ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Index Copernicus Value 2016: 69.88
Volume 7; Issue 1; January-February 2019; Page No. 01-15

m-PRIVACY FOR COLLABORATIVE DATA PUBLISHING

Akanksha Jain, Dr. Dinesh Shrimali

JRN Rajasthan Vidhyapeeth University

Abstract

The collaborative data publishing problem for anonymizing horizontally partitioned data at multiple data providers is considered. A new type of "insider attack" by colluding data providers who may use their own data records (a subset of the overall data) in addition to the external background knowledge to infer the data records contributed by other data providers. This new threat and makes several contributions. The notion of m-privacy, which guarantees that the anonymized data satisfies a given privacy constraint against any group of up to m colluding data providers. A heuristic algorithm exploiting the equivalence group monotonicity of privacy constraints and adaptive ordering techniques for efficiently checking m-privacy given a set of records is presented. A data provider-aware anonymization algorithm is presented with adaptive m- privacy checking strategies to ensure high utility and m-privacy of anonymized data with efficiency. Experiments on real-life datasets suggest that this approach achieves better or comparable utility and efficiency than existing and baseline algorithms while providing m-privacy guarantee.

The goal is to publish an anonymized view of the integrated data such that a data recipient including the data providers will not be able to compromise the privacy of the individual records provided by other parties.

Key words: Anonymization, Adversary, Algorithm.

INTRODUCTION

There is an increasing need for sharing data that contain personal information from distributed databases. For example, in the healthcare domain, a national agenda is to develop the Nationwide Health Information Network (NHIN) to share information among hospitals and other providers, and support appropriate use of health information beyond direct patient care with privacy protection. Privacy preserving data analysis and data publishing has received considerable attention in recent years as promising approaches for sharing data while preserving individual privacy. When the data are distributed among multiple data providers or data owners, two main settings are used for anonymization. One approach is for each provider to anonymize the data independently, which results in potential loss of integrated data utility. A more desirable approach is collaborative data publishing, which anonymizes data from all providers as if they would come from one source, using either a trusted third-party (TTP) or Secure Multi-party Computation (SMC) protocols to do computations.

Problem Definition

An m-adversary as a coalition of m colluding data providers or data owners, who have access to their own data records as well as publicly available background knowledge BK and attempts to infer data records contributed by other data providers.

Data Publishing and Data Privacy

Corresponding author: Akanksha Jain

Society is experiencing exponential growth in the number and variety of data collections containing person-specific information. This collected information is valuable both in research and business. Data sharing is common. Publishing the data may put the respondent's privacy in risk.

Objective:

Maximize data utility while limiting disclosure risk to an acceptable level

K-Anonymity

If the information for each person contained in the release cannot be distinguished from at least k-1 individuals whose information also appears in the release. Ex. If you try to identify a man from a release, but the only information you have is his birth date and gender. There are k people meet the requirement. This is k-Anonymity.

Classification of Attributes

Key Attribute:

Name, Address, Cell Phone which can uniquely identify an individual directly Always removed before release.

Quasi-Identifier:

5-digit ZIP code, Birth date, gender

A set of attributes that can be potentially linked with external information to re-identify entities 87% of the population in U.S. can be uniquely identified based on these attributes, according to the Census summary data in 1991.

Suppressed or generalized

Sensitive Attribute:

Medical record, wage etc. Always released directly. These attributes is what the researchers need. It depends on the requirement.

l-Diversity

Distinct l-diversity

Each equivalence class has at least l well-represented sensitive values

Limitation: Doesn't prevent the probabilistic inference attacks Ex. In one equivalent class, there are ten tuples. In the "Disease" area, one of them is "Cancer", one is "Heart Disease" and the remaining eight are "Flu". This satisfies 3-diversity, but the attacker can still affirm that the target person's disease is "Flu" with the accuracy of 70%.

Entropy l-diversity

Each equivalence class not only must have enough different sensitive values, but also the different sensitive values must be distributed evenly enough.

In the formal language of statistic, it means the entropy of the distribution of sensitive values in each equivalence class is at least log(l) Sometimes this maybe too restrictive. When some values are very common, the entropy of the entire table may be very low. This leads to the less conservative notion of l-diversity.

Recursive (c,l)-diversity

The most frequent value does not appear too frequently r1<c(rl+rl+1+…+rm)

Limitations of l-Diversity

l-diversity may be difficult and unnecessary to achieve.

A single sensitive attribute

Two values: HIV positive (1%) and HIV negative (99%)

Very different degrees of sensitivity

l-diversity is unnecessary to achieve

2-diversity is unnecessary for an equivalence class that contains only negative records

l-diversity is difficult to achieve

Suppose there are 10000 records in total
To have distinct 2-diversity, there can be at most 10000*1%=100 equivalence classes

REVIEW OF LITERATURE

Most of existing work on Privacy preserving data analysis and publishing has focused on a single data provider setting and considered the data recipient as an attacker. Existing study assumes limited background knowledge of the attacker and defines privacy using relaxed adversarial notion by considering specific types of attacks. Techniques such as k-anonymity, l-diversity, and t-closeness were proposed.

However, no work considers data providers as potential attackers in the collaborative data publishing setting and explicitly models the inherent instance knowledge of the data providers as well as potential collusion between them for any weak privacy. Thus an efficient technique is required to provide collaborative data publishing.

EXISTING SYSTEM

Attacks by External Data Recipient Using Anonymized Data

- A data recipient could be an attacker and attempts to infer additional information about the records using the published data and some background knowledge (BK) such as publicly available external data.
- Bayes-optimal privacy notion is used to protect against specific types of attacks by assuming limited background knowledge.
- For example, k-anonymity, prevents identity disclosure attacks by requiring each equivalence group, records with the same quasi-identifier values, to contain at least k records.
- Representative constraints that prevent attribute disclosure attacks include l-diversity, which requires each equivalence group to contain at least l “well-represented” sensitive values
- t-closeness, which requires the distribution of a sensitive attribute in any equivalence group to be close to its distribution in the whole population.
- Differential privacy publishes statistical data or computational results of data and gives unconditional privacy guarantees independent of attackers background knowledge.

Attacks by Data Providers Using Intermediate Results and Their Own Data

- The data providers are semihonest, commonly used in distributed computation setting. They can attempt to infer additional information about data coming from other providers by analyzing the data received during the anonymization.
- A trusted third party (TTP) or Secure Multi-Party Computation (SMC) protocols can be used to guarantee there is no disclosure of intermediate information during the anonymization.

Disadvantages

- TTP or SMC do not protect against data providers to infer additional information about other records using the anonymized data and their own data
- Less privacy

PROPOSED SYSTEM

Attacks by Data Providers Using Anonymized Data and Their Own Data

- Collaborative data publishing setting with horizontally partitioned data across multiple data providers, each contributing a subset of records is considered.
- A data provider could be the data owner itself who is contributing its own records.
- Each provider has additional data knowledge of their own records, which can help with the attack.

This issue can be further worsened when multiple data providers collude with each other.

Advantages

- “Insider attack” by data providers is considered.

- High privacy for published data

METHODOLOGY

The project is developed in the following stages

* Analysis – analysis of the customer need
* Design – design of the desired solution
* Development – (technical) development of the solution
* Implementation – deployment of the developed solution in the organization
* Evaluation – evaluation the implemented solution

Waterfall development

The Waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance.

The basic principles are: * Project is divided into sequential phases, with some overlap and splashback acceptable between phases. * Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time. * Tight control is maintained over the life of the project via extensive written documentation, formal reviews, and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

Integrated development environment

An integrated development environment (IDE) also known as integrated design environment or integrated debugging environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a:

* Source code editor,
* Compiler and/or interpreter,
* build automation tools, and
* debugger (usually).

IDEs are designed to maximize programmer productivity by providing tight-knit components with similar user interfaces. Typically an IDE is dedicated to a specific programming language, so as to provide a feature set which most closely matches the programming paradigms of the language.

The IDE used in our project work is Netbeans.

Programming paradigm

A programming paradigm is a fundamental style of computer programming, in contrast to a software engineering methodology, which is a style of solving specific software engineering problems. Paradigms differ in the concepts and abstractions used to represent the elements of a program (such as objects, functions, variables, constraints...) and the steps that compose a computation (assignation, evaluation, continuations, data flows). We use java language as our program paradigm.

SYSTEM REQUIREMENTS SPECIFICATION

- Functional requirements

Req 1: Assume that hospitals P1, P2, P3, and P4 wish to collaboratively anonymize their respective patient databases

Req 2: consider patient database T1, T2, T3, and T4

Req 3: Name is an identifier, {Age, Zip} is a quasi-identifier (QI)

Req 4: Disease is a sensitive attribute. T∗ a is one possible QI-group-based anonymization using existing approaches that guarantees kanonymity and l-diversity

Req 5: ldiversity holds if each equivalence group contains records with at least l different sensitive attribute values.

Req 6: present a data provider-aware anonymization algorithm with adaptive strategies of checking m-privacy to ensure high utility and mprivacy of sanitized data with efficiency.

- Non-functional Requirements

User Interfaces
Hardware Interfaces
Ethernet
ISDN
Software Interfaces
Communications Interfaces

- Performance Requirements
- Safety Requirements
- Security Requirements
- Hardware requirements:

Processor: Any Processor above 500 MHz.
Ram: 128Mb.
Hard Disk: 10 Gb.
Compact Disk: 650 Mb.
Input device: Standard Keyboard and Mouse.
Output device : VGA and High Resolution Monitor.

- Software requirements:

Operating System: Windows Family.
Language: JDK 1.5
Database: MySQL 5.0
Software Description
JAVA
Java is an object-oriented multithread programming languages .It is designed to be small, simple and portable across different platforms as well as operating systems.
Connected Device Configuration
CDC is a smaller subset of Java SE, containing almost all the libraries that are not GUI related.
Foundation Profile
A headless version of Java SE.
Personal Basis Profile
Extends the Foundation Profile to include lightweight GUI support in the form of an AWT subset.
Personal Profile
This extension of Personal Basis Profile includes a more comprehensive AWT subset and adds applet support.

PROPOSED ALGORITHM

The binary verification algorithm
Data: A set of records T provided by P1,…., P_nG, a monotonic privacy constraint C, a privacy fitness scoring function scoreF and the m value Result: true if T $*$ is m-private, false otherwise
Step 1: begin
Step 2: sites = sort_sites(P, increasing order, scoreF)
Step 3: use_adaptive_order_generator(sites, m)
Step 4: while is_m-privacy_verified(T $*$, m) = false do
Step 5: Isuper = next_coalition_of_size(n − 1)
Step 6: if privacy_is_breached_by(Isuper) then
Step 7: continue

Step 8: Isub = next_sub-coalition_of(Isuper;m)
Step 9: if privacy_is_breached_by(Isub) then
Step 10: return false //early stop
Step 11: while is_coalition_between(Isub, Isuper) do
Step 12: I = next_coalition_between(Isub, Isuper)
Step 13: if privacy_is_breached_by(I) then
Step 14: Isuper = I
Step 15: else
Step 16: Isub = I
Step 17: prune_all_sub-coalitions(Isub)
Step 18: prune_all_super-coalitions(Isuper)
Step 19: return true

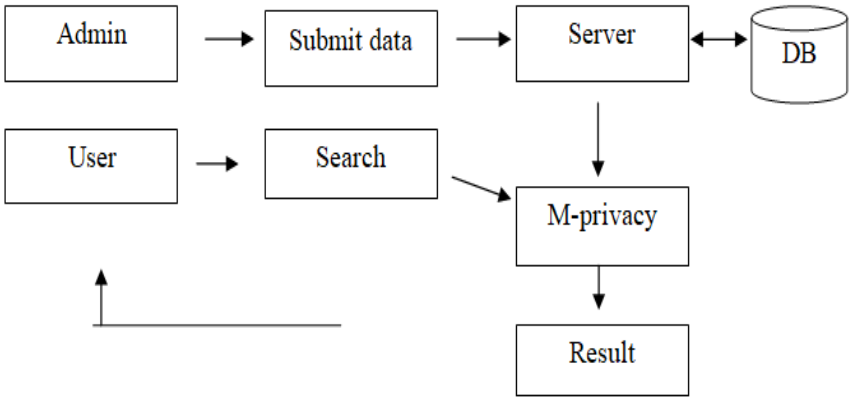
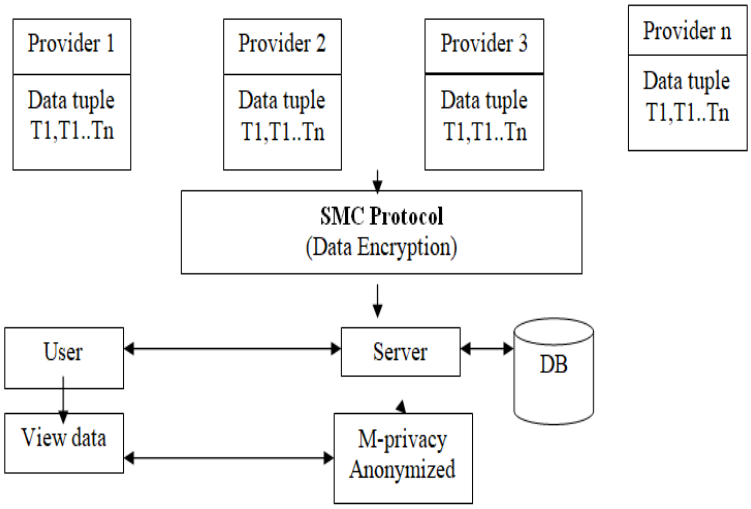


Figure1: System architecture

The above figure represents the system architecture of m-privacy, collaborative data publishing model. The above figure represents each model designed in our study.

PROJECT FLOW CHART



ACTIVITY DIAGRAM

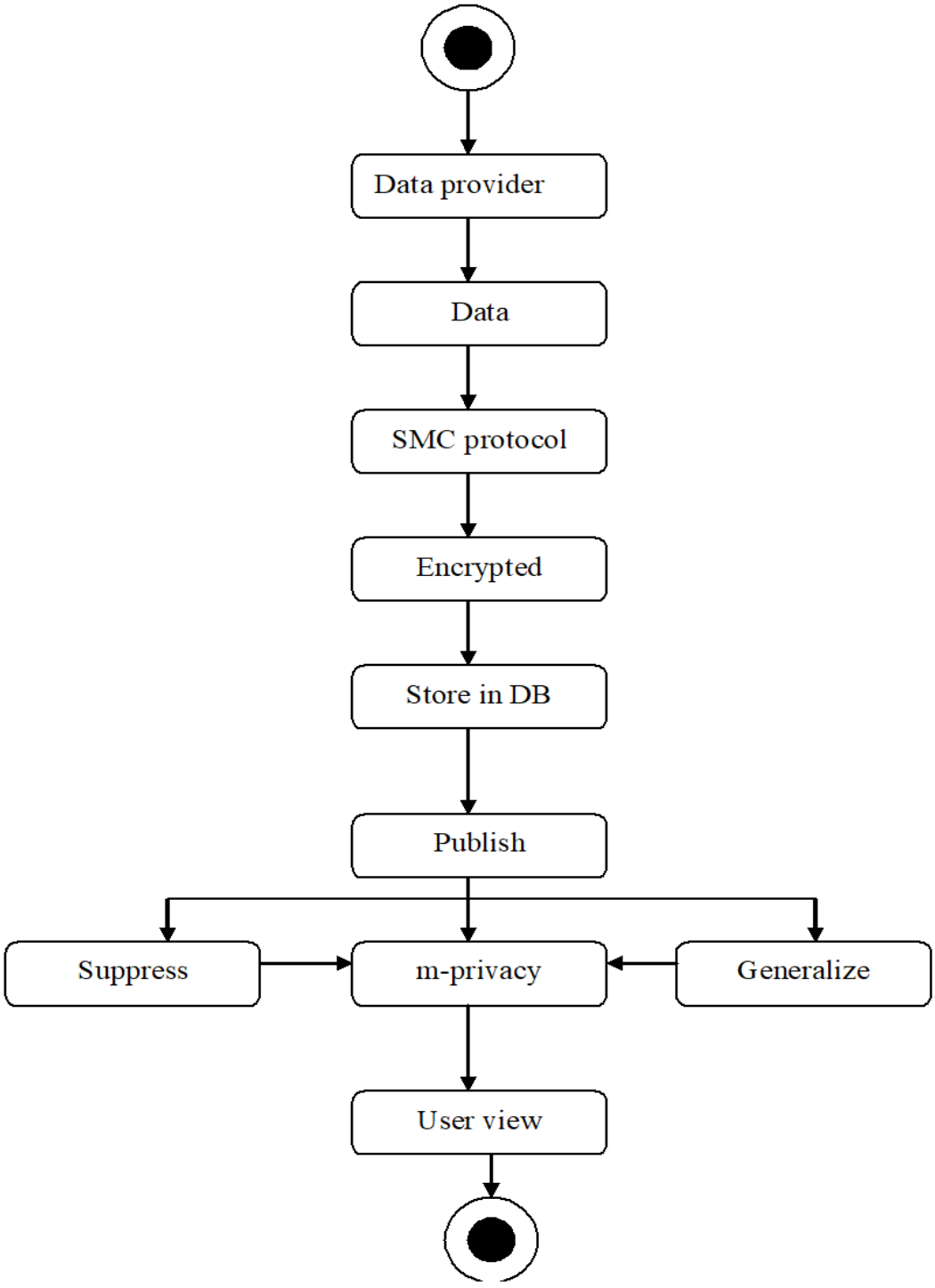


Figure 3: Activity Diagram

The above figure shows various activities handled by the system such as encryption, suppression, generalization and m-privacy.

Database Name: mprivacy
SAMPLE CODE

Dataprovider.java

```
package com.design;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JRadioButton;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.table.DefaultTableModel;

import java.sql.*;

import java.io.*;

import java.net.*;

import org.jvnet.substance.SubstanceLookAndFeel;

import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.util.*;

import com.database.DatabaseConnect;

import com.logic.Encrypter;

import com.logic.Removeduplicate;

public class DataProvider extends JFrame implements ActionListener {

      Socket requestSocket;

      ObjectOutputStream out;

      ObjectInputStream in;

      String message;
```

```
DatabaseConnect databaseConnect = new DatabaseConnect();
Removeduplicate rdup=new Removeduplicate();
Encrypter enc =new Encrypter();
String status = "";
public DefaultTableModel defaultTableModel;
public JTable jTable;
JButton button,cancel;
JLabel providername,paramName,selectarea,selectposition,ssalary;
JTextField pname,name,age,txt,txt1,disease,pincode;
static String agentName;
Connection con=null;
DataProvider() {
}
public void Design() {
      setTitle("m-Privacy for Collaborative Data Publishing -Data Provider");
      setLayout(null);
      button = new JButton("SUBMIT");
      cancel = new JButton("CLEAR");
      providername = new JLabel("Enter the Provider :");
      pname = new JTextField();
      paramName = new JLabel("Name :");
      name = new JTextField();
      selectarea = new JLabel("Age:");
      age=new JTextField();
      selectposition = new JLabel("Pincode:");
      pincode=new JTextField();
      ssalary= new JLabel("disease :");
      disease = new JTextField();
      providername.setBounds(150, 120, 100, 30);
      pname.setBounds(290, 120, 150, 30);
```

```
            paramName.setBounds(150, 160, 100, 30);
            name.setBounds(290, 160, 150, 30);
            selectarea.setBounds(150, 200, 100, 30);
            age.setBounds(290, 200, 150, 30);
            selectposition.setBounds(150, 240, 100, 30);
            pincode.setBounds(290, 240, 150, 30);
            ssalary.setBounds(150, 280, 100, 30);
            disease.setBounds(290, 280, 150, 30);
            button.addActionListener(this);
            button.setBounds(250, 320, 100, 30);
            cancel.addActionListener(this);
            cancel.setBounds(400, 320, 100, 30);
            add(button);
            add(cancel);
            add(providername);
            add(pname);
            add(paramName);
            add(name);
            add(age);
            add(selectarea);
            add(pincode);
            add(selectposition);
            add(ssalary);
            add(disease);
            setSize(700, 500);
            setVisible(true);
try{
                  requestSocket = new Socket("localhost", 2004);
                  System.out.println("Connected to localhost in port 2004");
                  out = new ObjectOutputStream(requestSocket.getOutputStream());
```

```
                  out.flush();
                  in = new ObjectInputStream(requestSocket.getInputStream());
            }catch(Exception e){
            }
}
      public static void main(String[] args) {


try {
                  SubstanceLookAndFeel
      .setCurrentTheme("org.jvnet.substance.theme.SubstanceLimeGreenTheme");
                  SubstanceLookAndFeel
      .setCurrentWatermark("org.jvnet.substance.watermark.SubstanceBinaryWatermark"
);
                  SubstanceLookAndFeel
      .setCurrentGradientPainter("org.jvnet.substance.painter.SpecularGradientPaint
er");
                  SubstanceLookAndFeel
      .setCurrentButtonShaper("org.jvnet.substance.button.ClassicButtonShaper");
UIManager.setLookAndFeel(new SubstanceLookAndFeel());
            } catch (Exception e) {
                  e.printStackTrace();
            }
DataProvider dataprovider= new DataProvider();
            dataprovider.Design();
      }
public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            if (e.getSource() == button) {
```

```
            boolean pin=verifypincode(pincode);

            System.out.println(pin);

            boolean ag=verifyage(age);

            System.out.println(ag);


            if(pin == true && ag==true) {

            try{

int gg=0;

            con = databaseConnect.DbCon();

            PreparedStatement st = con.prepareStatement("INSERT INTO
data(pname,name,age,pincode,disease)
VALUES(?,?,AES_ENCRYPT(?,'key'),AES_ENCRYPT(?,'key'),AES_ENCRYPT(?,'key'))");

            st.setString(1, pname.getText());

            st.setString(2, name.getText());

            st.setString(3, age.getText());

            st.setString(4, pincode.getText());

            st.setString(5, disease.getText());


                  gg=st.executeUpdate();

                  //JOptionPane.showMessageDialog(null,"Record Insert
Successfully");

if(gg==1)

            {

                  JOptionPane.showMessageDialog(null,"Record Insert Successfully");

            }

            else{

                  JOptionPane.showMessageDialog(null,"Record Not Insert
Successfully");

            }

                  //st.close();

            con.close();

            } catch (Exception ex) {
```

```
            System.out.println(ex);
            }


            }

            } if (e.getSource() == cancel) {


                  pname.setText("");
                  name.setText("");
                  age.setText("");
                  pincode.setText("");
                  disease.setText(""


      }
      }
     public boolean verifypincode(JTextField input) {
       String text = ((JTextField) input).getText();
       try {
          Integer.parseInt(text);
          int length = (int) Math.log10(Integer.parseInt(text)) + 1;
          System.out.println(length);
          if(length == 6){
                      return true;
         } else {


                  JOptionPane.showMessageDialog(null,"Enter 6 Digit Value for
pincode");


                  return false;
          }
```

```
        } catch (NumberFormatException e) {

            JOptionPane.showMessageDialog(null,"Enter Integer Value for pincode");
 return false;
        }

    }

    public boolean verifyage(JTextField input) {
        String text = ((JTextField) input).getText();
        try {
            Integer.parseInt(text);
            int length = (int) Math.log10(Integer.parseInt(text)) + 1;
            System.out.println(length);
            if(length == 2){

            return true;
          } else {

                    JOptionPane.showMessageDialog(null,"Enter 2 Digit Value for
age");

                    return false;
            }


        } catch (NumberFormatException e) {

            JOptionPane.showMessageDialog(null,"Enter Integer Value for age");

            return false;
```

}

CONCLUSION

A new type of potential attackers in collaborative data publishing – a coalition of data providers, called m-adversary is considered. To prevent privacy disclosure by any m-adversary we showed that guaranteeing m-privacy is enough. Heuristic algorithms are presented exploiting equivalence group monotonicity of privacy constraints and adaptive ordering techniques for efficiently checking m-privacy. We introduced also a provider-aware anonymization algorithm with adaptive m-privacy checking strategies to ensure high utility and m-privacy of anonymized data.

There are many remaining research questions. Defining a proper privacy fitness score for different privacy constraints is one of them. It also remains a question to address and model the data knowledge of data providers when data are distributed in a vertical or ad-hoc fashion. It would be also interesting to verify if our methods can be adapted to other kinds of data such as set-valued data.

REFERENCES

1. C. Dwork, “Differential privacy: a survey of results,” in Proc. of the 5th Intl. Conf. on Theory and Applications of Models of Computation, 2008, pp. 1–19.
2. B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, “Privacy-preserving data publishing: A survey of recent developments,” ACM Comput. Surv.,vol. 42, pp. 14:1–14:53, June 2010.
3. C. Dwork, “A firm foundation for private data analysis,” Commun. ACM vol. 54, pp. 86–95, January 2011.
4. N. Mohammed, B. C. M. Fung, P. C. K. Hung, and C. Lee, “Centralized and distributed anonymization for high-dimensional healthcare data,” ACM Transactions on Knowledge Discovery from Data (TKDD), vol. 4, no. 4, pp. 18:1–18:33, October 2010.
5. W. Jiang and C. Clifton, “Privacy-preserving distributed k-anonymity,” in Data and Applications Security XIX, ser. Lecture Notes in Computer Science, 2005, vol. 3654, pp. 924–924.
6. W. Jiang and C. Clifton, “A secure distributed framework for achieving k-anonymity,” VLDB J., vol. 15, no. 4, pp. 316–333, 2006.
7. O. Goldreich, Foundations of Cryptography: Volume 2, Basic Applications. Cambridge University Press, 2004.
8. Y. Lindell and B. Pinkas, “Secure multiparty computation for privacypreserving data mining,” The Journal of Privacy and Confidentiality, vol. 1, no. 1, pp. 59–98, 2009.
9. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” in ICDE, 2006, p. 24.
10. P. Samarati, “Protecting respondents’ identities in microdata release,” IEEE T. Knowl. Data En., vol. 13, no. 6, pp. 1010–1027, 2001.