.

```
i,j,k,numcol1,numcol2,f are integer
rec is a string
j=1;
FOR i=1 TOMain.Table_fd..Occurrence PAS 1

FOR j=1 TOTableTID..NombreColonne PAS 1
IF (Main.Table_fd[i].x={TableTID..Nom + "." + TableEnumèreColonne(TableTID, j), indChamp}..Titre) Then
numcol1=j;
END
IF (Main.Table_fd[i].y={TableTID..Nom + "." + TableEnumèreColonne(TableTID, j), indChamp}..Titre) Then
numcol2=j;
END
END
TableAjouteLigne(Tabledep,Main.Table_fd[i].fd,numcol1,numcol2)


Saisie1=numcol1; Saisie2=numcol2
FOR k=1 A TableTID..Occurrence PAS 1
rec="";
IF ChaîneOccurrence(TableTID[k,numcol2],"*")>0 Then
FOR f=1 TOTableOccurrence(TableTID) PAS 1
IF TableTID[f,numcol1]=TableTID[k,numcol1] Then
IF ChaîneOccurrence(TableTID[f,numcol2],"*")=0 ALORS
IF rec="" Then
rec= "r"+f;
ELSE
rec=rec+",r"+f
END
END
END
END

END
IF (ChaîneOccurrence(TableTID[k,numcol2],"*")>0) AND (rec<>"") THEN
TableADDLine(Table_check,"FD"+i+"=("+Main.Table_fd[i].x+"-->"+Main.Table_fd[i].y+")","r"+k,rec,"No robust","FD"+i,numcol1,numcol2)
```

```
END
IF (ChaîneOccurrence(TableTID[k,numcol2],"-")>0) AND (rec="") THEN
TableADDLine(Table_check,"FD"+i+"=("+Main.Table_fd[i].x+"-->"+Main.Table_fd[i].y+")","r"+k,rec,"Robust","FD"+i,numcol1,numcol2)
END
END

END
```

Appendix B.2: Generation of Buckets H

This code is used for implementation of set of Bucket H based on what have been done previously. An example of result is shown in the third table in Figure 16 after clicking on button **GENERATE H.**

```
i,j,k,i1,t are integer
h,sr,er,tev,i2,ev1 are string

FOR i=1 A Main.Table_fd1..Occurrence PAS 1
h="";  er=""; tev="";i2=""
FOR j=1 A Length(Main.Table_fd1[i].sr) PAS 1
sr="";
IF Main.Table_fd1[i].sr[[j]]="r" Then
i1=j+1;
WHILE     (i1<=Length(Main.Table_fd1[i].sr))     AND (Main.Table_fd1[i].sr[[i1]]<>",")
i2=i2+Main.Table_fd1[i].sr[[i1]]
i1=i1+1
END

sr="r"+i2
i2=""

IF h=""Then
h="(({"+sr+"} "
FOR k=1 TOTable_check..Occurrence PAS 1
IF Table_check[k].sr= sr Then
er=Table_check[k].er
h=h+"{"+Table_check[k].er+"})"
IF tev="" Then
tev=Table_check[k].er
ELSE
ev1="";
FOR t=1 TOLength(Table_check[k].er)
IF (Table_check[k].er[[t]]<>";")  Then
```

```
ev1= Table_check[k].er[[t]]
END
IF (Table_check[k].er[[t]]=";") OR (t=Length(Table_check[k].er) ) Then
IF ChaîneOccurrence(tev,ev1)=0 Then//StringCount
tev=tev+","+ev1
ev1=""

END
END
END

END
END
END

ELSE
h=h+" , ({"+sr+"} "
FOR k=1 TOTable_check..Occurrence PAS 1
IF Table_check[k].sr= srThEn
er=Table_check[k].er
h=h+"{"+Table_check[k].er+"})"
IF tev="" Then
tev=Table_check[k].er
ELSE
ev1="";
FOR t=1 TOLength(Table_check[k].er)
IF (Table_check[k].er[[t]]<>";")  Then
ev1= Table_check[k].er[[t]]
END
IF (Table_check[k].er[[t]]=";") OR (t=Length(Table_check[k].er) ) Then
IF ChaîneOccurrence(tev,ev1)=0 Then//StringCount
tev=tev+","+ev1
ev1=""

END
END
END
END
END
END

END

END
END
h=h+")"
TableADDLine(TableH_H,Main.Table_fd1[i].fd,"H"+(TableH_H..Occurrence+1),h)
TableADDLine(Table_h,Main.Table_fd1[i].sr,tev

END
```

Appendix C: Partial Encryption

These blocs of code are used for solving problem of FDs attacks. It will then goes through each Bucket and encrypt cells in sensitive or evidence record based of minimum encryption overhead. An example of result is obtained in second table in Figure 20 or Figure 21 based on the method used after clicking on button **ITERATION**

Appendix C.1: Partial Encryption for PEM

```
sr,i2 Are string
i,j,k,i1,tsont des entiers
ChronoStart()//Start counting execution time
FOR i=1 TOExpress_d1_PEM.Table_h..Occurrence PAS 1
i2=""
IF ChaîneOccurrence(Express_d1_PEM.Table_h[i].sr,"r")<=ChaîneOccurrence(Express_d1.Table_h[i].er,"r") Then// Occurrence of r
Saisie1=(Length(Express_d1.Table_h[i].sr))
FORj=1 TO (Length(Express_d1_PEM.Table_h[i].sr)) PAS 1
sr=""
IF Express_d1_PEM.Table_h[i].sr[[j]]="r" Then
i1=j+1;
WHILE (i1<=Length(Express_d1_PEM.Table_h[i].sr)) AND (Express_d1_PEM.Table_h[i].sr[[i1]]<>",")
i2=i2+Express_d1_PEM.Table_h[i].sr[[i1]]
i1=i1+1
END
sr="r"+i2
i2=""
Saisie1=sr
FOR k=1 TOExpress_d1_PEM.Table_check..Occurrence
IF sr=Express_d1_PEM.Table_check[k].sr Then
FOR t=1 TOTableTID..Occurrence
IF TableTID[t,1]=srThEn
IF ChaîneOccurrence(TableTID[t,Express_d1_PEM.Table_check[k].numcol1],"*")=0AThen
TableTID[t,Express_d1_PEM.Table_check[k].numcol1]=Vigenere_Crypt(TableTID[t,Express_d1_PEM.Table_check[k].numcol1])
END
```

```
END

END
sr=""
END
END
END
END
ELSE

FOR  j=1  TO  (Length(Express_d1_PEM.Table_h[i].er)) PAS 1
sr=""
IF Express_d1_PEM.Table_h[i].er[[j]]="r" Then
i1=j+1;
WHILE     (i1<=Length(Express_d1_PEM.Table_h[i].er)) AND (Express_d1_PEM.Table_h[i].er[[i1]]<>",")
i2=i2+Express_d1_PEM.Table_h[i].er[[i1]]
                    i1=i1+1
END
sr="r"+i2
i2=""
FOR k=1 TO Express_d1_PEM.Table_check..Occurrence
IF ChaîneOccurrence(Express_d1_PEM.Table_check[k].er,sr)<>0 Then
FOR t=1 TO TableTID..Occurrence
IF TableTID[t,1]=srTHen
IF ChaîneOccurrence(TableTID[t,Express_d1_PEM.Table_check[k].numcol2],"*")=0       Then
TableTID[t,Express_d1_PEM.Table_check[k].numcol2]=Vigenere_Crypt(TableTID[t,Express_d1_PEM.Table_check[k].numcol2])
END
END

END
END
END
END
sr=""
END
END
END

t=(ChronoStop())//Stop counting time
Execution_time=t/1000
TableDisplay(TableTID)
```

Appendix C.2 Partial Encryption for PEM-M

```
sr,i2 Are string
i,j,k,i1,tsont des entiers
ChronoStart()//Start counting execution time
FOR i=1 TOExpress_d1.Table_h..Occurrence PAS 1
i2=""
IF ChaîneOccurrence(Express_d1.Table_h[i].sr,"r")<=ChaîneOccurrence(Express_d1.Table_h[i].er,"r") Then// Occurrence of r

Saisie1=(Length(Express_d1.Table_h[i].sr))
FORj=1 TO (Length(Express_d1.Table_h[i].sr)) PAS 1
sr=""
IF Express_d1.Table_h[i].sr[[j]]="r" Then
i1=j+1;
WHILE (i1<=Length(Express_d1.Table_h[i].sr)) AND (Express_d1.Table_h[i].sr[[i1]]<>",")
i2=i2+Express_d1.Table_h[i].sr[[i1]]
i1=i1+1
END
sr="r"+i2
i2=""
Saisie1=sr
FOR k=1 TOExpress_d1.Table_check..Occurrence
IF sr=Express_d1.Table_check[k].sr Then
FOR t=1 TOTableTID..Occurrence
IF TableTID[t,1]=srThEn
IF ChaîneOccurrence(TableTID[t,Express_d1.Table_check[k].numcol1],"*")=0AThen
TableTID[t,Express_d1.Table_check[k].numcol1]=Vigenere_Crypt(TableTID[t,1]+TableTID[t,Express_d1.Table_check[k].numcol1])
END
END

END
sr=""
END
END
END
END
ELSE

FOR j=1 TO (Length(Express_d1.Table_h[i].er)) PAS 1
sr=""
IF Express_d1.Table_h[i].er[[j]]="r" Then
i1=j+1;
WHILE (i1<=Length(Express_d1.Table_h[i].er)) AND (Express_d1.Table_h[i].er[[i1]]<>",")
```

```
i2=i2+Express_d1.Table_h[i].er[[i1]]
                    i1=i1+1
END
sr="r"+i2
i2=""
FOR k=1 TO Express_d1.Table_check..Occurrence
IF ChaîneOccurrence(Express_d1.Table_check[k].er,sr)<>0 Then
FOR t=1 TO TableTID..Occurrence
IF TableTID[t,1]=srTHen
IF ChaîneOccurrence(TableTID[t,Express_d1.Table_check[k].numcol2],"*")=0      Then
TableTID[t,Express_d1.Table_check[k].numcol2]=Vigenere_Crypt(TableTID[t,1]+TableTID[t,Express_d1.Table_check[k].numcol2])
END
END

END
END
END
END
sr=""
END
END
END

t=(ChronoStop())//Stop counting time
Execution_time=t/1000
TableDisplay(TableTID)
```

Appendix D: Execution Time for PEM or PEM-M

Evaluation time is performed by CHRONO used in WD language. The structure is

ChronoStart() // To start counting

ChronoStop() // To stop counting.

It is noticeable in Appendix C.1 il Line 576 that execution time start and stop in Line 632