Information Systems Frontiers (2021) 23:811–833
https://doi.org/10.1007/s10796-021-10178-w

Approaches to Enforce Privacy in Databases: Classical to Information Flow-Based Models

R.K. Shyamasundar[1] · Pratiksha Chaudhary[1] · Arushi Jaiswal[1] · Aniket Kuiri[1]

Accepted: 20 July 2021 / Published online: 13 August 2021

Abstract
Ever since databases became an ubiquitous part of enterprises or businesses, security and privacy became a requirement. Traditionally, privacy was realized through various methods of database access control and relied much on the use of statically defined *views*, which are essentially logical constructs imposed over database tables that can alter or restrict the data that can be viewed by an user. Privacy is about the responsible maintenance of private information. This responsibility is hard to define, which is why laws are necessary. With a vast accumulation of personal data in databases, there has been a heightened awareness and concern about the storage and use of private information leading to privacy-related guidelines, regulations and legislations, Compliance with these regulations has become one of the major concerns for organizations and companies. Traditionally, privacy in databases (DBs) have been addressed through access control techniques including multi-level security (MLS) based on mandatory access control (MAC), and restricted views to the users. As view definitions to comply with regulations became quite complex for accommodating all the restrictions in one view, explicit constructs for specifying privacy policies were introduced for complying with medical regulations like HIPAA (Health Insurance Portability and Accountability Act) from USA, in relational database systems. These enabled fine grained access control (FGAC) capable of enforcing disclosure control enunciated databases. Application of information flow control that is needed for multi-level security (MLS) databases to preserve privacy among multiple users but have their challenges like new abstractions for managing information flow in a relational database system, handling transactions and integrity constraints without introducing covert channels etc. As the DBs need to work alongside information flow controlled programming languages and operating systems for tracking flows, there is a need to enforce the security policy not only on the DBMS but also on the application platform. Due to the underlying requirement of decentralization, it calls for declassification/endorsement and santization requirements on the DB. In this paper, we shall first review some of the major privacy enhancing techniques used traditionally for DBs including MLS DBs, and then explore application of decentralized information flow control models for realizing information flow secure DBs in a robust manner. Towards the end, we shall also touch upon some of the roles of anonymization and psuedonymization including inference control and differential privacy in realizing privacy in practice.

Keywords Privacy · Security · Information flow · Declassification · Relational database

1 Introduction

Privacy had been mostly a legal term till computing and communication became ubiquitous with daily life of citizens. One of the most influential essays by Samuel Warren and Louis Brandeis[1] in the history of American law and is widely regarded as the first publication in the United States to advocate a right to privacy, articulating that right primarily as a “right to be let alone”. Alan Westin’s[2] “Privacy and Freedom” is a detailed and comprehensive evaluation of the conflict between privacy and surveillance in modern society and can be summarized by “right

✉ R.K. Shyamasundar
shyamasundar@gmail.com

Extended author information available on the last page of the article.

[1]“The Right to Privacy” (4 Harvard L.R. 193 (Dec. 15, 1890)) is a law review article written by Samuel Warren and Louis Brandeis, and published in the 1890 Harvard Law Review.

[2]Alan F. Westin, Privacy And Freedom, 25 Wash. & Lee L. Rev. 166 (1968),

to control, edit, manage, and delete information about themselves and decide when, how, and to what extent information is communicated to others”. As seen in recent days, using technological services like Internet, online social networks, mobile phones, databases, etc., has lead to sacrifice of privacy either directly or indirectly. One of the main sources of privacy leaks may be attributed to the large data stores used for a spectrum of reasons including personal benefits of usage that have personal identifiable information. In this paper, we shall take a look at techniques that have been in use to mitigate such privacy leaks in the use of databases.

Traditional database systems are used to store data efficiently and fetch them when required. They are optimized for storage, fast retrieval, and access. Primary concern they address is the performance of the database system. Many organizations use databases for a spectrum of applications. Some of them are storing user credentials (which may be encrypted) for storing a medical history of a patient in a medical database, storing purchase history and billing information in a supermarket database etc. These types of data are sensitive as they carry some kind of personal information either explicitly or implicitly. Preserving privacy has become an important factor in the use of DBs and data, particularly in the context of health data and health regulations like HIPAA, GDPR etc. Privacy is about responsible maintenance of private information which is hard to define[3] and hence, for preserving privacy, using laws of the nations has become necessary. In view of this, initially security checks on the DB had been done at coarse level. With the growth of complexity, accesses were limited to restricted view of DBs keeping in view the privacy requirements. When, views became too complex due to varieties of regulations on the use of data, specification of privacy policies (Rakesh et al., 2002) were deployed with database systems. These privacy policy specification techniques introduced the notion purpose-based policies. Such approaches captured the following privacy interpretation: privacy is the ability of an individual or a group to seclude themselves, or information about themselves, and thereby express themselves selectively. Privacy is about: what information goes where? Security is about protection against unauthorized access. Security helps enforce privacy policies. They can be at odds with each other. For instance, invasive screening at the airports is done to make us more “secure” against terrorism than otherwise. Thus, discussions about whether *privacy is a security property* is beyond the scope of this paper. For some more interesting discussions on privacy, the reader is referred to (Patil & Shyamasundar, 2017). In the sequel, we interpret privacy as responsible maintenance of personal information and that realizing privacy requires realization of security of data in many ways.

With the proliferation of multiple users and concurrent usage, limitations of the traditional discretionary access control (DAC) to realize privacy was realized. Discretionary access policy only restricts the unauthorized access to information provided by the system. For example, access control list of any system only defines the authorized access of data. But discretionary policies do not define how information would be used after proper accessing of that information. Thus, information flow control checks as to how information is treated after accessed by an authorized user became important leading to mandatory access control (MAC). MAC is a method of restricting unauthorized users from accessing objects that contain some sensitive information. Incorporation of such requirements lead to multi-level database system, where data is defined as an object having security labels and each user is defined as a subject having a clearance label. Users who need to access data should have the appropriate security classification level. With the increase in security concerns, multilevel security or multiple levels of security (MLS) denoting processing of information with different security levels, permitting access to users with different security clearances along with principles like *need-to-know* etc., are becoming popular to prevent users from obtaining access to information for which they lack authorization. Further, they can enforce information flow control in the system in a fine granularity such as an attribute or a tuple thus leading to the realization of FGAC. MLS systems need to overcome the covert channel problem: the possibility that a lower level user can predict some unauthorized information from a higher security level or there is a conflicting requirement of data integrity/confidentiality. Such an approach was initiated through the SeaView model (Denning et al., 1988). This work has lead to a significant amount of research efforts, some of the representative works can be found in Jajodia & Sandhu (1991), Smith & Winslett (1992).

With decentralized systems becoming dominant, wherein there is no single central authority controlling flow of information, applying information flow control on DBs raises many challenges like:

- The classical relational query model does not provide a good basis for reasoning about flows of sensitive information; for instance, a query for records about hospital patients who do not have cancer can plausibly reveal indirectly which patients do have cancer.
- The DBMS must provide ways to manage the information flows that arise through mechanisms such as complex queries, stored procedures, and views.

[3]For notions like inversion privacy, the reader is referred to Gurevich et al. (2016).

– Important DB features such as transactions and constraints can lead to information leaks if proper precautions are not taken.

This has lead to research efforts in the application of decentralized information flow control (Myers & Barbara, 1997) to secure MLS databases. IFDB (Schultz, 2013) denoting a DBMS that secures databases by using decentralized information flow control (DIFC) was the first to apply DIFC (Myers & Barbara, 1997) for building a comprehensive model for securing MLS databases. It realizes an interpretation of labelled DB by:

– extending the earlier work on MLS databases, such as SeaView (Denning et al., 1988), to support DIFC providing a practical way to do relational queries while respecting information flow rules.
– introducing declassifying views and stored authority closures to manage information flows in the database,
– makes transactions and constraints safe, through the introduction of concepts such as transaction commit labels and DECLASSIFYING clauses.

One of the major drawbacks of IFDB is that it does not have a robust declassification or sanitization operation due to the underlying DIFC (Myers & Barbara, 1997) model that uses discretionary declassification (or endorsement) rule. That is, any user can declassify data to any other user leading to information leakage, if the declassification is not done carefully; thus, the security depends on the trust of the agents or the application is to be trusted to work in a trusted environment. On the other hand, in the RWFM security model (Narendrakumar & Shyamasundar, 2014), the declassification is not purely discretionary; an owner, S, can declassify an information, say D, to subject S1 only if S1 has influenced D or participated in computing D and hence, information or data cannot be declassified to a randomly selected user. Further, RWFM allows a clean interpretation of labelled data as the label that is a triple of the form (owner, set of readers, set of writers/influencers) that explicitly provides almost a complete information upto that point about the read/write accesses and the ownership. Thus, using RWFM on MLS DB leads to information flow secure DB in a robust manner. Towards the end, we shall also touch upon some of the roles of anonymization and psuedonymization including inference control and differential privacy in realizing privacy in practice.

Rest of the paper is organized as follows: Section 2 introduces some of the DB concepts required for our exposition and also a brief overview of the RWFM model. This is followed by a discussion on traditional approaches for preserving privacy in DBs such as classic coarse-grained access, view-based security and purpose-based privacy to preserve privacy in DBs in Section 3. Section 5 describes privacy preserving approaches for MLS followed by the application of RWFM to realize information-flow secure DBs in PostgreSQL in Section 6. A case study of the application of the information-flow secure PostgreSQL, called SecPostgreSQL, on HotCRP is discussed in Section 7. In Section 8, we discuss the use of anonymization and psuedo-anonymization in realizing privacy in practice. The paper concludes with a discussion in Section 9.

2 Background

In this section, we provide a brief overview of terminology used in databases including multi-level relations, and an overview of the information flow model RWFM.

2.1 Data Representation in the Relational Data Model

In this section, we describe a few definitions that shall clarify notions of coarse-grained to fine-grained security. In the relational data model, data is represented as sets of n-tuples in the following way:

1. A field or column is an atomic unit of data, often termed attribute over a domain. For instance, Rama is an attribute over the domain of names, while 14/4/1981 is an attribute over the domain of dates. A special value called, *null value*, is included to denote information that is missing, as it may be unknown or not-visible etc.
2. Tuples, or rows, are ordered collections of attributes representing relationships among attributes. For instance, 14/4/1981 could be the date of birth of Rama.
3. Relation is an unordered set of tuples. Each relation has a schema defining a name and domain for each attribute in the relation; all tuples follow the schema. For example, the schema for a relation having names and dates of birth (dobs) could be (name : string, dob : date).
4. Relations are nothing but mathematical objects on sets, and thus do not allow duplication.
5. Each relation has a primary key, which is a collection of attributes that uniquely identifies any tuple in the relation. The analogous concept of tables in most database systems optionally allows for duplicates, in which case there may be no primary key corresponding to the bag semantics.

2.2 Multi-Level Relations

The basics of multi-level relations are given below.

1. When access class is assigned to a tuple, it is called a multilevel tuple or MTUPLE.
2. Multilevel relations are sets of multilevel tuples. That is, $MRELATION \subset 2^{MTUPLE}$
3. The relation value corresponding to the empty tuple set ϕ is called the null relation.

2.3 Readers-Writers Flow Model (RWFM): An Overview

To overcome the drawback of traditional MAC models several new information flow models have been proposed. Readers Writers Flow Model (Narendrakumar & Shyamasundar, 2014) is one such model. It is a powerful lattice-based information flow model based on Denning's model (Denning, 1976). It can be used to provide both confidentiality and integrity. It supports dynamic labeling and downgrading operations. Also, it can capture several well-known models like BLP, Biba, Chinese Wall, Brewer & Nash model etc. Its labeling and access rules are described below. The label of subjects and objects in RWFM is a triple: (Owner/Authority, {Set of Readers}, {Set of Writers/Influencer}). The first component specifies the owner of the information, the second component denotes the subjects who can read the information, and the third component represents the subjects who have contributed/influenced the information so far.

Readers-Writers Flow Model (RWFM): It is an eight tuple $(S, O, SC, \leqslant, \oplus, \otimes, \top, \bot)$, where S and O are sets of subjects and objects in the information system, $SC = S \times 2^S \times 2^S$ is the set of labels, $\leqslant = (-, \supseteq, \subseteq)$ is the permissible flows ordering, $\oplus = (-, \cap, \cup)$ and $\otimes = (-, \cup, \cap)$ are the join and meet operators respectively, and $\top = (-, \emptyset, S)$ and $\bot = (-, S, \emptyset)$ are respectively the maximum and minimum elements in the lattice.

Consider the labelling function λ that maps a subject/object to its respective security label from the lattice, and defined as, $\lambda : S \cup O \rightarrow S \times 2^S \times 2^S$. Let $R_\lambda(e)$ and $W_\lambda(e)$ be the projection of the label $(\lambda(e))$ to the second and third component respectively.

RWFM follows a floating-label approach for subjects, with labels $(s, S, \{s\})$ and $(s, \{s\}, S)$ denoting the "**default label**" - the label below which a subject cannot write, and "**clearance**" - the label above which a subject cannot read, for a subject s respectively. Object labels are fixed and are initially provided by the desired policy. The state transition rules of RWFM (Narendrakumar & Shyamasundar, 2014) are summarized below:

READ Rule

"*Subject s with label* (s_1, R_1, W_1) *requests read access to an object o with label* (s_2, R_2, W_2).
If $(s \in R_2)$ then
change the label of s to $(s_1, R_1 \cap R_2, W_1 \cup W_2)$
ALLOW
Else
DENY"

WRITE Rule

"*Subject s with label* (s_1, R_1, W_1) *requests write access to an object o with label* (s_2, R_2, W_2).
If $(s \in W_2 \wedge R_1 \supseteq R_2 \wedge W_1 \subseteq W_2)$ then
ALLOW
Else
DENY"

CREATE Rule

"*Subject s labelled* (s_1, R_1, W_1) *requests to create an object o*
Create a new object o, label it by (s_1, R_1, W_1) and add it to the set of objects."

DECLASSIFICATION Rule

"*Subject s with label* (s_1, R_1, W_1) *requests to downgrade an object o from its current label* (s_2, R_2, W_2) *to* (s_3, R_3, W_3).
If $(s \in R_2 \wedge s_1 = s_2 = s_3 \wedge R_1 = R_2 \wedge W_1 = W_2 = W_3 \wedge R_2 \subseteq R_3 \wedge (W_1 = \{s_1\} \vee (R_3 - R_2 \subseteq W_2)))$ then
ALLOW
Else
DENY"

The declassification rule says, *the owner of an object can declassify the content to a subject only if that subject has influenced the information upto this point.*

For a given security lattice, the RWFM state transition rules precisely determine the labels of information at different stages of control flow. Note that, in RWFM, as the information moves up in the lattice, the readers set continues to reduce, i.e., becomes more restrictive whereas, writers set grows as more subjects contribute to the information.

In Narendrakumar & Shyamasundar (2017), it is shown that RWFM is complete with respect to Denning's lattice flow (Denning, 1976). The reader is referred to Narendrakumar & Shyamasundar (2017), Narendra Kumar & RKS (2016) for discussions on its characteristic properties and comparisons with other DIFC models. Some of the major advantages of RWFM are:

– Security policies are enforced stringently as every data access request always passes through the reference monitor. Even the data after processing is assigned labels, which prevents information leakage indirectly using covert channels.

- Information flow rules take into account the current label of the subject which has been influenced by all the information that has been read by the subject so far.
- Confidentiality of data takes into account both the readers set and writers set, unlike other security models.
- Labels are defined succinctly and are easy to understand. They also provide almost a complete view of information flow.
- RWFM has also been applied for information flow-security of programming languages (Ghosal et al., 2019), operating systems (Shyamasundar et al., 2018; Vyas et al., 2021), Hadoop (Shyamasundar et al., 2019) as well as document security (Vamshi et al., 2017).
- SELinux, one of the widely used operating systems uses a limited MAC policy. One of the issue is the consistency of the user defined policy with reference to information flow. Radhika et al. (Radhika et al., 2020) describe a tool based on RWFM that is capable of finding possible information leaks in SELinux policies automatically.

3 Security in Traditional Databases

In this section, we shall briefly overview the approach of realizing security in DBs using authorization control or query rewriting technique, view-based security and privacy-policy specific enforcement.

3.1 Authorization

In the relational model a database is a finite set of named relations. The data in a relation can be represented either physically as stored data or logically as a derivation rule that defines how the data is computed. The reason for modeling derived data is that it allows inter-dependencies and inference rules among stored and derived data to be expressed within a single framework. Thus, security is realized through valid authorization rules and query rewriting techniques. For instance, it could be done through SQL GRANT or REVOKE statements as illustrated below:

```
GRANT privilege name ON object name
      TO user name [WITH GRANT OPTION];
REVOKE privilege name ON object name
 FROM user name;
```

Through such rewrites, one can have Table/column-level authorizations and it does not support access pattern restrictions. Some of the advantages of the simplistic approach are: (i) no need for query modification, (ii) easy to administer, (iii) authorization-transparent querying, and (iv) security within the database holds.

This is one of the earliest techniques that has been in use and the reader is referred to Silberschatz et al. (2013) for further details.

3.2 View-Based Security

To allow fine-grained access control, view-based security (Rizvi et al., 2004) is being used as illustrated below:

```
CREATE VIEW  ShawnGrades AS
SELECT  * FROM Grades
WHERE name = 'Shawn'
```

View-based security is capable of arbitrarily fine grain policy specification and preserves advantages (i)-(iv) given above. Administering per user views becomes difficult; one need to resort to parameterized views as illustrated below:

```
CREATE VIEW costudent-grades AS
SELECT student-id, course, grade
FROM  grades, registered
WHERE grades.course=registered.course and
 registered.student-id = $user-id
```

Some of the disadvantages of view-based security are (i) Applications may need to generate different queries for different users, (ii) Programmers need to know database schema and views become complex. There are also approaches like Oracle's private databases (Rizvi et al., 2004) which overcome some of the issues but also add issues due to modification of queries.

One may ask questions like:

1. Is policy P1 compliant wrt policy P2?
2. Is query Q1 consistent with policy P1?

At a broader level such questions need to consider the underlying information flow as well as inference control in statistical databases aggregates, sanitized information etc., (Denning, 1982). Information flow could be analysed by transforming the policies into RWFM policies. For instance, if a database administrator offers another view on the DB, one needs to check whether all the views currently active together are consistent with the policy. Some of these aspects are discussed informally in Section 4.3.

4 Enforcing Privacy Policies on Databases

In this section, we shall briefly discuss an approach envisaged in Rakesh et al. (2002) for enforcing privacy policies on Hippocratic databases automatically.

In general, purpose-based privacy refers to giving access to a person by the owner of data based on the purpose defined with which the person wants to get access to the data. In Hippocratic Databases, each data owner has control on his/her own data. The owner can specify possible accesses to data based on a table called privacy policy

Table 1 An illustrative schema

Table name	Attributes
Customer	Customer-Id, Name, Shipping Address, e-mail, Credit-card info
Order	Customer-Id , Transaction-Id, product info, Status

Table 2 Privacy metadata schema

Table name	Attributes
Privacy-policies	Customer-Id, Purpose, table, Attribute, external-recipients, retention
Privacy-authorization	Customer-Id, Purpose , table, attribute, authorized-users

table. Each data owner also specifies for what purpose a user can access the data. For understanding purpose, let us consider the DB schema shown in Table 1. Assume that the database is maintained by a company that sells products. The company can use the data collected for various purposes as below:

- Company can give e-mail address to third party for promotion of their products via mailing.
- Using shipping address, company can make a statistical analysis of the products being sold at a particular region. In one place a particular product may be sold more than some other place. Such data would be very useful in maximizing the profit by giving relevant product details to each region for new users.
- If the DB also gives the name of the person to others, it is as good as giving a sensitive information as it can reveal what a particular person is buying online. Many customers would not like to disclose their identity.

It is possible that some customers may like their data to be shared with others while some may not. Thus, to maintain trust with the customers the company must ensure that they use the data of those customers who have given permission to use their data. Let us say that the company has many divisions like:

- **Customer-Service** : Their job is to help customers in need – thus needing the name, e-mail and address of the customers for servicing.
- **Shipping** : The job is to deliver the product to the customer for which they need at least the delivery address of the customer. Note that the Shipping department can be some other third party.
- **Charge** : They have the job to ensure that the product payment has been done by the customer for which they need credit-card information.
- **Purchase-Circle** : They make a statistical analysis of the sale of products. They find regions where a particular product is popular and provide recommendation of that product to new users in that particular region.

4.1 Purpose Based Privacy Control

Suppose there is a customer Bob who does not want any of his data to be retained by the company once his purchase is completed, and there is another customer Alice who is happy with the fact that the company can trace her purchase history to refer new and relevant products to her. But she does not want to share her information for purchase circles. Thus, in the case of Alice, we can see that she is willing to share data to the company but for specific purposes. As already said in order to ensure trust of Alice, the company has to follow her explicit intent, leading to *Purpose Based Privacy*. For the above example, let us define relevant purposes like (i) PURCHASE, (ii) REGISTRATION, (iii) RECOMMENDATION, and (iv) PURCHASE-CIRCLES. Now, we can easily say that Alice wants to share data for the purpose PURCHASE and RECOMMENDATION but not for PURCHASE-CIRCLE.

Rakesh et al. (2002) realize it through *privacy metadata table* as shown in Table 2. The metadata schema is split into:

1. **Privacy-Policies Table** (Table 3) stores the list of external users who can get a particular data of a particular customer. The list is stored in the triplet:(table,attribute,purpose).

   Thus, for each table and each attribute in the table, the purpose for which that attribute is permitted by the customer is specified; it could be even to an outside company. A typical privacy policy is shown in Table 3.
2. **Privacy-Authorization Table** shown in Table 4, stores the list of users (we can treat them as departments or subjects) within the company who can get access to a particular data of a particular customer.

From the privacy authorization table for an user, say Alice, it will be clear as to who are the subjects that can access the respective attributes of Alice, for the specified purpose. In the DB shown in Table 5, the data owners are Alex, Jim, Olly and Brian. Each of the data owners will have their own privacy authorization table defining access of their attributes to subjects for a purpose. Privacy authorization for Alex is shown in Table 6.

Table 3 A privacy policy

Purpose	Table	Attribute	External-recipient	Retention
PUECHASE	Customer	Name	deliver-company, credit-card company	6 months

Table 4 Privacy authorization

Purpose	Table	Attribute	External-recipient
PURCHASE	Customer	Name	deliver-company, credit-card company

Suppose user Shipping wants to get access to name attribute for a Cable customer. The query will be

```
Select name from customer for purpose
  PURCHASE;
```

In the privacy authorization table of Alex, one need to check whether Alex has given access to shipping for purpose PURCHASE. As he has given access (can be seen clearly from the privacy authorization table), shipping will get access to the attribute name. Now suppose OLAP wants to get access to email attribute for purpose PURCHASE-CIRCLE, then he can query

```
Select email from customer for purpose
  PURCHASE-CIRCLE;
```

In the privacy authorization table, Alex has not given access to attribute email for the purpose PURCHASE-CIRCLE. Hence, OLAP will not get access to the data. In this way privacy is controlled using the privacy authorization tables.

4.2 Policy Enforcement in Hippocratic Databases

A policy is a rule to give access (read/write) to a database element of a table to a person (subject in this case). Read access can be given to a subject by giving access to select statements in SQL. Write access can be given to a subject by giving access to insert, update and delete statements.

A general structure of a policy specification (Rakesh et al., 2002) is given below with the standard interpretation:

```
create restriction restriction-name
on table-x
for auth-name-1 [ except auth-name-2]
( ( (to columns column-name-list)
j (to rows [ where search-condition ] )
j (to cells (column-name-list [ where
  search-condition ] )+ ) )
[ for purpose purpose-list ]
[ for recipient recipient-list ]
)+
restricting access to (all j (select j
  delete j insert j update)+ )
```

Table 5 An example DB

Name	Shipping-address	e-mail	Credit-card info
Alex	Washington	alex@gmail.com	7895-4578-4512-7896
Jim	Texas	jim@hotmail.com	8974-6524-1452-3254
Olly	Boston	olly@ymail.com	7896-2541-3256-2541
Brian	New York	brain@gmail.com	2145-9874-6589-4785

Table 6 Privacy authorization table for Alex

Purpose	Attribute	Authorized users
PURCHASE	name	shipping,charge, customer-service
PURCHASE	shipping-address	shipping,customer-service
PURCHASE	e-mail	customer-service
PURCHASE	credit-card info	charge
REGISTRATION	name	register,customer-service
REGISTRATION	shipping-address	
REGISTRATION	e-mail	register,customer-service
PURCHASE-CIRCLE	shipping-address	olap

Let us consider the table schema for Customer given before and a privacy authorization table of subject (data-owner) Brian as shown in Table 7.

Let us apply the following policy to Table 7:

```
Create restriction r1
on customer
for user customer-service
to cells name
for purpose purchase
restricting access to select
```

Privacy authorization table after implementation of the policy is shown in Table 8. Note that after implementing the policy, customer-service is added in the authorized users column where purpose is PURCHASE and attribute is name.

4.3 Using RWFM Model for Compliance Checking of Policies

There can be situations where data-owners may have several questions like:

1. A given user may want to find the compliance of two policies?

Table 7 Authorization table of Brian

Purpose	Attribute	Authorized users
purchase	name	shipping
purchase	shipping-address	shipping
purchase	e-mail	shipping,customer-service
purchase	credit-card info	charge
registration	name	register,customer-service
registration	e-mail	register,customer-service

Table 8 Resulting authorization table

Purpose	Attribute	Authorized users
purchase	name	shipping,customer-service
purchase	shipping-address	shipping
purchase	e-mail	shipping,customer-service
purchase	credit-card info	charge
registration	name	register,customer-service
registration	e-mail	register,customer-service

2. If the users have conflict-of-interest, can one find out whether the policy of one user contradicts that of the other.
3. In the context of multi-level security, it would be necessary to see whether there is a possibility of leak of data.

For answering such questions, one needs to exhaustively check the possibilities. In the following, we illustrate informally how such compliance checking can be done through RWFM modelling (Aniket, 2018) of the privacy policies of Hippocratic DB. In the RWFM model each of the data as well as the users will have RWFM labels on them. For illustration, let us consider the ,"select" operation on DB. Let us consider the reader sets of objects and subjects with respect to this operation. As data is accessed in the table based on the purpose, let us see whether keeping PURPOSE in the reader set is consistent with the original model or not. Consider the privacy authorization shown in Table 9:

We can observe that the reader-set for attributes is given by:

– **Name:** purchase, registration
– **Shipping-address:** purchase, registration, purchase-circle
– **E-mail:** purchase, registration
– **Credit-card info:** purchase

and the reader set for subjects is given by:

– **Shipping:** purchase
– **Charge:** purchase
– **Customer-service :** purchase, registration
– **Olap:** purchase-circle

Since subject **charge** will get access to all attributes, PURPOSE alone cannot form the reader-set; we need to specify which users are given the authority to access the data along with the purpose. Keeping this in view, we can refine the sets as follows:

1. Reader-set for attributes:
   – **Name:** {purchase:shipping, charge, customer-service}, {registration:register, customer-service}
   – **Shipping-address:** {purchase:shipping},{registration:register},{purchase-circle:olap}
   – **E-mail:** {purchase:shipping, customer-service},{ registration:register,c ustomer-service}
   – **Credit-card info:** {purchase:charge}
2. The subjects list would be: Shipping-purchase, Charge-purchase, Customer-service-purchase, Customer-service-registration, Olap-purchase_circle.

With the above sets on hand, we can consider accesses authorized in Hippocratic DB and the corresponding RWFM abstraction. In the RWFM model, the reader set has both the PURPOSE and authorization together and thus, we can check directly whether a particular user has access to data or not. Subjects will also have PURPOSE clubbed with the subject name; for example customer-service-purchase where customer-service is the subject name and PURCHASE is essentially the PURPOSE. The reason for

Table 9 An example authorization table

Purpose	Attribute	Authorized users
purchase	name	shipping,charge,customer-service
purchase	shipping-address	shipping
purchase	e-mail	shipping,customer-service
purchase	credit-card info	charge
registration	name	register,customer-service
registration	e-mail	register,customer-service
purchase-circle	shipping-address	olap

adding ,"PURPOSE" is that there can be many *purposes* for which access may not be granted. For example, in the reader-set of attributes mentioned above subject, OLAP cannot access any attribute for purpose PURCHASE. So a new subject is created as olap-purchase. Now according to RWFM access rules, olap-purchase when defined as a subject, cannot access any attribute. Thus, in the new model which is a conversion of the purpose-based model based on privacy authorization table to RWFM model, all accesses remain consistent.

5 Multi Level Security

The notion of multilevel security arises (Denning et al., 1988) when a computer system contains information with a variety of classifications and has users who may not have cleared for the highest classification of data contained in the system.

A security access class consists of a hierarchical authorization/sensitivity level, e.g.,
$TOP - SECRET > SECRET > CONFIDENTIAL > UNCLASSIFIED$
and a set of non-hierarchical categories. In order for a user to be granted access to information, the user must be cleared for the sensitivity level as well as for each of the categories in the information's access class. The sensitivity levels are linearly ordered. The categories do not have such a linear ordering. However, the set of access classes ($< sensitivitylevel, categoryset >$ pairs) is partially ordered and forms a lattice(Denning, 1976). The partial ordering relation is called the dominance relation. Access class A dominates access B if the sensitivity level of A is greater or equal to the sensitivity level of B and the security categories of A include all those of B.

The MLS model defines the mandatory security policy using MAC (Mandatory Access Control) over its subjects and objects using the Bell La Padula (BLP) security model. The MAC model assigns two access classes to each subject S: read-class(S) and write-class(S) (these two classes are equal for untrusted subjects), where read-class(S) $\geq$ write-class(S). The access requirements (Denning, 1976) are formalized by the following two rules:

1. A subject S can read data of access class c only if read-class(S) $\geq$ c, and
2. A subject S can write data of access class c only if write-class(S) $\leq$ c.

An MLS model is a finite state transition system that satisfies the requirements for read and write as above that executes operations of the form: $op(s_1, S, x_1, ..., x_n \rightarrow s_2)$ where S is a subject and $x_1, ...x_n$ are other parameters. Each abstract command represents an atomic action that causes a state transition from state s_1 to state s_2.

A state is secure[4] if and only if it satisfies all state transition system properties. A command $op(s_1, S, x_1, ..., x_n \rightarrow s_2)$ is secure if and only if, for all subjects S and all parameters satisfy all the transitions properties (also type properties), s_2 is secure whenever state s_1 is secure. Other properties like reachability etc., follow on the lines of BLP.

5.1 SeaView Model for MLS

SeaView model (Denning et al., 1988) initiated efforts in building secure MLS databases. The SeaView model consists of two layers:

1. An inner layer called the MAC layer defining the mandatory security policy – this is essentially the security kernel or the reference monitor. This defines essentially the underlying security model, Bell–LaPadula (BLP) model, and has no component specific to databases and thus, builds a relationship between the security model and the database system.
2. An outer layer called the Trusted Computing Base (TCB) layer defining the DAC and other supporting policies. It specifies components of a multi-level secure relational database system defining the multi-level relational data model, views, integrity/classification constraints, transactions and discretionary authorizations. This layer is modeled on top of the MAC layer.

The model is used for databases that contain data belonging to different classes of sensitivities and not all the users have the highest clearance to access the entire database. It essentially follows lattice information flow model (Denning, 1976) (BLP is contained in this model). Thus, the SeaView security model prevents information leakage from covert channels. Within the lattice, no information can flow from a higher level of security to lower level and also, information can only be written to lower levels. The concept of reference monitor is introduced, which actually guards all access to the objects by subjects. All the access requests for objects pass through the reference monitor, which checks whether it conforms to the information flow rules. All the security-critical operations must pass through reference monitor. The design approach of SeaView model implements multilevel relations using views at different security levels. Subjects interact with the relations using views, which fetches data from a single level relation stored physically in a file. This transition from single level to multi-level relation is transparent to users. Reference monitor lies just above the single level relation. Since for all the requests data is ultimately fetched from single level relation, which

[4]Note that it also has to use the *tranquility principle* (Denning, 1976).

passes through the reference monitor in order to perform label comparison to determine if access should be granted or not. The model assumes a central labelling authority and faces issues like repeated join, spurious tuples, high cost due to left outer join etc. due to decomposition and the underlying recovery algorithm. Jajodia & Sandhu (1991) provide a refinement of the SeaView model and provide an improvement over its recovery algorithm. But the model is semantically ambiguous and incomplete in terms of operations. Smith & Winslett (1992) model provides security labels to primary key attributes and the tuples. It does not provide security at the level of attributes.

5.2 Multi-Level Database Security via Views

Denning et al. (1987) extend the notion of views in traditional DBs to MLS DBs wherein all data entering the database are labeled according to views called classification constraints, which specify access classes for related data. It introduces class of views with aggregation constraints that restrict access to aggregates of information. Thus, all accesses are confined to a third set of views called access views. For purposes of illustration, consider the flight data schema relations given below:

```
   ITEM(ITEM#, ITEMNAME, WEIGHT)
   FLIGHTS(FLIGHT#, DATE, DEST, WEIGHT)
   PAYLOAD(FLIGHT#, ITEM#, QTY, WEIGHT)
```

The set of schemas defining the relations in the database is itself represented as a relation: RELATIONS (RELNAME, ATTRNAME), which contains an entry for each attribute of each relation.

In a MLS DB, let us suppose that PAYLOAD.WEIGHT is a derived data, defined in terms of the join of the two relations as given below:

```
  PAYLOAD.WEIGHT:=
  ITEM.WEIGHT * PAYLOAD.QTY
  where ITEM.ITEM# = PAYLOAD.ITEM#
```

Issues of classifications in MLS should become clear with the following examples (Denning et al., 1987):

1. Let us consider the relationship among ITEM. WEIGHT, PAYLOAD.QTY, and PAYLOAD.WEIGHT places constraints on how the data are classified. If PAYLOAD.WEIGHT is regarded as TOP-SECRET, then it would not be secure to classify both ITEM.WEIGHT and PAYLOAD.QTY as SECRET since a user with a SECRET clearance could access these attributes and deduce PAYLOAD.WEIGHT. More generally, it would not be secure to classify any two of the attributes lower than the third, and this is true regardless of whether PAYLOAD.WEIGHT is actually stored in the database.
2. Let us consider an example where we sum the elements in records in the DB. If the individual elements are all SECRET, then is it fine to release the sum at a lower level, say CONFIDENTIAL, on the grounds that it sufficiently sanitizes the individual elements. Again, the decision depends only on the inferences that can be drawn from it, and not on whether it is actually stored in the database.

As the DB need not represent the inference closure, the user could take a appropriate views via the query language for enforcing the required security/privacy as briefed in the following section.

5.2.1 Views on ML DB

A view (Denning et al., 1987) is a mapping (multivalued function) from a database to a relation (or set). For security purposes, we need to consider the subset of data in the database that is used to compute the resulting relation.

Example 1: PAYLOAD.WEIGHT definition in terms of a view is given below:

```
   view PAYLOAD.WEIGHT
   ITEM.WEIGHT * PAYLOAD.QTY
   where ITEM.ITEM# = PAYLOAD.ITEM#
```

All elements associated with the attributes ITEM.ITEM#, ITEM.WEIGHT, PAYLOAD.ITEM#, and PAYLOAD.QTY form the source and the target is the elements associated with PAYLOAD.WEIGHT

Example 2: A parameterized view, is given below;

```
   view HEAVIER-THAN(x) := ITEM.all
   where ITEM.WEIGHT > x
```

5.2.2 Labelling Data with Constraints

A classification constraint specifies an access class to be assigned to all data in its target. The access class can be expressed as a constant or as a formula over the access classes of the data in the source using the lattice operators lub (least upper bound) and glb (greatest lower bound).

Example 3: A type dependent classification constraints example is given below:

```
   classification constraints on PAYLOAD
   FLIGHT#, ITEM#, QTY
   class SECRET
   WEIGHT class TOP-SECRET
```

Example 4: A value dependent classification constraint is given below:

```
   classification constraints on FLIGHTS
   ALL
  class TOP-SECRET where FLIGHTS.DEST = Iran
  class SECRET where else
```

It is also possible to define further derivation rules to specify the computation of data through views (Denning et al., 1987) .

Sanitization Rule: A **santization rule**, is also a derivation rule to sanitize its source so that the access class of the target can be strictly dominated by the least upper bound (lub) of the source access classes. It must also be noted that property realization actually relies on trusted processes that reside in layers above the reference monitor.

An example is given below:

```
sanitization rule on FLIGHTS
 FLIGHTS.WEIGHT := sum(PAYLOAD.WEIGHT)
 where count(PAYLOAD.ITEM#) > 1 0
 and FLIGHTS.FLIGHT# = PAYLOAD.FLIGHT#
 class G {FLIGHTS.FLIGHT#.class, PAYLOAD.
  FLIGHT#.class, SECRET}
```

It sanitizes values of PAYLOAD.WEIGHT, thereby excluding the attribute PAYLOAD.WEIGHT from the access class specification. The access class specification includes all other attributes in the source (even though the total weight discloses no information about flight numbers), and also states that the resulting access class will be at least SECRET. The clause

$$count(PAYLOAD.WEIGHT) > 10$$

ensures that a minimum number of items is on board each flight so that no single weight can be inferred from the sum.

Sanitization rules are required for most property associations and quantitative associations. With respect to property associations, note that simply assigning a higher access class to one or more of the attributes forming the aggregate often does not address the problem. For further refinements of aggregations, location sanitization etc., the reader is referred to Denning et al. (1987). It may be noted that views provide a good class-based security for MLS data. It must be noted that the set of classification constraints must be complete and consistent. A set of classification constraints is said to be complete if an access class is defined for each element; it is consistent if no two constraints, both of which must be satisfied simultaneously, define conflicting classes.

6 Realizing Information Flow Secure DBs using RWFM

In this section, we show how MLS DBs can be made information flow secure using RWFM. Before we discuss the integration of RWFM with a relational DB, let us briefly look at possible ways in which relational databases are labelled as that is basic to decentralized information flow control.

6.1 Labelling Relational Databases

Labelling of information provide a very flexible way of controlling sensitive information. Some of the notable characteristics are given below:

1. A label provides users a sort of constraints about the operations that can be performed on the data.
2. A label provides the sensitivity of data/object item. An object label shows the level of object's sensitivity and provides a criteria that a user must meet to gain access to that object.
3. Database labelling can be done in many ways like

   a. **Table-Based Label System:** Here, labels are provided for each of the tables and hence, corresponds to all the tuples in that table. It does not fulfill the security needs because one table stores the data on behalf of many users and in multilevel database system, every user has different security requirements. Therefore table based label system requires different tables for different user's data.

   b. **Tuple-based label system**: Here, each tuple is labelled; note that the data in each tuple is related to a single entity. Per-tuple labels represent a good trade-off between the ability to label data at a fine granularity and the space overhead associated with the labels. A typical labelling is shown in Table 10. Therefore labels of each attribute value is not clearly provided in this case.

   c. **Cell-Based Label System:** It provides labels on each data of tables in database. Each data is considered as individual object. A typical labelling is shown in Table 11

4. The labeling granularity denotes the level at which information flow is controlled. That is, information flow in table-, tuple-, or cell-based system is defined the perimeter of the relation, record or cell respectively.
5. For secure database systems, there is a requirement to control access at a higher granularity level and we need to present a view to the user only after applying the filtering logic on the object as well as sensitivity level of the user. Thus, while a user can access the table, each user sees the data that is allowed according to the labels on the underlying entities.

Table 10 Labelled patient table: tuple label

Patent-name	DOB	Problem	Condition	Label
John	5/12/65	Cancer	Good	{John-Label}
Allison	4/3/1970	Trauma	Serious	{Allison-Label}
Kathy	3/11/1951	Myopia	Critical	{Kathy-Label}

Table 11 Labelled patient table: cell-label

Patent-name	P-label	DOB	dob-label	Problem	prob-label	Condition	Cond-label
John	S,{P1,D1}	5/12/65	S,{P1,D1}	Cancer	S,{P1,D1}	Good	{P1,D1}{D1}
Allison	S,{P2,D2}	4/3/1970	S,{P2,D2}	Trauma	S,{P2,D2}	Serious	{P2,D2}{D2}
Kathy	S,{P3,D1}	3/11/1951	S,{P3,D1}	Myopia	S,{P3,D1}	Critical	{P3,D1}{D1}

6.2 Realizing Information Flow Security of DB via RWFM

RWFM assumes a cell-based labels of the data in the DB. In a sense, each data cell acts as a separate object in the database. A separate column has been introduced for each attribute for storing these labels in the relation. A label is a triplet containing owner, set of readers and set of writers. All the subjects and objects in the system are assigned labels to depict their level of sensitivity in the lattice. The subject labels are dynamic in nature while the object labels remain static.

As discussed already, security of a DB depends upon the security of the operations on the DB by a subject. Below, we shall describe the semantics of constructs (Arushi, 2016; Chaudhary, 2017) assuming the label of the user who is executing the query. For illustrative purposes, often we use PostgreSQL, as our prototype is built on it.

Data Definition Language (DDL) statements and Data Manipulation Language (DML) statements are transformed to incorporate the label checking before reading or writing data. In DDL, CREATE TABLE statement is modified to support labels per column basis without the user being aware of it. In DML, SELECT, INSERT and UPDATE statements are modified in order to perform label checking before permitting any information flow.

CREATE Table Query This statement creates a new table in the database with the structure defined by the user in the query. When this query is executed by the user, the PostgreSQL creates label columns automatically for each of the user defined columns without requiring user awareness of the addition of the label. The addition of label columns is done at parsing level. The table schema is shown in Table 12.

SELECT Query

1. Select statement is used to retrieve data from table in database. In the output, only those rows must be shown whose readers set contains the user on behalf of whom query is being executed.

Table 12 Table schema of RWFM

$attr_1$	ℓ_attr_1	$attr_2$	ℓ_attr_2	$\cdots$	$attr_n$	ℓ_attr_n

2. The check must also be performed on all the columns present in the where clause condition, if it is also a part of query.
3. To perform the label check, a few changes are needed to be done while creating the query tree and also, when the plan is executed, i.e., when the actual values are fetched from the database (we will not go into details of implementation here).
4. If the current user is present in the readers set of all the label columns, send the tuple to the destination to be shown as output, else ignore the tuple.

INSERT Query Insert statement is used to insert new rows into a table. Insert statement can be considered as creating a new object in database. In insert query, only new data is going to be written in the given table corresponding to a write operation of RWFM. Some of the subtle points are:

1. According to the create rule of RWFM, each new inserted data label is updated with the current subject's label.
2. If the target table has primary key and the key value to be inserted already in the table, then a check needs to be performed to see if the current subject is in the readers set of present label of the key column. If it is not, then the row should be inserted without throwing any duplicate key value error as according to RWFM model, this won't be considered as violation of data integrity constraint as the key along with label becomes unique. Otherwise, PostgreSQL must report an error to the user normally; but this may leak the information of the key being there and based on the security perception either a message is posted or ignored without taking any action.
3. Label values for the columns values to be inserted are initialized with the label of subject in the rule for insert-statement defined. Now, the PostgreSQL looks for the index of primary key, if it exists and if the key value already exists, fetch the whole tuple by de-referencing the record pointer value stored in index. After fetching the row, check if readers set in the label of key column contains the current subject. If yes, then an error message of existence duplicate value to the user normally; however based on the security perception, a message is posted to the user or ignored but no action takes place. Otherwise, insert the row in the table and update the index as well.

UPDATE Query Update statement changes the values of specified columns in all rows that satisfy the given condition. Only columns to be modified need to be mentioned in the SET clause; columns not explicitly modified retain their previous values. Update statement corresponds to, first reading a row from the table and then writing (updating) new data in table. Thus, update query first follows read rule and then write rule of RWFM. According to the RWFM model, the labels of the columns to be updated must have current subject in their writer set and if the query has where-clause, then the labels of all the columns on which the condition is applied in the where-clause must have current subject in their reader set. Before performing the check, labels of all the columns present in update-target-list and where-clause-list should be added to the update-target-list.

Transaction A transactions is a set of multiple SQL queries forming a single logical operation. A transaction either gets executed completely or not executed at all; i.e., it is atomic in nature. Each of the querries executed by PostgreSQL is a part of the transaction. Individual queries create a new session for a transaction and once the execution is done, transaction is committed or aborted. The creation of a new session initializes the subject label with the initial value; i.e., owner is the subject itself, reader set contains all the subjects in the system and writer set contains the subject itself. In the case of multiple statements getting executed as part of a transaction, subject label might climb up in the lattice as it is dynamic and can only be changed if subject reads information. Further, if the subject inserts a tuple in a relation, the label of the attributes of the tuple would reflect the updated label of the subject. That is, subject label has to be maintained in some data structure in main memory. It need not be stored in a persistent storage as once the transaction gets committed or aborted, the subject label will not be valid and need to be initialized again for execution of other queries. When a subject begins a transaction, a flag is set to indicate if the queries are going to be executed as part of the transaction or not. If yes, then the subject label is fetched from the data structure and relevant checks are performed on that label. The semantics of the basic SQL queries remain the same as discussed above; the difference is that the subject label is fetched from the data structure as the label of the subject might change during the course of the transaction. When user executes commit or abort the flag is again reset and the label of the subject becomes invalid.

6.3 Realizing Secure PostgreSQL

A prototype secure PostgreSQL version 9.4.4 has been realized and the prototype is called SecPostgreSQL (Chaudhary, 2017). Both PostgreSQL and SecPostgreSQL have been compared over a number of queries using TPC-H benchmarking tool for measuring the performance overhead. The database schema was used for measuring the performance of the existing PostgreSQL implementation. However, SecPostgreSQL uses a slight variant of table schema as it adds extra columns for each attribute to specify whether it is private or non-private. Thus, the Insert query needs to be modified to accommodate the same. Rest of the queries are used as used in the classic PostgreSQL.

Fig. 1 Performance of SELECT

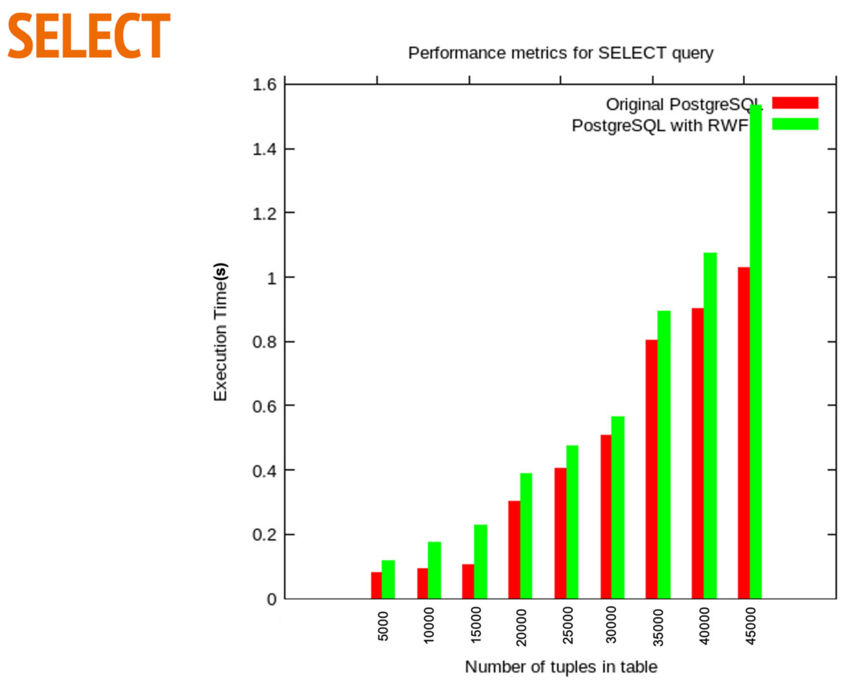
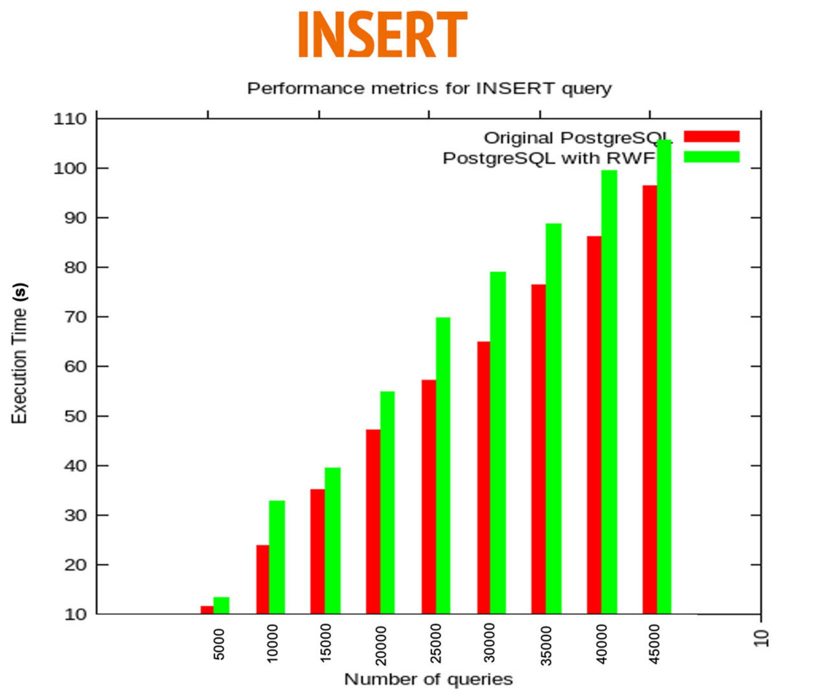


Fig. 2 Performance of INSERT



Benchmarking was done for the general DML queries such as SELECT, INSERT, DELETE and UPDATE. Time taken was plotted for doing the specified operation on a discretely step-wise increasing the number of tuples. SecPostgreSQL demonstrated a slight degradation in performance in each type of query. This can be attributed

Fig. 3 Performance of UPDATE

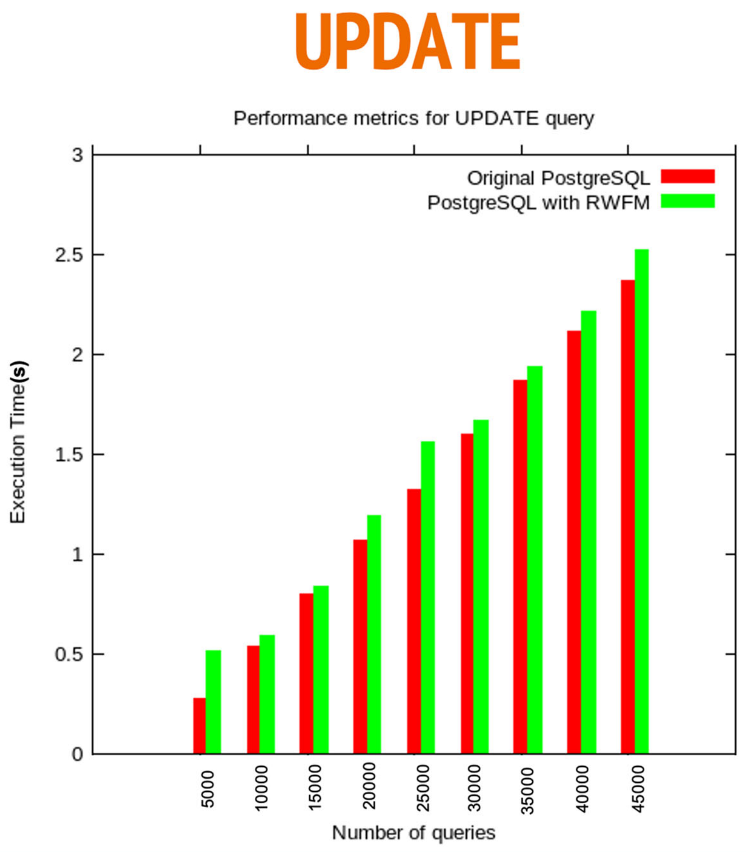
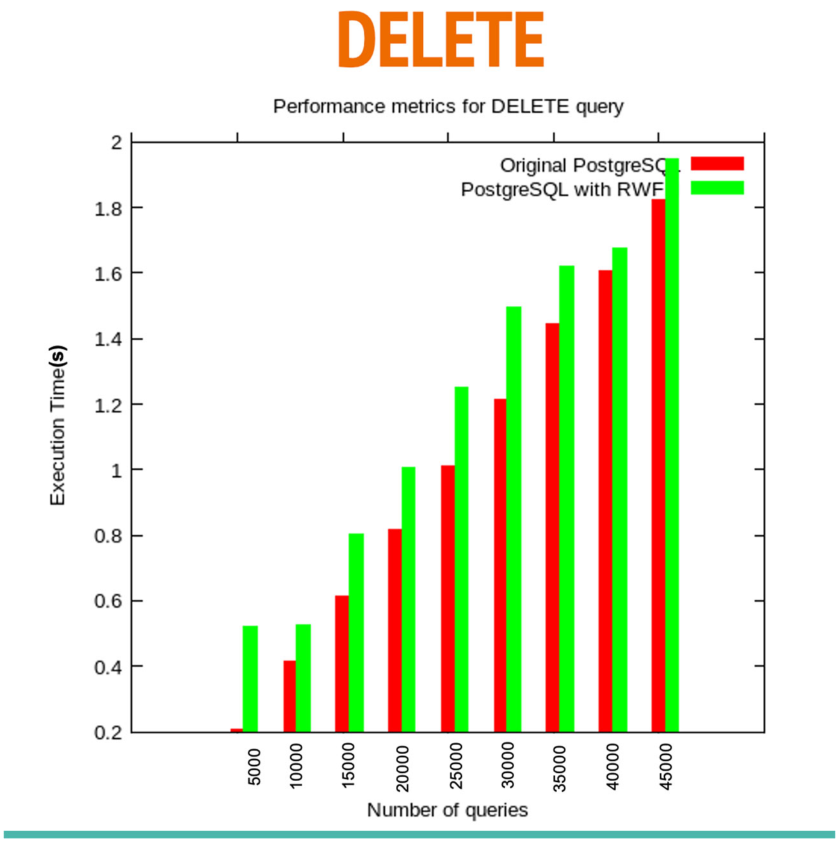


Fig. 4 Performance of DELETE



to the additional computation for various checks that are being done for each tuple. For example, to display contents of a table, only those tuples should be displayed which satisfy MLS constraints. This check is done for each of the tuple that is being fetched. Thus, the performance degrades slightly. Performance comparisons are shown in Figs. 1, 2, 3 and 4, and it is evident that the overhead is quite small.

7 HotCRP: A Case Study

In this section, we shall consider the application of RWFM for securing HotCRP – widely used conference management system, in an abstract way. It is used as a paper reviewing system and uses security policies to control information flow within the system. Authors add their papers and Reviewers review the papers and give score to them. The authors then get to know about the scores given to their papers by the reviewers without knowing the individual scores given by the reviewers; it is also possible to model to have the individual scores. The whole process is controlled by an administrator known as Program Chair. After scores of all the papers are finalized, Program Chair can declassify paper reviews to the author of the paper in using his discretion. Thus, even after using the security policies, HotCRP is still vulnerable to information leakage as the Program Chair uses essentially Discretionary Access Control for declassification; one has to trust the Program Chair(s). The subjects and objects in HotCRP are predefined and so we can use RWFM model to control information leakage and make the system more secure.

The different set of subjects present in HotCRP are as follows:

– A central administrator known as Program Chair, PC that controls the overall system. It declassifies information to other subjects as and when required.
– Set of Authors, Auth, who enter details about their papers to be reviewed.
– A set of Reviewers, Rev, who review the papers entered by Authors.

The main objects in the system are (i) set of Papers P and (ii) set of Reviews, Rev. Let us enforce the following policies:

1. Authors cannot review papers i.e., Auth $\cap$ Rev = ϕ
2. Program Chair cannot review any paper i.e., PC $\cap$ Rev = ϕ
3. A paper P when added can be viewed by all reviewers and also by the Program-Chair.
4. Each paper has an Assigned Set (AS). An assigned set is also a mapping from a paper P to a set of Reviewers

Rev. $P \rightarrow 2^{|Rev|}$ Only the reviewers present in the AS are allowed to review the paper. So C ∩ AS = ϕ

5. A reviewer is allowed to access reviews of other reviewers for a particular paper only if the reviewer has submitted his own review.

Let us understand the work flow of HotCRP with an instance of the system abstracted below:

– Let the program chair be C.
– Let us consider two authors as A1 and A2 authoring papers P1 and P2 respectively, and two reviewers RR1 and RR2. Thus, the total number of subjects are: {A1,A2,RR1,RR2,C}
– Let R11 be the review of reviewer RR1 for paper P1 and R12 be the review of reviewer RR2 of paper P1 and thus the set of objects in the system are { P1,P2,R11,R12 }.

The initial RWFM labels for the objects and the subjects are given below:

– For Paper P1 Author A1 is the author, hence A1 is the owner of P1. According to the HotCRP rule initially a paper can be read by the author, program chair and all the reviewers. So the reader set would be { A1,RR1,RR2,C }. As author A1 has written it so the writer set would be A1 itself. The RWFM labels for the paper P1 would be
  ({A1}, {A1,RR1,RR2,C}, {A1})
– Similarly for P2 the RWFM labels would be: **({A2}, {A2,RR1,RR2,C}, {A2})**
– In HotCRP a review is submitted to the Program Chair. Initially the reviewer is the owner of a review. As only the reviewer and program chair has access to the review the reader-set for a review would be the reviewer and program-chair. For the review author of the paper influences the review hence the author has to be in the writer set along with the reviewer. So RWFM labels for R11 (review of paper P1 by reviewer RR1):
  (RR1, {C,RR1}, {A1,RR1})
  RWFM labels for R12 (review of paper P1 by reviewer RR2):
  (RR2, {C,RR2}, {A1,RR2})
– For program chair, initially it can read and write only its data. hence the initial label would be
  (C, {C}, {C})

After a review is submitted by a reviewer the Program Chair(C) accesses it using the RWFM rules. After accessing the review the Program-chair stores the review on a table. The structure of the table is shown in Table 13.

According to the RWFM rule, the label for the review in the new table Review-decision by program-chair is:

(C, {C,RR1} ∩ **{C }, {A1,RR1}** ∪ **{ C})** which equals to **(C, {C} , {A1, RR1, C})**

Table 13 Table review-decision

Paper	Author	RR1	l-RR1	RR2	l-RR2	Decision
–	–	–	–	–	–	–

Similarly, for review R12, the label in the table: Review-Decision would be:

(C, {C} , {A1, RR2, C})

The decision column decides the final label of the reviews. Suppose review R11 comes first followed by R12. The label of decision would be the least upper bound of label of R11 and R12 which is **(C, {C} , {A1, RR1,RR2, C})**. The label of decision column determines who can get access to the reviews finally. It can be observed that if the program has received only one of the reviews, the chair can only declassify the object to only the reviewers who has submitted his review. However, if the chair has received both the reviews then the chair can declassify so that the RR1 can read RR2 and RR2 can read RR1. Declassification of an object to a subject is possible only when the subject is present in the writer-set of the object. Also there is a condition in HotCRP that a reviewer cannot get access to other reviews unless he submits his own review. The decision column of the paper-review table controls such possibilities. Whenever a reviewer submits his review the label of the reviewer and the current label of Decision are joined (least upper bound is taken) that includes the new reviewer in the writer set of the Decision column. This ensures that the current reviewer has submitted his review and so he can now get access to other reviews for the paper after the program-chair declassifies the paper to him. For example, after R11 is submitted it has to be checked whether RR2 has submitted his review. If only RR2 has submitted the review then only he can access the the review of RR1 that is R11. Similarly it would be for reviewer RR1 for accessing R12. RR1 can access R12 only if he has submitted his own review that is R11. Hence, the declassification process is restricted by the above conditions. This is ensured by the Decision column of the paper review table. It is to be noted that a reviewer enters into the writer set of Decision column only after his review is submitted as evident from Table 14.

Let us illustrate the process with an example. First at time T1, R11 submits the review for paper P1. Now, the Review-Decision table is as shown in Table 14.

Table 14 Review-decision - after submitting review R11

Paper	Auth	RR1	l-RR1	RR2	l-RR2	Decision
P1	A1	R11	({C}, {C}, {A1,RR1,C})	–	–	({C}, {C}, {A1,RR1,C})
.	.	.		–	–	

Table 15 Review-decision - after submitting review R12

Paper	Auth	RR1	l-RR1	RR2	l-RR2	Decision
P1	A1	R11	({C}, {C},	R12	({C}, {C},	({C}, {C}
–	–	–	{A1, RR1, C})	–	{A1, RR2,C})	{A1, RR1, RR2, C})

Table 16 Review-decision - after declassifying R11 and R12

Paper	Author	RR1	l-RR1	RR2	l-RR2	Decision
P1	A1	R11	({C}, {C},	R12	({C}, {C},	({C}, {C,RR1,RR2}
–	–	–	{A1, RR1, C})	–	{A1, RR2,C})	{A1, RR1, RR2,C})

Table 17 Review-decision - after adding reviews R21 and R22

Paper	Auth	RR1	l-RR1	RR2	l-RR2	Decision
P1	A1	R11	({C}, {C},	R12	({C}, {C},	({C}, {C}
–	–	–	{A1, RR1, C})	–	{A1, RR2,C})	{A1, RR1, RR2,C})
P2	A2	R21	({C}, {C},	R22	({C}, {C},	({C}, {C}
–	–	–	{A2, RR1, C})	–	{A2, RR2,C})	{A1,A2, RR1, RR2,C})

Table 18 Review-decision - after declassifying R21 and R22

Paper	Auth	RR1	l-RR1	RR2	l-RR2	Decision
P1	A1	R11	({C}, {C},	R12	({C}, {C},	({C}, {C}
–	–	–	{A1, RR1, C})	–	{A1, RR2,C})	{A1, RR1, RR2,C})
P2	A2	R21	({C}, {C},	R22	({C}, {C},	({C}, {C,RR1,RR2,A2}
–	–	–	{A2, RR1, C})	–	{A2, RR2,C})	{A1,A2, RR1, RR2,C})

The program chair cannot declassify the review R11 to RR2 as RR2 is not present in the writer-set of decision. This satisfies the condition in HotCRP that a reviewer cannot see other review unless he submits his own review. When RR2 submits the review R12 for paper P1. The review decision table would be as shown Table 15.

Now, program chair can:

```
Declassify to RR1 for paper P1 ;
Declassify to RR2 for paper P1 ;
Declassify to A1 for paper P1 ;
```

Updated table Review-Decision is as shown in Table 16.

Now after the declassification process RR2 gets access to the review R11, RR1 gets access to review R12 and author A1 gets access to reviews R11 and R12 . Let P2 be now submitted by author A2. Label for P2 would be **(A2, {A2,RR1,RR2,C} , {A2})** . Then RR1 and RR2 submit their reviews as R21 and R22. Decision label will perform lub operation with the new labels. Similar to R11 label of R21 would be **(RR1, {C,RR1} , {A2,RR1})** and for R22 it would be **(RR2, {C,RR2} , {A2,RR2})**. In the Decision column, least upper bound (lub) operation will be performed between the current label of decision and label of new review being submitted. The table after submitting R21 and R22 will be as in Table 17.

Program-Chair can now declassify the reviews to RR1, RR2 and A2 so that they can get access to the reviews R21 and R22. The Review-Decision table would look like as follows:

As each of the papers P3, P4 .... P_n is submitted the corresponding decision column label is changed using the least upper bound operation. Thus, after both the reviews are submitted for paper P_n the label of decision column is updated as **({C}, {C}, {A1,A2,.... A_n, RR1, RR2,C})**. Declassification is then done to the desired authors and reviewers by the program chair as depicted in Table 18.

8 Anonymization and Psuedonymization

In the previous sections, we have discussed various techniques in use for preserving privacy in DBs. As briefed already, personally identifiable information (PII) are the main source of privacy violations. PIIs, are of various forms such as, identities like name, address, Aadhar (is an identity database for Indian residents) number, location, activities like web history, friends history, online purchases, transactions, tracking on the web, health records etc. Thus, in this ubiquitous networked world, some natural questions arise:

1. Do we sacrifice privacy by using various network services (Internet, online social networks, mobile phones)?
2. How does the structure/topology of a network affect its privacy properties?

Thus, privacy leaks could be plugged, if we can prevent the leak of PIIs. Some of the techniques that are in use to mitigate the risks of PII leaks are:

1. Anonymization: the basic idea is to avoid use of real names. This technique is discussed below in some more detail.
2. Encryption: is a technique that enables to have transactions telling the outsider that it is NOYB (none of your business) or realizing the underworld technique of flyByNight. The technique of homomorphic encryption is one of the most powerful techniques that has been developed and once this is deployed, it should overcome the problem substantially modulo the last data or the last mile interface.
3. Decentralization: Leads to tighter control over data; we have discussed one side of the usage of decentralization. The other side of usage that will almost lead to democratizing of data is beyond the scope of the paper (Vishwas & Shyamasundar, 2018; Patil & Shyamasundar, 2017; Keil et al. 2019).

In the following, we briefly discuss the notions of anonymization and psuedonymization that have become essential in the context of complying with regulations like HIPAA or GDPR etc., while sharing data.

8.1 Anonymization

Anonymization is a data processing technique that removes or modifies personally identifiable information. That is, anonymisation is the process of removing personal identifiers, both direct and indirect, that may lead to an individual being identified. Anonymized data can be shared externally without putting the privacy of users at risk; for instance, the data will not fall within the scope of regulations like GDPR (General Data Protection Regulations) and it becomes easier to use. It must be noted that while there may be incentives for some organisations to process data in anonymised form, this technique may devalue the data, as the modified data of utility for some purposes. Therefore, before anonymization consideration should be given to the purposes for which the data is to be used.

Two methods of anonymizing are briefed below (i) Generalization of Data and (ii) Differential Privacy.

8.2 Generalization of Data

Certain data elements are more easily connected to some of the individuals. To protect such individuals, generalization of data is used that removes a portion of the data or replaces some part of it with a common value.

- Replacing segments of all area codes or phone numbers with the same sequence of numbers would satisfy the requirement.
- Generalization allows us to achieve k-anonymity, where k is a number that represents the size of a group, where for any individual in the data set, there are at least k-1 individuals who have the same properties.
- If all individuals in a data set share the same value of a sensitive attribute, sensitive information may be revealed simply by knowing these individuals are part of the data set in question. To overcome this risk, we may leverage *L-diversity*, to describe some level of diversity in the sensitive values. For example, imagine a group of people searched for the same sensitive health topic (e.g. flu symptoms) all at the same time. If we look at this data set, we would not be able to tell who searched for the topic, thanks to k-anonymity. However, there may still be a privacy concern since everyone shares a sensitive attribute (i.e. the topic of the query). L-diversity means the anonymized data set would not only contain flu search but also other searches alongside the flu search to further protect user privacy.

8.2.1 Pseudonymization

Psuedonymization is defined within the GDPR as ”the processing of personal data in such a way that the data can no longer be attributed to a specific data subject without the use of additional information, *as long as such additional information is kept separately and subject to technical and organizational measures to ensure non-attribution to an identified or identifiable individual*. However, it must be noted that unlike anonymization, pseudonymisztion techniques will not exempt controllers from the ambit of GDPR altogether.

Some of the examples of the above techniques have been

1. Hiding identity, removing PII
2. At a network level, using a proxy server thus hiding the IP.
3. Health data released for research purposes after removing PIIs.

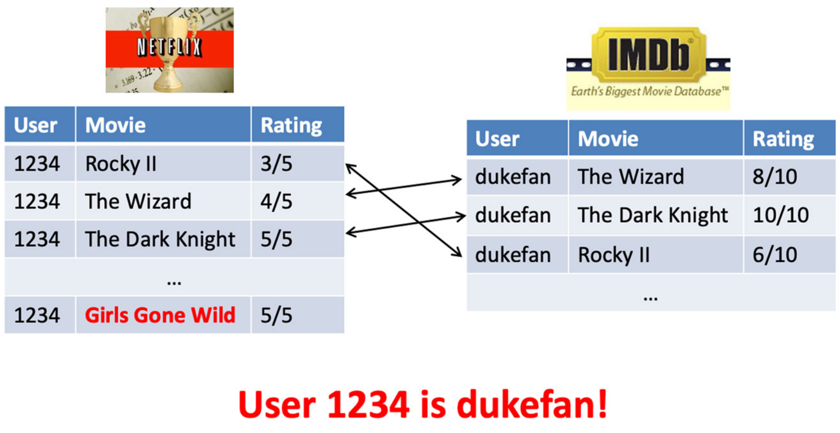
The threat for the above approaches is deanonymization. For instance consider, the famous release of Netflix prize dataset released in 2006. It had 100,000,000 (private) ratings from 500,000 users and the main competition was to improve recommendations i.e., if user X likes movies A,B,C, will also like D. In the dataset, user names were anonymized, where user name was replaced by a number. The threat could arise, if one can combine “private” ratings from Netflix with public reviews from IMDB to identify users in dataset. This may expose embarrassing info about members. For instance, considering the movie names as shown in Fig. 5 (from a private presentation from Cuervo & Shakimov (2016)), the two datasets could be linked that will lead to deanonymize the data; of course, these are probabilistic answers based on some external inputs. This implies that:

- It is not always possible to anonymize data simply by removing identifiers vulnerable to aggregating data from multiple sources/networks

This is one reason as to why one would need to look at predictability of humans and AI systems.

8.3 Inference Control and Disclosure of Information

When information derived from confidential data must be declassified for wider distribution, it is sufficient to follow information flow control as discussed already in earlier sections. For instance, in statistical databases like census data, it is possible to provide access to statistics about groups of individuals, while restricting access



User	Movie	Rating
1234	Rocky II	3/5
1234	The Wizard	4/5
1234	The Dark Knight	5/5
	...	
1234	Girls Gone Wild	5/5

User	Movie	Rating
dukefan	The Wizard	8/10
dukefan	The Dark Knight	10/10
dukefan	Rocky II	6/10
	...	

Fig. 5 An example of deanonymization

to information about any particular individual. Usually statistics contain traces of the original information. By correlating different statistics, a clever user may be able to deduce confidential information about some individual. For instance, by comparing the total salaries of two groups differing only by a single record, the user can deduce the salary of the individual whose record is in one group but not in the other. The purpose of inference controls (Denning, 1982; Farkas & Jajodia, 2002) is to ensure that the statistics released by the database do not lead to the disclosure of confidential data.

General-purpose database systems often provide both statistical and non-statistical access. For instance, in a hospital database, doctors may be given direct access to patients' medical records, while researchers are only permitted access to statistical summaries of the records. It must be noted that a statistic is said to be sensitive if it discloses too much confidential information about some individual. Obviously, a statistic computed from confidential information in a query set of size 1 is always sensitive. This means that clearly, all sensitive statistics must not permitted. Further, it may be necessary to restrict certain nonsensitive statistics if they could lead to disclosure of sensitive ones. Denning (1982) articulates, various types of statistics, statistical disclosures, entropy etc., and describes various inference control mechanisms for disclosure of information. Similar to perfect secrecy in the context of cryptography, Dwork and Naor (Dwork & Naor, 2010), quote the following desideratum: "access to a statistical database should not enable one to learn anything about an individual that could not be learned without access." and then argue that such a type of privacy cannot be achieved in general. The main issue is in information available to the adversary other than from access to the statistical database. They argue that in any 'reasonable' setting there is a piece of information that is in itself innocent, yet in conjunction with even a modified or a noisy version of the data yields a privacy breach. The example given in Fig. 5 belongs to such a category. For understanding the intuition of the authors' argument, we quote the parable given in (Dwork & Naor, 2010): "Suppose one's exact height were considered a sensitive piece of information, and that revealing the exact height of an individual were a privacy breach. Assume that the database yields the average heights of women of different nationalities. An adversary who has access to the statistical database and the auxiliary information *Terry Gross is two inches shorter than the average Lithuanian woman* learns Terry Gross' height, while anyone learning only the auxiliary information, without access to the average heights, learns relatively little". For this reason they argue for *Differential Privacy*. A surprising aspect of the height example is that the compromise can occur even if Terry Gross is not in the database. To sidestep this issue, one could shift from absolute guarantees about disclosures to relative ones: any given disclosure – indeed, any output at all of the privacy mechanism – should be, within a small multiplicative factor, just as likely independent of whether any individual opts in to, or opts out of, the database. As a consequence, there is a nominally increased risk to the individual in participating, and only nominal gain to be had by concealing or misrepresenting one's data. Note that a bad disclosure can still occur, but the differential privacy guarantee assures the individual that it will not be the presence of her data that causes it, nor could the disclosure be avoided through any action or inaction on the part of the user. A general foundation of differential privacy has been discussed in (Dwork & Naor, 2010). In the following, we discuss as to how it could be adopted for realizing privacy.

8.4 Differential Privacy (Dwork & Roth, 2014)

It describes a technique for adding mathematical noise to data. With the added noise, it's difficult to ascertain whether any one individual is part of a data set looking at the output of a given algorithm regardless of whether any one individual's information is included or omitted.

- For instance, while searching for flu across a geographic region, we can realize differential privacy by adding noise to the data set. By doing so, we may add or subtract the number of people searching for flu in a given neighbourhood, but adding noise would not affect measurement of the trend across the broader geographic region.
- However, it is important to note that adding noise to a data set may render it less useful.

General Remarks: The following points should be noted with respect to anonymisation and differential privacy:

- Note that anonymisation or differential privacy is just one process we use to maintain commitment to user privacy. Other processes include strict controls on user data access, policies to control and limit joining of data sets that may identify users.
- It must be noted that it is important to understand the responsibilities for compliance and utility, and choose appropriate hybrid techniques of anonymisation, access control and trust along with the utility (or purpose) of anonymisation of datasets.

9 Discussions and Challenges

In this paper, we have discussed an overview of the privacy techniques that had been pursued traditionally on DBs and also on application of MAC policies to arrive at FGAC on MLS DBs. We have further discussed robust realization of security with respect to information-flow using RWFM and illustrated a case study on such an application on HotCRP. While exploring privacy techniques for DBs/MLS DBs, two important aspects may be noticed:

1. Operations such as restrictions, sanitization, or declassification etc., are essential for flexible, controlled and shared usage.
2. These operations also need some level of trust in the use of the underlying reference monitors.

While techniques discussed are in use widely for realizing security and privacy, the task of sharing data is a horrendously complex task with varied challenges from application to application. Some of the principle broad categories of privacy requirements are:

1. Medical data for research: Medical wisdom is realized through a large number of experiments by a multiple parties. In the creation of such datasets, two properties are vital: privacy and provenance (Smith et al., 2004). The former is very important as the medical information the patient needs to be kept private by the individual and can be used for purposes of treatment and possible to gather data for advisories/warnings for the community. The latter becomes important for reconstructing intermediate results or new experiments from intermediate ones. This is important as due credits can be offered to the laboratories for sharing the data as that would further encourage collaboration and sharing.
2. Sharing Personal EHR (Electronic Health Records): Privacy and ownership of data by the patient are important concerns when different providers share health information of any patient. The patient may only choose to share information with the provider of her choice, and this needs to be guaranteed. That is, patient-driven interoperability of medical records becomes an important requirement (Ray, 2019)
3. Data Shared Online Social Networks (OSN): Preserving privacy on OSNs has lot of challenges (Vishwas & Shyamasundar, 2018) as one need to consider various layers such as their privacy policy with the users, the apps that are made available as well the advertising apps on OSNs. While democratising data is the requirement, data being the gold of digital world, it is not clear as to how much one can achieve it and at what cost. Issues of cybersecurity, privacy and ethical issues are explored in Acqusiti et al., (Acquistiv2009).
4. Regulations like HIPAA, GDPR etc: As highlighted already, anonymization (also psuedo-anonymization) plays a vital role in complying with such regulations. While powerful techniques exist to anonymize data, it must be noted that while anonymizing data, one has to keep in mind the utility or the application for sharing. It is for this reason that the use of the powerful technique such as differential privacy (Dwork & Roth, 2014) has limited usage. Anonymization drastically changes over the type of utility like business analytics or cybsersecurity attacks,
5. AI and ML: Training data for ML/AI preserving privacy is yet another challenge.
6. With the increasing adoption of NoSQL (not only SQL) types of database management systems by major cloud storage providers as they allow developers to store huge amounts of unstructured data, giving them a lot of flexibility; including specifying confidentiality constraints in the application at time of query instead of executing the query. This is another potential place to analyse how robustly privacy can be maintained.

Acknowledgements The work was done under the Information Security Research and Development Center at IIT Bombay sponsored by Ministry of Information Technology and Electronics, Government of India, New Delhi. The author thanks the anonymous referees for their helpful comments. It is a pleasure thank Prof. H.R. Rao for the invitation to deliver a keynote address at SKM 2019, Goa and his patience in getting the final manuscript, Dr. Vishwas Patil for his critical comments and the conference TPC chairs and organizers of SKM 2019 for the hospitality in Goa.

References

Acquisti, A., Dinev, T., & Keil, M. (eds.) (2019). Cyber security, privacy and ethics of information systems, information system frontiers, special issue. Vol. 21 6. Springer: Berlin.

Aniket, K. (2018). *Security analysis in multi-level databases*. IIT Bombay: M.Tech, Dissertation, Department of Computer Science and Engg.

Arushi, J. (2016). *Database security using Reader Writer Flow Model*. IIT Bombay: Department of Computer Science and Engg.

Chaudhary, P. (2017). *SecpostgreSQL: A system for flow-secure view, transaction, sanitization and declassification on mls database*. IIT Bombay: M.Tech. Thesis, Department of Computer Science and Engineering.

Cuervo, E., & Shakimov, A. (2016). Privacy and Networks, CPS96, private presentation (ppt).

Denning, D. E. (1976). A lattice model of secure information ow. *Communications of the ACM*, *19*(5), 236–243.

Denning, D. E. (1982). *Cryptography and data security*. Reading MA: Addison-wesley.

Denning, D. E., Lunt, T. F., Schell, R. R., Shockley, W. R., & Heckman, M. (1988). The SeaView security model. In *Proceedings, 1988 IEEE symposium on security and privacy, Oakland, CA, USA, pp. 218–233*.

Denning, D. E., Akl, S. G., Heckman, M., Lunt, T. F., Morgenstern, M., Neumann, P. G., & Roger, R.S. (1987). Views for multilevel database security. IEEE Trans. on Software Engineering.

Dwork, C., & Naor, M. (2010). On the difficulties of disclosure prevention in statistical databases or the case for differential privacy. *Journal of Privacy and Confidentiality*, *2*(1), 93–107.

Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, *9*(3–4), 211–407.

Farkas, C., & Jajodia, S. (2002). The inference problem: a survey. *SIGKDD Explorations*, *4*(2), 6–11.

Ghosal, S., Shyamasundar, R. K., & Narendra Kumar, N.V. (2019). Compile-time security certification of imperative programming languages. In *E-business and telecommunications, revised selected papers of 15th int. joint conference ICETE'18, pp 159–182, Springer CCIS vol., 1118*.

Gurevich, Y., Hudis, E., & Wing, J.M. (2016). Inverse privacy. *Communications of the ACM*, *59*(7), 38–42.

Jajodia, S., & Sandhu, R. (1991). Toward a multilevel secure relational data model. In *Proceedings of the 1991 ACM SIGMOD international conference on management of data (SIGMOD 1991), Association for computing machinery, New york, NY, USA, pp 50–59*.

Keil, M., Culnan, M., Dinev, T., et al. (2019). Data governance, consumer privacy, and project status reporting: Remembering h. Jeff smith. *Information Systems Frontiers*, *21*, 1207–1212. https://doi.org/10.1007/s10796-019-09964-4.

Myers, A., & Barbara, L. (1997). A decentralized model for information flow control. In *Proc. of the 16th ACM symposium on operating systems principles (SOSP 1997), pp 129–142 Saint Malo France*.

Narendrakumar, N. V., & Shyamasundar, R. K. (2014). Realizing purpose-based privacy policies succinctly via information-flow labels. In *IEEE int. conf. on big data and cloud computing (BdCloud), Sydney 3-5*.

Narendrakumar, N. V., & Shyamasundar, R. K. (2017). A complete generative label model for lattice-based access control models. In *Software engineering and formal methods, Trento, Italy, September 4-8, 2017, LNCS 10469, Springer International Publishing, pp. 35–53*.

Narendra Kumar, N. V., & RKS (2016). A decentralized information flow security model for multilevel security and privacy domains, US Patent 9,507,929.

Patil, V. T., & Shyamasundar, R. K. (2017). Privacy as a currency: un-regulated? In *Proc. of the 14th Int. Jt. conf.on e-business and telecommunications, Vol. 4: SECRYPT, pp.586-595, SciTePress, INSTICC, ISBN 978-989-758-259-2*.

Radhika, B. S., Kumar, N. V. N., Shyamasundar, R. K., & Vyas, P. (2020). Consistency analysis and flow secure enforcement of selinux policies. *Computer Security*, *94*, 101816.

Rakesh, A., Kiernan, J., Srikant, R., & Yirong, X. (2002). Hippocratic databases. In *Proceedings of the 28th international conference on very large data bases (VLDB 2002), VLDB Endowment, pp. 143–154*.

Ray, D. (2019). Privacy patient and ownership of electronic health records on a blockchain, ICBC 2019, LNCS, 11521, pp. 95–111.

Rizvi, S., Mendelzon, A., Sudarshan, S., & Prasan, R. (2004). Extending query rewriting techniques for fine grained access control. ACM SIGMOD.

Schoepe, D. (2014). *Information flow in databases for free*. Sweden: Masters Thesis, Chalmers University of Technology, Gothenburg.

Schultz, D. (2013). Barbara Liskov IFDB: decentralized information flow control for databases. In *Proceeding EuroSys '13, proceedings of the 8th ACM european conference on computer systems, p. 43*.

Schultz, D. (2012). Decentralized information flow control for databases, Doctoral Dissertation, v.

Shyamasundar, R. K., Narendra Kumar, N. V., Taware, A., & Vyas, P. (2018). An Experimental Flow Secure File System. In *17th IEEE international conference on trust, security and privacy in computing and communications, 1-3 pp. 790–799*.

Shyamasundar, R. K., Satheesan, S., Mittal, D., & Chaudhary, A. (2019). Sechadoop: A privacy preserving hadoop. In *Proceedings of the 12th IEEE/ACM international conference on utility and cloud computing (UCC 2019), association for computing machinery, New York, NY, USA, pp. 111–121*.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2013). Database system concepts. 6th Edition, McGraw Hill.

Smith, K., & Winslett, M. (1992). Entity modeling in the MLS relational model. VLDB.

Smith, K., Jajodia, S., Swarup, V., Hoyt, J., & Hamilton, G. (2004). Enabling the sharing of neuroimaging data through well-defined intermediate levels of visibility. *NeuroImage*, *22*, 1646–1656.

Vamshi, C., Nihita, G., Naren, N., & Shyamasundar, R.K. (2017). Secure document management through information-flow control, 7th secure knowledge management workshop (SKM 2017), Oct. 6-7 2017, St Pet. Florida.

Vishwas, P., & Shyamasundar, R. K. (2018). Efficacy of the right-to-be-forgotten on facebook, ICISS 2018, LNCS, 11281, pp. 364–385.

Vyas, P., Shyamasundar, R. K., Patil, B., Borse, S., & Sen, S. (2021). SPLinux: A information flow secure Linux, 15th IEEE SpaCCS, Oct 2021, NY, USA.

R. K. Shyamasundar is a Life Fellow of IEEE a Fellow of ACM, Distinguished ACM Speaker, a Distingusihed Alumnus of Indian Institute of Science, served as IEEE Distinguished Speaker, and was the founding Dean of School of Technology and Computer Science at the Tata Institute of Fundamental Research. He has been JC Bose National Fellow at TIFR and IIT Bombay as well as Distinguished V. Professor at the Department of Computer Science, IIT Bombay. He is an Adjunct Professor, NITIE, Mumbai. He is a fellow of all the national academies of science and engineering in India and also a fellow of world academy of sciences, Trieste, Italy. He is serving as Editor-in-Chief for Sadhana- Journal of Engineering Sciences of the Indian Academy of Sciences for Computing and Data Sciences, Subject-Area Editor for IET Blockchain Journal, serves on the Editorial boards of Journal of Parallel and Distributed Computing, and Journal of Banking and Financial Technology.

Pratiksha Chaudhary completed her M.Tech. from Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, India recently.

Arushi Jaiswal completed her M.Tech. from Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, India recently.

Aniket Kuiri completed his M.Tech. from Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, India recently.

Affiliations

R.K. Shyamasundar[1] · Pratiksha Chaudhary[1] · Arushi Jaiswal[1] · Aniket Kuiri[1]

Pratiksha Chaudhary
pratikshac@gmail.com

Arushi Jaiswal
arushi0603@gmail.com

Aniket Kuiri
aniketiitb@cse.iitb.ac.in

[1] Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Powai, Mumbai 400076, India