Detecting Privacy Violations in Multiple Views Publishing

Deming Dou and Stéphane Coulondre

Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205, 69621, Lyon, France
{deming.dou,stephane.coulondre}@liris.cnrs.fr

Abstract. We present a sound data-value-dependent method of detecting privacy violations in the context of multiple views publishing. We assume that privacy violation takes the form of linkages, that is, identifier-privacy value pair appearing in the same data record. At first, we perform a theoretical study of the following security problem: given a set of views to be published, if linking of two views does not violate privacy, how about three or more of them? And how many potential leaking channels are there? Then we propose a pre-processing algorithm of views which can turn multi-view violation detection problem into the single view case. Next, we build a benchmark with publicly available data set, Adult Database, at the UC Irvine Machine Learning Repository, and identity data set generated using a coherent database generator called Fake Name Generator on the internet. Finally, we conduct some experiments via Cayuga complex event processing system, the results demonstrate that our approach is practical, and well-suited to efficient privacy-violation detection.

Keywords: Privacy violation, Multi-view publishing, Pre-processing algorithm, Cayuga system.

1 Introduction

Recently, privacy-preserving data publishing has been proposed whereby the data owner prevents linking some identity to a specific data record and sensitive information in the released data while, at the same time, preserves useful information for data mining purpose. Data publishing exports a set of selection-projection views instead of the whole proprietary table, data users can only access data by submitting queries against these views. Even though each published view is secure after de-identification, while an adversary can re-identify an identity by linking two or more views on their common attributes. Consider a private table $Patient(SSN, Name, ZIP, DOB, Sex, Job, Disease)$ in Table 1, SSN and Name are explicit identifiers (EIs) which can explicitly identify record holders, Disease is one of the sensitive attributes (SAs) which contain some privacy information. Suppose that the data owner releases views $v_1 = \prod_{SSN,DOB,Job,Disease}(Patient)$, $v_2 = \prod_{Name,ZIP,Job,Sex}(Patient)$ to a data recipient, and we define privacy violations by $\prod_{SSN,Disease}(Patient)$, $\prod_{Name,Disease}(Patient)$ and $\prod_{SSN,Name,Disease}(Patient)$ which means that an *EI* and a *SA* cannot appear at the same time. Obviously, each of them does not violate privacy, however, by joining v_1 and v_2 using $v_1.Job = v_2.Job$, an adversary can uniquely re-identify that *Dan* is a *Flu* patient which violates *Dan*'s privacy.

Table 1. Private table *P*

SSN	Name	ZIP	DOB	Sex	Job	Disease
387-399	Alice	47677	09/09/80	F	Manager	OC
387-200	Bob	47602	24/05/87	F	Engineer	OC
387-486	Carol	47678	08/11/82	M	Engineer	PC
387-756	Dan	47905	27/08/66	M	Dancer	Flu
387-665	Ellen	47909	04/10/57	F	Engineer	HD
387-588	Jack	47906	10/10/62	M	Manager	HD

The privacy violation presented here is not schema-dependent but data-value-dependent, for example, if we release another view $v_2^{'} = \prod_{Name,ZIP,Job,Sex}(\sigma_{Name!='Dan'}Patient)$, $v_2^{'}$ has the same schema with v_2, but the joining of v_1 and $v_2^{'}$ does not violate any privacy.

Now, we further suppose that the data holder releases $v_3 = \prod_{SSN,DOB,Sex}(Patient)$, $v_4 = \prod_{ZIP,DOB,Job}(Patient)$ and $v_5 = \prod_{ZIP,Job,Disease}(Patient)$ to another data recipient, Each of these views does not violate privacy, and neither does their pairwise joining. However, if an adversary performs at first a joining of v_3, v_4 using $v_3.SSN = v_4.SSN$ and $v_3.DOB = v_4.DOB$ which generates a mediated view $v_{34}(SSN, ZIP, DOB, Sex, Job)$, and eventually the adversary can also uniquely deduce that *Dan* is a *Flu* patient by joining v_{34} and v_5 using $v_{34}.ZIP = v_5.ZIP$ and $v_{34}.Job = v_5.Job$. Furthermore, if these two data recipients collude with each other behind the scene, the situation will inevitably become more complicated, they have 5 views in all, v_1, v_2, v_3, v_4 and v_5, and there will be much more violation channels and the risk increases. The challenge is how to detect all these potential violation channels when the number of published views is in dozens. To the best of our knowledge, no previous work takes this problem into account, and this paper is the first to present a formal analysis of privacy violation in multiple views publishing, and the first to propose an algorithm of multi-view pre-processing.

2 Related Work

Since Sweeney [8] introduced the concept of *linking attack* and the concept of *k-anonymity* in 1998, most of previous works have addressed the case of privacy protection using anonymization-based methods, either over traditional data tables, such as *l*-diversity [4], *t*-closeness [6], or over data streams, such as SWAF [10], SKY [5], CASTLE [1] and so on. We refer readers to [3] for a survey of different privacy preserving approaches with respect to privacy-preserving data publishing. Miklau and Suciu [7] performed a theoretical study of the query-view perfect-security problem to analyze the privacy violation between published relational views and queries, while the security standard is so strong that it has some practical limitations. They also presented a spectrum of information disclosure (privacy violation), total, partial and minute among which we aim to detect the first two types. In 2009, Vincent et al. [9] presented a method of detecting privacy violations using disjoint queries in database publishing, with the aim of overcoming the limitations of the perfect-security approach and provided an alternative in applications where a less stringent, but computationally tractable, method of detecting privacy violations is required. Yao et al. [11] assume that privacy violation takes the form of linkages, that is, pairs of values appearing in the same data record, the assumption of privacy violation in this paper is similar to Yao's.

3 Contribution

Similar with steps that compose the KDD (Knowledge Discovery in Databases) process, we proposed a privacy violation detection method consisting of three steps, multi-view pre-processing, data merging and event processing. In this section, we will at first give an formal analysis of the problem of detecting privacy violations in the context of multiple views publishing, then we will introduce all the steps mentioned above.

3.1 Problem Formulation and Definitions

We assume an instance I of an universal relation R with schema D, where D contains *EIs*, *SAs* and other attributes which do not fall into these two categories, written as *OAs*, and we use R^D to denote the set of all possible relations on D. Data are being published under the form of a being set which is a pair (q,r), where q is a list of selection-projection queries $(q_1,q_2,\cdots,q_n)$ on I, and r is a list of relations $(r_1,r_2,\cdots,r_n)$ such that $\exists G \in R^D, r_i = q_i(G)$ for each $i = 1,2,\ldots,n$. If q is given, we can abbreviate (q,r) to V which contains n views, and denote (q_i,r_i) by v_i where v_i is in V.

Definition 1 (privacy violation). *Given a direct published or a mediated view v_i of a underlying data table, $\alpha \subseteq EIs$ and $\beta \subseteq SAs$, and a is a constant value of α and b of β, if the result of $\prod_{\alpha,\beta}(v_i)$ is a finite set of data records, a privacy violation occurs if and only if* $|\ \sigma_{\alpha=a,\beta=b}(\Pi\alpha,\beta(v_i))\ | = 1$.

Problem Definition. Given a set of published views $v_1,v_2,\cdots,v_n$ of the same underlying data table, detect any total and partial privacy violation which could be deduced from one single view v_i or the joins $\bowtie (v_i,v_j,\cdots,v_k)$, where $2 \le k-i \le n$.

3.2 Multi-view Pre-processing

In this paper, we regard the views as nodes of a graph with an edge between two nodes representing a natural join of them. We model our approach as a simple undirected graph $G(V,E)$ with a finite non-empty view set V and a set of symmetric view pairs E. For each ordered view pair $(w,u) \in E$, the pair (u,w) also belongs to E. G_i owns some part of G, denoted as $G(V_i,E_i)$ where $\bigcup_i V_i = V$, $\bigcup_i E_i \subseteq E$. The natural join of each view pair in E is an edge, E itself is also called the *edge set* of G.

As we mentioned above, two views having common attributes do not always violate privacy, for graph G, that can be saying as inexistence of some edges of E. Take v_1,v_2,v_3 as an example, the sub-graph G_1 can be defined by the view set $V_1 = \{v_3,v_4,v_5\}$ and $E_1 = \{\{(v_3,v_4),(v_4,v_3)\},\{(v_3,v_5),(v_5,v_3)\},\{(v_4,v_5),(v_5,v_4)\}\}$ among which the view pairs $\{(v_3,v_4),(v_4,v_3)\}$ generate edge v_{34} and $\{(v_4,v_5),(v_5,v_4)\}$ generate edge v_{45}, while the view pairs $\{(v_3,v_5),(v_5,v_3)\}$ generate no edge.

Definition 2 (an edge). *Given view pairs $\{(v_a, v_b), (v_b, v_a)\}$, $v_a \in G_i$, $v_b \in G_j$, an edge exists if and only if $\exists t : t \in v_a \bowtie v_b$, t is a data record. If $i = j$, the edge is called an "inner-edge", otherwise we call it an "inter-edge".*

In our approach, we define the published views $v_1, v_2, \cdots, v_n$ as V_1 in level G_1, then $| V_1 |= n$, and $| E_1 | \leq C_2^n$, E_1 can be denoted as following:

$$E_1 = \{\{(v_1, v_2), (v_2, v_1)\}, \{(v_1, v_3), (v_3, v_1)\}, \cdots, \{(v_a, v_b), (v_b, v_a)\}\}, 1 \leq a, b \leq n$$

According to Definition 2, we know that the edge number of G_1 is no larger than C_2^n. It is possible that there is no edge in level G_1.

Proposition 1. *Given $v_a \in V_i$, $v_b \in V_j$, $i \neq j$, an "inter-edge" equals to a view v_c in V_m or v_d in V_j where $m > j > i$.*

Proof. By the commutative property, distributive property and associative property of binary operations in relational algebra.

As shown in Fig.1, for G_1, we assume that all view pairs in E_1 generate m edges, next we take these m edges as the view set of G_2, by executing natural join of each view pair in E_2, we get k edges, and then we take these k edges as the view set of G_3, the rest can be deduced by analogy. Corresponding to these operations, we propose a multi-view pre-processing algorithm, as shown in Algorithm 1, in this algorithm, we take n views $v_1, v_2, \ldots, v_n$ and their respective schemes $r_1, r_2, \ldots, r_n$ as original inputs, at first we get the universal relation R by the union loop $R = R \cup r_i$, we assume that we know the number of views in the first level G_1 is $L_1 = n$, and after pre-processing of these n views, we get L_2 views in second level G_2 and their corresponding schemes, the rest operations can be done in the same manner until there is no more view in next level. Finally, we collect the views in each level and get $n + l$ views $v_1, v_2, \ldots, v_n, v_{n+1}, v_{n+2}, \ldots, v_{n+l}$.

One advantages to perform this multi-view pre-processing operation is that we can transform partial information disclosure detection problem to total information detection case. Take v_3, v_4 and v_5 as an example, if they are the input views in our algorithm, after being processed using our multi-view pre-processing algorithm, we get three more

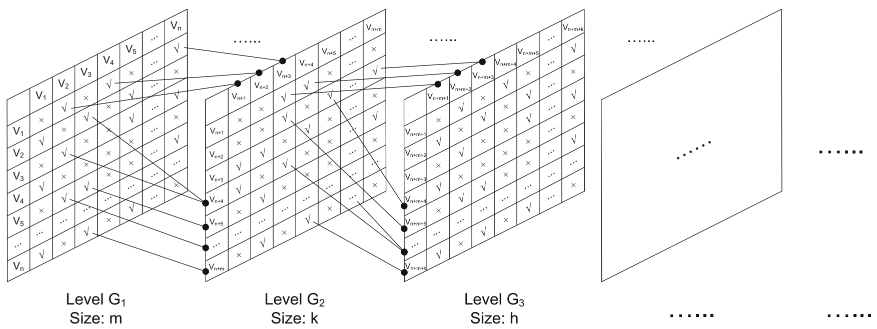


Fig. 1. Schematic diagram for multi-view pre-processing

Algorithm 1. Pre-processing Algorithm of Multiple Views

```
Input: $n$ views: $v_1, v_2, \ldots, v_n$ and corresponding schemes $r_1, r_2, \ldots, r_n$ .
Output: $(n+l)$ views: $v_1, v_2, \ldots, v_n, v_{n+1}, v_{n+2}, \ldots, v_{n+l}$.
1: Let $R = r_1, s = 1, i = 1, l = 0$
2: for $i \leftarrow 2 \ldots n$ do
3:     $R = R \cup r_i$ // R is a universal relation
4: end for
5: Let $L_i$ be the number of views in level $G_i$, $L_1 = n$.
6: repeat
7: for each level $G_i$ in $G$ do
8:     for $j \leftarrow (l+1) \ldots L_i - 1$ do
9:         for $k \leftarrow (j+1) \ldots L_i$ do
10:            if $r_j \neq R$ and $r_k \neq R$ then
11:                if $r_j \cap r_k \neq \emptyset, r_j \subset (r_j \cup r_j), r_k \subset (r_j \cup r_k)$ then
12:                    $v = v_j \bowtie v_k$ //natural join of two views
13:                    if $v \neq \emptyset$ then
14:                        $v_{L_i+s} = v_j \bowtie v_k$ //generate views for next level $G_{i+1}$
15:                        $r_{L_i+s} = r_j \cup r_k$ //generate corresponding relations
16:                        $s = s + 1$
17:                    end if
18:                end if
19:            end if
20:        end for
21:        $L_{i+1} = L_i + s$ //number of views in next level
22:        $l = l + s$ //total number of views in all levels
23:        $i = i + 1$ //total number of views in all levels
24:    end for
25: end for
26: until $s = 0$ //no views in next level
```

views v_{34}, v_{45} and v_{345}. At this time, v_3, v_4, v_5, v_{34}, v_{45} and v_{345} are the final inputs to data merging step, what we need to do is to check if every tuple of these final views violate specific privacy.

3.3 Data Merging

We use Cayuga [2] as our event processing engine, however Cayuga can only read stream data from disk files, such as a text file, or TCP sockets, this implies that we must merge our multiple pre-processed views into a single file and then store it in a disk. In our system, we adopt an universal relational assumption, however, the original published views and the views generated from the pre-processing algorithm have different schemes, we must add to different views different additional attributes and set their values as *Null*. Take v_1 and v_2 as an example, the universal relation is {Name, ZIP, Job, DOB, Sex, Disease}, v_1 does not have attributes {Name, Sex}, while v_1 does not have {DOB, Disease}, so we merge them like follows in Table 2:

Table 2. Merged data of v_1 and v_2

Name	ZIP	DOB	Job	Sex	Disease
	47677	09/09/80	Manager		OC
	47602	24/05/87	Engineer		OC
	......	......	......		......
Alice	47677		Manager	F	
Bob	47602		Engineer	M	
......	......		......	......	

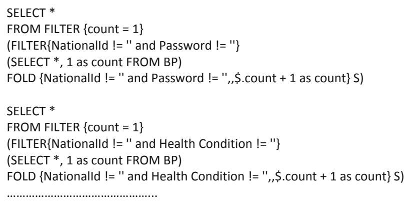
3.4 Event Processing

After data merging, we now investigate the details of event processing step which is Cayuga-based. Cayuga system is an expressive and scalable Complex Event Processing (CEP) system developed at the Cornell Database Group, it provides a SQL-like query language, Cayuga Event Language (CEL), for expressing complex event patterns, instead of operating on tables, Cayuga system performs continuous queries on data streams with multi-query optimization, it can serve as an event engine in a larger software environment without being obtrusive where the end user/application interacts with Cayuga by submitting queries written in CEL to it and receiving query result streams from it. For further information, we refer readers to Cayuga's official website `http://www.cs.cornell.edu/bigreddata/cayuga/`. The benefit of using Cayuga system other than the other DBMS is that it has an high-speed query processing engine which can handles simultaneous events for large-scale data-sets and data streams. What the data owner should do is to launch specific continuous queries over the final merged data.

4 Case Study

4.1 Experimental Settings

Dataset Benchmark. We adopt the publicly available Adult Database at the UCI Machine Learning Repository with 15 attributes, if records with missing values and unknown values are removed, there are 45,222 instances. In order to build a representative benchmark *BP*, we get another identity data set generated using a coherent database generator called Fake Name Generator on the internet which contains 30 attributes. We integrate these two datasets and keep the same number of instance, 45,222 tuples, and only retain some of their attributes. Then we add a column with sensitive values called "Health Condition" consisting of HIV, Cancer, Phthisis, Hepatitis, Obesity, Asthma, Flu, Indigestion to the extracted data and randomly assign one sensitive value to each record. The random technique works in the following way. First, assign a number to each sensitive attribute, i.e., 1: HIV, 2: Cancer, 3: Phthisis, 4: Hepatitis, 5: Obesity, 6: Asthma, 7: Flu, 8: Indigestion. Second, for each tuple (record), generate a random number from 1-8. Then, assign the corresponding sensitive attribute value to the tuple. For example, for the first tuple in the data set, if the random number is 5, then this record has the sensitive value "Obesity". "NationalId" and "Givenname" are chosen as the *EIs*, and "Health Condition" and "Password" are *SAs*.



```
SELECT *
FROM FILTER {count = 1}
(FILTER{NationalId != '' and Password != ''}
(SELECT *, 1 as count FROM BP)
FOLD {NationalId != '' and Password != '',,$.count + 1 as count} S)

SELECT *
FROM FILTER {count = 1}
(FILTER{NationalId != '' and Health Condition != ''}
(SELECT *, 1 as count FROM BP)
FOLD {NationalId != '' and Health Condition != '',,$.count + 1 as count} S)
..............................................
```

Fig. 2. Cayuga continuous queries

Table 3. Six views to be published

Views to be published
$v_1 = \Pi_{NationalId,Birthday,Gender}(BP)$
$v_2 = \Pi_{Givenname,Zipcode,Education}(BP)$
$v_3 = \Pi_{Zipcode,Company,CCType}(BP)$
$v_4 = \Pi_{Gender,CCType,Password}(BP)$
$v_5 = \Pi_{Birthday,Zipcode,Health\ Condition}(BP)$
$v_6 = \Pi_{Givenname,CCType,Birthday}(BP)$

Views and Continuous Queries. In our experiments, we try to publish six views $v_1, v_2, v_3, v_4, v_5, v_6$ on our benchmark *BP*, as shown in Table 3. In our benchmark *BP*, we assume that there are two *EIs*, "NationalId" and "Givenname", and two *SAs* "Health Condition" and "Password", therefore the number of continuous queries we should launch is $C_1^2 \times C_1^2 = 4$, as shown in Fig.2, the continuous queries are written in the structured Cayuga Event Language (CEL) and will be executed in the query engine of Cayuga system.

4.2 Detecting Results

For the six published views, the universal relation contains 10 attributes, they are NationalId, Givenname, Birthday, Gender, Zipcode, Education, Company, CCType, Password, and Health Condition. After multi-view pre-processing, the second level has 13 views, and every two of them have common attributes, meanwhile their natural joins have more than one data records, therefore in the third level we have $C_2^{13} = 78$ views, while for the fourth level and fifth level, the number of view is not a combination of the previous levels.

After being processed via all steps of multi-view pre-processing, data merging, and event processing using Cayuga system, we successfully get more than 45222 records which violate privacy, because privacy violations occur in some levels of *G*, as shown in Table 4.

Table 4. Detection outputs of privacy violation

NationalId	Givenname	Birthday	Gender	···	Password	Health Condition
170-40-9312	John	6/5/1988	male	···	be6Poowooph	Asthma
508-98-9112	Steven	6/30/1983	male	···	acoTo5ah	Asthma
130-84-3154	Patrick	2/2/1941	male	···	paiS0roo3	Flu
······	······	······	······	···	······	······

5 Conclusion and Future Work

We present a formal analysis of privacy violation in the context of multi-view publishing: given a set of published views, if linking of two views does not violate privacy, how about linking of multiple views? And how many potential violation channels are there? We propose a pre-processing algorithm of multiple views which can transform partial violation problem to the total case. Then, we build a benchmark with Adult Database at UCI Machine Learning Repository, and an identity data set generated using Fake Name Generator on the internet. Finally, we conduct some experiments via Cayuga system, the results demonstrate that our approach is practical, and well-suited to efficient privacy-violation detection. However, we did not conduct the performance evaluation of the proposed algorithm given that the number and the size of the input views can be very large, and we did not take the timestamps of views into account, they are useful if we want to detect privacy violation within a specific time interval. Therefore, future work includes performance evaluation, detecting minute privacy violation, algorithm optimization and develop a Cayuga-based privacy violation detector.

References

1. Cao, J., Carminati, B., Ferrari, E., Tan, K.-L.: Castle: A delay-constrained scheme for ks-anonymizing data streams. In: ICDE, pp. 1376–1378 (2008)
2. Demers, A., Gehrke, J., Cayuga, B.P.: A general purpose event monitoring system. In: CIDR, pp. 412–422 (2007)
3. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. ACM Comput. Surv. 42(4), 1–53 (2010)
4. Kifer, D., Gehrke, J.: l-diversity: Privacy beyond k-anonymity. In: ICDE 2006: Proceedings of the 22nd International Conference on Data Engineering, p. 24 (2006)
5. Li, J., Ooi, B.C., Wang, W.: Anonymizing streaming data for privacy protection. In: ICDE 2008: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering, pp. 1367–1369. IEEE Computer Society, Washington, DC (2008)
6. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE (2007)
7. Miklau, G., Suciu, D.: A formal analysis of information disclosure in data exchange. In: SIGMOD 2004: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 575–586. ACM, New York (2004)
8. Samarati, P., Sweeney, L.: Generalizing data to provide anonymity when disclosing information. Technical report (March 1998)
9. Vincent, M.W., Mohania, M., Iwaihara, M.: Detecting privacy violations in database publishing using disjoint queries. In: EDBT 2009: Proceedings of the 12th International Conference on Extending Database Technology, pp. 252–262. ACM, New York (2009)
10. Wang, W., Li, J., Ai, C., Li, Y.: Privacy protection on sliding window of data streams. In: COLCOM 2007: Proceedings of the 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing, pp. 213–221. IEEE Computer Society, Washington, DC (2007)
11. Yao, C., Wang, X.S., Jajodia, S.: Checking for k-anonymity violation by views. In: VLDB 2005: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 910–921. VLDB Endowment (2005)