รายงานวิจัยฉบับสมบูรณ์

โครงการการพัฒนาดัชนีสำหรับแนวทางการรักษาความเป็นส่วนตัวแบบการเรียงสับเปลี่ยน

โดย

จักรพงศ์ นาทวิชัย

Xue Li

อัศนีย์ ก่อตระกูล

พฤษภาคม 2555

สัญญาเลขที่ TRG5380024

รายงานวิจัยฉบับสมบูรณ์

โครงการการพัฒนาดัชนีสำหรับแนวทางการรักษาความเป็นส่วนตัวแบบการเรียงสับเปลี่ยน

จักรพงศ์ นาทวิชัย มหาวิทยาลัยเชียงใหม่
Xue Li The University of Queensland Australia
อัศนีย์ ก่อตระกูล มหาวิทยาลัยเกษตรศาสตร์

สนับสนุนโดยสำนักงานกองทุนสนับสนุนการวิจัย

Acknowledgement

This work was supported in parts by Thailand Research Fund and Thai Network Information Center Foundation (THNIC). The investigator really appreciates both organizations for giving the chance to carry on the project.

The investigator would like to thank Associate Professor Xue Li at The University of Queensland, Australia, and Associate Professor Asanee Kawtrakul at Kasetsart University for their advices throughout the project. In addition, the investigator is grateful to my colleagues at Department of Computer Engineering, Faculty of Engineering, Chiang Mai University for their supports.

Abstract

Project Code: TRG5380024

Project Title: Development of Indexing for Permutation-based Privacy Preservation Approach

Investigator: Juggapong Natwichai, Chiang Mai University

E-mail Address: juggapong@eng.cmu.ac.th

Project Period: May 31, 2010 – May 30, 2012

The emerging of the internet-based services poses a privacy threat to the individuals. Data transformation to meet a privacy standard becomes a requirement for typical data processing for the services. (*k, e*)-anonymization is one of the most promising data perturbation-based transformation approaches, since it can provide high-accuracy aggregate query results. Our work focuses on 1) study the effect of the permutation-based privacy-preservation processes on the indexes in term of the efficiency, 2) propose an efficient index structure for aggregation querying the permuted data. In order to achieve such goals, we begin with analyzing the dynamism on the indexes caused by the data updating. Specifically, we focus on the appending of data to the permuted dataset. We start with making the observation on the data appending theoretically. Subsequently, an algorithm based on the observation is proposed. In which the quadratic-complexity processing on some part of the dataset can be replaced by the linear-complexity processing. Eventually, two indexes, which can tolerate the data updating, are proposed to improve the efficiency of the data processing. The experiments have been conducted to validate our work. From the results, the proposed work composed of an algorithm and the indexes is highly efficient comparing with the non-incremental algorithm and an approximation algorithm, while the same results with re-applying the optimal non-incremental algorithm can be guaranteed.

Keywords : Privacy Preservation; Indexing; Incremental Processing

บทคัดย่อ

รหัสโครงการ: TRG5380024

ชื่อโครงการ:การพัฒนาดัชนีสำหรับแนวทางการรักษาความเป็นส่วนตัวแบบการเรียงสับเปลี่ยน

ชื่อนักวิจัย: จักรพงศ์ นาทวิชัย (มหาวิทยาลัยเชียงใหม่)

E-mail Address: juggapong@eng.cmu.ac.th

ระยะเวลาโครงการ: 31 พฤษภาคม 2553 – 30 พฤษภาคม 2555

การได้รับความนิยมของบริการทางเทคโนโลยีสารสนเทศผ่านเครือข่ายอินเตอร์เน็ตส่งผลให้การละเมิดความเป็นส่วนตัวของข้อมูลสามารถทำได้ง่ายขึ้น ดังนั้น การแปลงข้อมูลเพื่อรักษาความเป็นส่วนตัวได้กลายเป็นสิ่งจำเป็นก่อนการประมวลผลข้อมูลในการใช้บริการ ทั้งนี้ (*k*, *e*)-Anonymization เป็นการแปลงข้อมูลเพื่อรักษาความเป็นส่วนตัวในแนวทางของการสับเปลี่ยนที่สำคัญประเภทหนึ่ง เนื่องจากการแปลงข้อมูลด้วยวิธีดังกล่าวสามารถให้ข้อมูลผลลัพธ์ที่ยังนำไปใช้งานต่อได้อย่างแม่นยำ ในงานวิจัยนี้ มีวัตถุประสงค์ 2 ประการ ได้แก่ 1) ศึกษาผลกระทบของการรักษาความเป็นส่วนตัวด้วยแนวทางการสับเปลี่ยนต่อดัชนีในแง่ของประสิทธิภาพ 2) การเสนอโครงสร้างดัชนีสำหรับการสอบถามแบบเชิงกลุ่มบนข้อมูลที่แปลงแล้ว เพื่อที่จะบรรลุวัตถุประสงค์เหล่านี้ ผู้วิจัยได้วิเคราะห์การเปลี่ยนแปลงของดัชนีซึ่งเกิดจากการปรับข้อมูล โดยมุ่งเน้นที่กระบวนการเพิ่มข้อมูลเข้าสู่ข้อมูลที่แปลงแล้วเป็นหลัก การแก้ปัญหาเริ่มด้วยการสร้างข้อสังเกตเกี่ยวกับการเพิ่มข้อมูลอย่างเป็นระบบ จากนั้นได้นำเสนออัลกอริทึมแบบเพิ่มขึ้นจากข้อสังเกตนั้น ทั้งนี้อัลกอริทึมสามารถหลีกเลี่ยงการประมวลผลที่มีความซับซ้อนเชิงคำนวณในลักษณะ Quadratic ในบางส่วนของข้อมูลด้วยการประมวลผลที่มีความซับซ้อนเชิงคำนวณในลักษณะ Linear ท้ายที่สุดดัชนีสองประเภทซึ่งสามารถรองรับการเปลี่ยนแปลงข้อมูลได้ถูกนำเสนอเพื่อเพิ่มประสิทธิภาพการประมวลผล จากการทดลอง งานที่นำเสนอทั้งในส่วนของอัลกอริทึมและดัชนีนั้น มีประสิทธิภาพสูงกว่าอัลกอริทึมแบบดั้งเดิม และอัลกอริทึมแบบประมาณ โดยที่สามารถให้ผลลัพธ์แบบดีที่สุดได้

คำหลัก : การรักษาความเป็นส่วนตัวข้อมูล; ดัชนี; การประมวลผลแบบเพิ่มขึ้น

Contents

1 Introduction 1

2 Related Work 5
2.1 Traditional Privacy Preservation 5
2.2 Current Privacy-Preserving 6
2.3 Sensitive Pattern Hiding 13
2.4 Other Related Work 17

3 Data Increment on (k, e)-Anonymization 21
3.1 Problem Definition 21
3.2 Effect of Data Increment 24
3.3 Incremental Processing 32

4 Incremental Algorithm 35

5 Indexing 39
5.1 Distinct Value Index 39
5.2 Positioning Index 40

6 Experiment Results 43
6.1 Configuration 43
6.2 Results and Discussion 44

7 Conclusion 51

Bibliography 53

Chapter 1

Introduction

In the information-burst era as these days, the organizations who provide services, or service provider, have more opportunity to collect the data from the individuals through their services. It is obvious that the collected data can be used to improve the quality of service which will benefit both the service providers and the individuals. An obvious evidence of the data utilization by services are the sale promotion through location-based services by Starbucks[26, 55, 63].

However, the benefit of the data holding is not obtained for free, the service providers are forced to preserve the privacy of the data both in term of the legislation or the social enforcement. Not only the privacy breach can damage the right of the individuals, but also the reputation of the service providers. For example, Vodafone have been warned to face a penalty since there is a leak of their customer information [46]. Or, in the medical domain, there was the privacy violation complained by the customers of Walgreens [23].

Consequently, the privacy preservation becomes one of the most active research areas. One of the first few privacy preservation approaches which have been proposed, is k-anonymization [60]. Its fundamental is to hide an individual within a k-size group of identical information. Subsequently, the l-diversity, which further protects the privacy by enforcing the minimum l-distinct sensitive data in the group, has been proposed in [41]. Such two mentioned approaches are based on the data transformation by generalization, i.e. changing a privacy sensitive value into its more-generalized value e.g. from a postal code 50202 to 5020*. Furthermore, there are a few more effective approaches based on such work to preserve the privacy, for example, t-closeness [38] which extends the l-diversity approach by preserving the distribution of the sensitive values in each

Table 1.1: An example dataset

Postal Code	Age	Sex	Salary
50200	35	Male	14,000
50210	36	Male	15,000
50230	40	Male	16,000
50300	41	Female	25,000
50310	43	Female	35,000
50330	47	Male	30,000
50300	53	Male	40,000
50310	54	Female	35,000
50330	58	Male	45,000

Table 1.2: The partitioned dataset

Postal Code	Age	Sex	Salary
50200	35	Male	15,000
50210	36	Male	16,000
50230	40	Male	14,000
50300	41	Female	25,000
50310	43	Female	30,000
50330	47	Male	35,000
50300	53	Male	35,000
50310	54	Female	40,000
50330	58	Male	45,000

group.

One of the most important approaches for preserving the privacy is (k, e)-anonymization [53, 54]. With this approach, the sensitive data are partitioned into groups in which each group the probability to identify the individuals is not exceeded the pre-specified thresholds, k and e. The k values use to control the number of distinct sensitive data in a group, while the e values use to control the range of the sensitive data. Subsequently, the sensitive data in each partition are swapped, or permuted, to protect the privacy. Not only that the privacy can be preserved effectively, but also a high data utility can be obtained by applying such approach. The utility in the (k, e)-anonymization is defined in term of the accuracy of the aggregate query results. In [54], the authors reported that the error from the aggregate query result is very low (1% error approximately), i.e. the result is very close to the non-privacy preserved result.

An example dataset to illustrate the (k, e)-anonymization is shown in Table 1.1. In the table, a dataset is composed by three non-sensitive attributes, i.e. postal code, age, and sex, of the individuals. The sensitive attribute is the

monthly-salary in Thai Baht (THB). If the k and the e values are set at 3 and 2,000 respectively, an optimal solution with regard to the minimum-sum-error objective (to be described in Chapter 2), is shown in Table 3.4.

From Table 3.4, the partition of the dataset is outlined, i.e. there are three partitions. It can be seen that there is at least 3 distinct salary values in each partition, as well as the range of the salary is at least 2,000 THB. The sensitive data have been permuted already. If an aggregate query "find the summation of the salary of the 40-48 years-old" is issued, the result from the permuted dataset is 104,000-106,000 THB (A data tuple from the first partition, all of the tuples from the second partition, and none from the last partition are involved in the query answering). Meanwhile the same query executed on the original dataset results in 106,000 THB which is very close to the result obtained from the privacy preserved dataset.

The complexity of the algorithm generated the optimal solutions is $O(n^2)$ where n is the number of data tuples in the given dataset [54]. Such algorithm composes of two phases, i.e. partitioning the given dataset according to (k, e)-anonymization, as well as permuting the sensitive values within each partition. The partitioning phase, which costs $O(n^2)$, is our focus in this paper. Though, such cost is not very high, the data to be processed might not be static. There are some situations that the data to be transformed can be appended all the time [57, 61]. For example, the Ministry of public health in Thailand requires each hospital to submit the monthly medical records to the global repository. Re-applying the $O(n^2)$-algorithm might not be an efficient way to approach the problem in such environment.

In this work, the main objectives are as follows.

1. Study the effect of the permutation-based privacy-preservation processes on the indexes in term of the efficiency.

2. Propose an efficient index structure for aggregation querying the permuted data.

In brief, we have achieved both of our objectives. That is, the changes by the privacy-preservation processes cause the indexes change as well. This can cause the efficiency issue due to the index updating. Index dropping or re-creating might not be appropriate since the cost can be high. The efficiency issue comes from the dynamism of the data which can be changed by the users. Thus,

we focus on the insertion of the data tuples to the permuted dataset. More specifically, we address the incremental privacy preservation problem based on the (k, e)-anonymization. We start with making the observation on the data appending theoretically. Subsequently, an algorithm based on the observation is proposed. In which the quadratic-complexity processing on some part of the dataset can be replaced by the linear-complexity processing. Eventually, two indexes are proposed to improve the efficiency of the algorithm. The experiments have been conducted to validate our work. From the results, the proposed work composed of an algorithm and the indexes is highly efficient comparing with the non-incremental algorithm and an approximation algorithm, while the same results with re-applying the optimal non-incremental algorithm can be guaranteed.

The organization of this report is as follows. The related work is presented in the next chapter. The basic notation for defining the problem as well as the observations on the data increment are presented in Chapter 3. Subsequently, the algorithm based on the observation is proposed in Chapter 4. The indexes for further efficiency improvement are proposed in Chapter 5. In Chapter 6, the experiment results to evaluate our work are reported. Finally, Chapter 7 gives the conclusion and outline of our future work.

Chapter 2

Related Work

In this chapter, we begin by briefly reviewing the works related to traditional privacy preservation. Subsequently, many prominent current works for privacy preservation are reviewed.

2.1 Traditional Privacy Preservation

First of all, the well-known database techniques such as security view management [21] can be used to investigate individual privacy problems. It could be easily done by creating different views for different privilege-level users.

It can also be addressed by access-control methods that have been proposed, substantially, among the database security research community [9, 34]. Traditionally, the methods in commercial Database Management Systems (DBMSs) are built on two authorization concepts: positive and negative authorizations. The concept of positive authorizations, which are based on the System R authorization model [25, 32], enforces all required accesses to be specified on any object within the DBMS. When a DBMS user requests access to an object which is already authorized, the user can access the object, otherwise the request will be denied. On the contrary, negative authorizations require the denial access on any object to be specified. Therefore, if a user requests access for an object which is not specified as denied to the user, the access can be performed, otherwise it will be denied. There are many works that propose the combination of these two concepts, and approach the problem when there conflicts exists between the authorization mechanisms [10, 11].

Statistical security-control [22] is another approach for addressing individual privacy preservation problems. Before data is released for any purpose, noise values are added into the original data set to make the data set different from the original data. While some specific statistical values, for instance, the mean or variance, relied on the noise, added data must be maintained as close to the values in the original data as possible.

2.2 Current Privacy-Preservation

Privacy-preserving is an emerging research area [17, 29, 64]. Such research problems consider the privacy of the data to be processed by data processing algorithms. In this section, we review works in the research domain.

Generally, the privacy to be preserved can be categorized into two types, the privacy of the individual and the privacy of the patterns. A common scenario for the problem in the domain context is: a data owner wants to share a data set with a collaborator for a data processing; however, there exist sensitive individual data (or patterns) that can be discovered in the data set. Nevertheless, the data set needs to be shared. Therefore, an algorithm to modify the data set to preserve the privacy may be applied. Meanwhile, the algorithm must also preserve the statistical characteristics of the data set which is to be used by the designate data processing. The characteristics can be referred to as the "usability", of the data set. The usability, in generally, is the information/knowledge left in the modified data set, which can still be discovered.

The approaches to modify, or "transform", the data set to achieve the goals in privacy preservation can be categorized into three main groups as follows.

1. Data perturbation. The modification is achieved by changing the selected value in the data set. For example, in the context of the transactional database, it can be done by changing the value in the transaction from 0 to 1, or 1 to 0. There are a number of works in the context of the association rules mining that address the problem by data perturbation [4, 39, 58]

2. Data reduction. It can also be done by removing some records or transactions completely from the data set. This can be considered as the data set sampling [16].

3. Data blocking. It is a modification that replaces some selected values with a pre-specific "unknown" value [56].

2.2.1 Individual Privacy

In this section, the privacy and usability metrics related to the individual privacy preservation problem in the context of PPDM are reviewed, following by some important works.

In the PPDM context, there are many privacy metrics which represent the privacy of the individuals. For example, the confidence interval is used in [3], where a modified data value can be estimated with a confidence value c that the value lies in the interval $[x_1, x_2]$, then, the width of the interval $(x_2 - x_1)$ represents the privacy at a c confidence level or the differential entropy of a random variable in [2].

Among many of the privacy metrics, an important metric is the k value in k-anonymity model [60]. The k-anonymity model are used in many works, for instance, [1, 28, 35, 37, 43, 59, 66]. Generally, for a given data set, it is k-anonymized data set iff every record in it can not be distinguished from k-1 other records by using pre-specified sensitive attributes; for example, the data set in Figure 2.1 is 2-anonymized data set (adopted from [60]) on Ethnicity, Birth, Gender and Postal Code attributes. When a data set is given, it will be considered whether it is k-anonymized data set. If not, it will be modified until it becomes k-anonymized. Usually, data perturbation and data blocking are used to modify given data sets.

Aside from privacy preservation, algorithms that modify data sets must also preserve the usability. When considering the usability, the problem becomes more complicated, because, in one way, the sensitive data must not be disclosed, while it must be revealed enough to preserve the usability with regard to a data mining purpose. In [43], the authors proved that achieving an optimal anonymization is an NP-hard problem. An example for data usability representation is presented in [33], in which the usability is considered as the quality of the classification results. A modified data set will lose its usability when a record is totally deleted or its class label is not the majority class among same attribute value records when the record is modified.

In [41], the authors propose another privacy metric, ℓ-diversity, which improves the k-anonymity model by preventing homogeneity and background knowledge attacks. The homogeneity attack can occur when the k-anonymized data

Table 2.1: An example of 2-anonymized data set.

No.	Ethnicity	Birth	Gender	Postal Code	Health Problem
1	Black	1965	male	0214*	short breath
2	Black	1965	male	0214*	chest pain
3	Black	1965	female	0213*	hypertension
4	Black	1965	female	0213*	hypertension
5	Black	1964	female	0213*	obesity
6	Black	1964	female	0213*	chest pain
7	White	1964	male	0213*	chest pain
8	White	1964	male	0213*	obesity
9	White	1964	male	0213*	short breath
10	White	1967	male	0213*	chest pain
11	White	1967	male	0213*	chest pain

Table 2.2: An example of 2-diversity data set.

No.	Ethnicity	Birth	Gender	Postal Code	Health Problem
1	Black	1965	male	0214*	short breath
2	Black	1965	male	0214*	chest pain
3	Black	1964-5	female	0213*	hypertension
4	Black	1964-5	female	0213*	hypertension
5	Black	1964-5	female	0213*	obesity
6	Black	1964-5	female	0213*	chest pain
7	White	1964	male	0213*	chest pain
8	White	1964	male	0213*	obesity
9	White	1964-7	male	0213*	short breath
10	White	1964-7	male	0213*	chest pain
11	White	1964-7	male	0213*	chest pain

set has too less diversity though it satisfies the k-anonymity standard. The background knowledge attack is when some background knowledge can be used to guess and narrow down the possibility of the sensitive attribute. For example, the dataset in Table 2.1 does not satisfy the 2-diversity subjected to the second partition. Such partition contains only one sensitive value. On the other hand, the dataset in Table 2.2 is 2-diversity.

Another extension to the k-anonymity model is the (α,k)-anonymity model presented in [68]. This model can also handle the homogeneity attack using the α parameter. The model takes a "severe" sensitive value (pre-specified) as an additional input. Addition to the k-condition, if the fraction between the frequency of the sensitive value and the size of the equivalent class is not higher than α, the dataset is said (α,k)-anonymity. For example, if the dataset in Table

Table 2.3: An example data set.

No.	Job	Birth	Postal Code	Health Problem
1	Lawyer	1965	02141	HIV
2	Artist	1965	02137	HIV
3	Teacher	1964	02132	hypertension
4	Teacher	1964	02131	hypertension
5	Artist	1965	02136	obesity
6	Lawyer	1964	02142	chest pain

Table 2.4: An example data set.

No.	Job	Birth	Postal Code	Health Problem
1	Lawyer	1965	0214*	HIV
2	*	1965	0213*	HIV
3	Teacher	1964	0213*	hypertension
4	Teacher	1964	0213*	hypertension
5	*	1965	0213*	obesity
6	Lawyer	1964	0214*	chest pain

2.3 is given. Suppose that the α parameter is set at 1, and the k parameter is set at 2. In addition, the HIV sensitive value is set as the "severe" sensitive value. The dataset can be transformed to satisfy (α, k) as in Table 2.4. It can be seen that the first and the sixth data tuples form an equivalent class, as well as the second and the fifth data tuples, and the third and the forth data tuples. Thus, the k condition is satisfied. Also, the first equivalent class has 0.5 as its α value (1 tuple with HIV sensitive value in the 2-sized class), as well as the third class. While the second equivalent class has zero α value. Thus, the dataset also satisfies the α condition.

In such paper, the authors presented a proof that achieving the optimal (α,k)-anonymity solution is also as NP-hard as for the standard k-anonymity. The authors also argued that the background knowledge attack should be treated separately and not be included in the model as the ℓ-diversity model.

2.2.2 (k, e)-anonymization

(k, e)-anonymization was first proposed in [54]. As opposed to the generalization-based approach, which the data values are transformed into a more general form of them, the approach begins with data partitioning process subjected to the k distinct values, and the e range. Then, the data within each partition are permuted to protect the privacy. The partitioning phase can help increasing the

Table 2.5: An example dataset

Postal Code	Age	Sex	Salary
50200	35	Male	14,000
50210	36	Male	15,000
50230	40	Male	16,000
50300	41	Female	25,000
50310	43	Female	35,000
50330	47	Male	30,000
50300	53	Male	40,000
50310	54	Female	35,000
50330	58	Male	45,000

Table 2.6: The partitioned dataset

Postal Code	Age	Sex	Salary
50200	35	Male	15,000
50210	36	Male	16,000
50230	40	Male	14,000
50300	41	Female	25,000
50310	43	Female	30,000
50330	47	Male	35,000
50300	53	Male	35,000
50310	54	Female	40,000
50330	58	Male	45,000

utility obtained from the data, since the data transformation is proceeded in the scope of the partition. Comparing with the t-closeness, the (k, e)-anonymization rather aims at optimizing the range of the sensitive data. Also, the aggregate query can be answered effectively with less optimization constraints.

In order to illustrate the effectiveness of the (k, e)-anonymization, consider the dataset in Table 2.5. In the table, a dataset is composed by three non-sensitive attributes, i.e. postal code, age, and sex, of the individuals. The sensitive attribute is the monthly-salary in Thai Baht (THB).

If the k and the e values are set at 3 and 2,000 respectively, a possible solution is shown in Table 2.6.

From Table 2.6, the partition of the dataset is outlined, i.e. there are three partitions. It can be seen that there is at least 3 distinct salary values in each partition, as well as the range of the salary is at least 2,000 THB. The sensitive data have been permuted already. If an aggregate query "find the summation of the salary of the 40-48 years-old" is issued, the result from the permuted dataset is 104,000-106,000 THB (A data tuple from the first partition, all of the

Table 2.7: An example dataset

Postal Code	Age	Sex	Salary
50200	35	Male	11,000
50210	36	Male	12,000
50230	40	Male	13,000
50300	41	Female	15,000
50310	43	Female	15,000
50330	47	Male	16,000
50300	53	Male	16,000
50310	54	Female	18,000

Table 2.8: An example dataset

Postal code	Age	Sex	Salary	Partition ID
50200	35	Male	11,000	1
50210	36	Male	12,000	1
50230	40	Male	13,000	2
50300	41	Female	15,000	2
50310	43	Female	15,000	1
50330	47	Male	16,000	1, 2
50300	53	Male	16,000	1, 2
50310	54	Female	18,000	2

tuples from the second partition, and none from the last partition are involved in the query answering). Meanwhile the same query executed on the original dataset results in 106,000 THB which is very close to the result obtained from the privacy preserved dataset.

In order to utilize the result datasets effectively, the optimum constraints, error, have to be defined. In [54], two error constraints can be defined in two approaches i.e. summation-error and maximum-error. Both of them are to be minimized when the datasets are transformed. The error is difference between the minimum sensitive values and the maximum sensitive values in a partition. Then, the summation-error is the addition of the errors in all the partitions. Meanwhile the maximum error is straightforward, i.e. the maximum error among all the partitions.

In Table 2.6, the optimal solution dataset subjected to minimize the summation-error is shown. For the maximum-error, suppose that a dataset in Table 2.7 is given, and the k value is set at 4, e value is set at 5000. An optimal solution subjected to minimize the maximum-error is shown in Table 2.8. It can be seen that there is overlapping in the solution.

Table 2.9: The partitioned dataset

PID	Hits	Sum LB	Sum UB	Min LB	Min UB	Max LB	Max UB
1	1	14,000	16,000	14,000	16,000	14,000	16,000
1	2	29,000	31,000	14,000	15,000	15,000	16,000
1	3	45,000	45,000	14,000	14,000	16,000	16,000
2	1	25,000	35,000	25,000	35,000	25,000	35,000
2	2	55,000	65,000	25,000	30,000	30,000	35,000
2	3	90,000	90,000	25,000	25,000	35,000	35,000
3	1	35,000	45,000	35,000	45,000	35,000	45,000
3	2	75,000	85,000	35,000	40,000	40,000	45,000
3	3	120,000	120,000	35,000	35,000	45,000	45,000

The complexity of the algorithm generated the optimal solutions for the summation-error problem is $O(n^2)$ where n is the number of data tuples in the given dataset [54]. Such algorithm composes of two phases, i.e. partitioning the given dataset according to (k, e)-anonymization, as well as permuting the sensitive values within each partition. The partitioning phase, which costs $O(n^2)$, is our focus in this paper.

In order to improve the efficiency of the aggregation query answering, the authors in [54], proposed an auxiliary structure that stores the summation lower/upper bounds, minimum lower/upper bounds, or the maximum lower/upper bounds for each partition. Such structure can help answering after the number of data tuples for the query is determined. For example, if the resulted dataset in Table 2.6 is used to answer the aggregation queries, the auxiliary structure in Table 2.9 can be built. Suppose that the query is to determine the minimum salary of females, it can be seen that two data tuples from the second partition and one data tuple from the third partition are involved in the answer. From the number of hit data tuples in the structure, it can see that the lower bound of the second partition with regard to the minimum is 25,000, while the upper bound is 30,000. For the third partition, the lower bound is 35,000, and the upper bound is 45,000. Thus, the answer is between 25,000 to 35,000 THB.

In [53], a 2-approximation algorithm for (k, e)-anonymization subjected to minimize sum-of- error problem is proposed. If the dataset is already sorted on the sensitive attribute, the algorithm requires only $O(n)$ complexity (where n is the number of data tuples). For the unsorted datasets, the complexity of the algorithm is bounded by the sorting cost, i.e. $O(n\ log\ n)$. For the problem of minimizing maximum-error, in the same work, the authors proposed a 2-approximation algorithm with $O(n\ log\ n)$ cost, as well as a dynamic algorithm

that computes the optimal solution.

2.2.3 Incremental Privacy Preservation

For the incremental privacy preservation, a few studies have been reported [13, 30, 52, 70]. Although the incremental algorithms without re-processing the whole datasets have been proposed, these algorithms deal with the situations where multiple versions of the datasets are released. The different versions may have different levels of anonymization e.g. different generalized or shuffling values from different privacy parameters. Unlike our work, we deal with only a single version of the dataset with a single level of privacy parameters (pre-specified by the responsible agencies e.g. the Ministry of public health in Thailand as mentioned above). It may be released publicly multiple times, however, joining them do not give the attacker any additional knowledge.

Additionally, an approach to preserve the privacy in data streams has been proposed in [73]. The approach focuses on transaction streams where the data of an individual may have been stored multiple times in a dataset. To deal with the continuous nature of the data streams, the lists of equivalence classes are kept and maintained for the incremental stream processing.

2.3 Sensitive Pattern Hiding

In the PPDM research domain, not only is the privacy problem of individuality concerned, but also the privacy threat from sensitive patterns. We present the first motivating example from [19] as follows: suppose that a large supermarket, BigMart, negotiates on a business matter with a paper-production company, called Dedtrees. Dedtrees offers a reduced price on their product if BigMart allows them to access the selling database. By using an association-rule mining algorithm, suppose that Dedtrees finds that customers who buy skim milk also buy Green paper. Then, Dedtrees starts the campaign "get 50 cents off skim milk with every purchase of a Dedtrees's product". This affects the sales of Green paper enormously. Finally, Dedtrees becomes a bigger company in the paper market, and BigMart has lost the competitive edge.

In [27], the authors presented a motivating example when a rule **(PostCode = 5409) $\wedge$ (Age = 18 to 25) $\wedge$ (Gender = Male) $\rightarrow$ HepBStatus = Yes** is discovered from a released data set; suppose that the postal code 5409 referred to an indigenous community or the national parliament. This rule can

be considered as an offense to the population in the area and should be hidden before the data set can be shared or released. By empirical studying, the same authors also presented an approach to justify which rule can be considered as the sensitive rule.

Another example requiring sensitive patterns to be hidden is presented in [6]. The motivating example was presented in the paper as follows: Consider an association rule $a_1 \wedge a_2 \wedge a_3 \rightarrow a_4$ with 80 supporting records and 98.7% confidence. From this information, we can derive the support of the antecedence as: $sup(a_1, a_2, a_3) = \frac{sup(a_1,a_2,a_3,a_4)}{conf(a_1,a_2,a_3 \rightarrow a_4)} = \frac{80}{0.987} = 81.05$, where $sup(x)$ is the number of supporting records of the itemset x and $conf(x \rightarrow y)$ is $sup(x \cup y)$ divided by $sup(x)$. We can infer that there is one individual record that has the support of $a_1 \wedge a_2 \wedge a_3 \neg a_4$ which can be considered as a privacy violation or a threat on anonymity. In this paper, the authors presented a method to detect sensitive frequent itemsets which violate the anonymity. Another work by the same authors proposed a method to prevent the disclosure of the sensitive frequent itemsets [5].

From the above examples, the need to hide the sensitive patterns can be seen. To hide such sensitive patterns in data sharing scenarios, data must be modified until the interestingness of the sensitive patterns fall below the specific threshold. This process of making the patterns disappear is referred as "pattern hiding".

When an optimal solution is required, the sensitive pattern hiding problem is proven as an NP-hard problem [4]. The proof is shown by reducing the problem to a HITTING SET problem, that is, given a set C of subsets of a finite set S, find the smallest subset S' of S such that every subset in C contains at least one element in S'. In the problem reduction, (1) the condition that the sensitive large itemsets must be hidden is inferred to in the condition that every subset in C contains at least one element in S', (2) the minimal side effect is inferred to in the smallest subset S'.

Because of the proof, there are many heuristic algorithms proposed to address the problem in the context of the association rules hiding. In [65], the authors presented many heuristic ways to modify the data set to address the problem in the association rule mining context. The proposed modification is achieved by data perturbation. The selected values in the data set will be perturbed to decrease the support and/or the confident values of the sensitive rules. The rules will be hidden successfully if their support and/or confident values are less than the specific thresholds. The authors proposed to hide sensitive

association rules through the two following options, (1) decrease the confidence of the rule, and (2) decrease the support of the rule. In the first option, the authors presented the analysis of the association rules's confidence formulation, that is, $\frac{|X \cup Y| \times 100}{|X|}$ for a rule $|X \rightarrow Y|$. Then, the authors suggested two ways to decrease the confidence; first, perturb the item Y from 1 to 0 in the transactions which partially support the rule $X \rightarrow Y$ to decrease the numerator part of formulation ($|X \cup Y|$). This will fix the value in denumerator part of the formulation ($|X|$). The other way it can be done is by increasing the value of the denumerator of the formulation which will make the confidence value decrease. It can be done by perturbing the item X from 0 to 1 in the partially supported transactions of the rule. The second option is achieved by perturbing the item X or either Y from 1 to 0 to decrease the support of the rule $X \rightarrow Y$. The authors also presented the experiment results of these proposed ways.

Oliveira and Zaïane proposed several heuristic algorithms to hide sensitive frequent itemsets [47, 49] in the context of the association rule mining. To boost the sensitive patterns hiding process, the authors proposed to apply the inverted index files [7]. Generally, after the data owner specifies sensitive patterns, the sensitive transactions with regard to the sensitive patterns must be identified for further modification. Using the inverted index file can help to identify the transactions efficiently. When the transactions have been identified, the algorithms will select the *victim* items and perturb them by changing the values from 1 to 0. The selection of the victim items is different among the three proposed algorithms of the Minimal Frequency Item, the Maximal Frequency Item, and the Item Grouping algorithms. The authors also introduced the notion of the *degree of conflict* of a transaction which supports the sensitive frequent itemsets. For a transaction, its degree of conflict refers to the number of sensitive patterns it supports. It is suggested by the authors that, to hide a sensitive pattern, modifying the transaction with a high degree of conflict can reduce the side effect. Also, the disclosure threshold ψ for the sensitive patterns is introduced; its value ranges from 0 to 1 and must be specified by the data owner. When it is set to 0, it means that the algorithm must hide the sensitive patterns until they can not be discovered, where, in the context of frequent itemsets, the support of sensitive patterns must be below the support threshold.

The algorithms proposed by the authors composed of four steps. First, the algorithms identify the transactions which support the sensitive patterns. Second, the victim items in the identified transactions are selected for perturbation by different criteria. In the Minimal Frequency Item algorithm, items which

have minimal support values are selected to be removed. While the Maximal Frequency Item algorithm removes the item that has maximal support value. In the last algorithm, the Item Grouping algorithm, tries to select the item that is common among sensitive frequent itemsets, then, removing the item can help hide many sensitive frequent itemsets at once. Third, the number of transactions to be perturbed is determined by the disclosure threshold ψ. The last step is the actual perturbation. For each sensitive pattern, it begins with the sorting of the supporting transactions of the pattern by the degree of conflict. Then, it perturbs a numbers of transactions from the third step by changing the item value from 1 to 0 in the victim items. By the same authors, the issue of balancing between the sensitive patterns privacy and the non-sensitive patterns disclosure is also addressed [48].

A border-based algorithm to hide the sensitive frequent itemsets is proposed by Sun and Yu in [58]. In the frequent itemset context, they introduced a new usability measurement, the relative frequency of remaining non-sensitive frequent itemsets, which, instead of counting the number of non-sensitive rules left in the modified data set as the side effect, the new measurement represents more details about the usability. This measurement is proposed to measure the aggregated quality of the modified data sets. Based on the concept of the border of itemsets [42], the algorithm was proposed to hide the lower border of the sensitive itemsets, instead of hiding every sensitive itemset. In [44], another approach, the Min-Max, which is based on border-based concept is proposed. Given many sensitive itemsets, the authors suggest hiding the sensitive itemsets with minimum support first, because they are the closest to the borders. Then, among the sensitive minimal support itemsets, the highest (or maximum) support itemsets will be selected. From such highest support itemsets, the victims item to be modified can be selected.

In [69], the authors present a different view on the association rule hiding. Instead of hiding the sensitive association rules and minimizing the side effect (ghost rules and false-drop rules), the authors suggested that the side effect can damage the applicability of the modified data set when it uses, for example, the ghost rules in the medical domain; this can lead to the wrong medical treatment suggestion. An approach is proposed to modify the data set such that there is no side effect and to minimize the number of disclosed sensitive rules. The experiment's results showed that the sensitive rules can be hidden successfully, mostly by their approach. Instead of the values changing, there is also another heuristic method to replace the selected values with unknown

values [56]. This approach hides sensitive patterns by introducing margins of support and confident values to some extent of uncertainty.

An inference problem with regard to the classification mining is presented in [67]. Instead of sensitive rule hiding, the authors address a problem of the blocking inference channel in the form $< IC \rightarrow \pi,\ h >$, where IC is a set of attributes, π is a class label, and h is a confidence threshold, for example $< \{Poscode, Age, Gender\} \rightarrow HepBStatus = Yes,\ 75\% >$. The authors also presented an algorithm to block the inference channel by modifying the data in top-down basis, that is, a sensitive attribute value will first be transformed into the most general value, then transformed into a more specific value when the algorithm proceeds further. The sensitive pattern hiding problem can be considered as a more specific problem addressed by this work.

2.4 Other Related Work

Other works focusing on privacy representation are as follows.

In [3], the authors address the problem of the decision tree-based classifiers building from perturbed data sets by adding noise. The authors propose to use the reconstruction approach, that is, perturbing the given data set and reconstructing the aggregate information of the data set to help in mining the data. The data set must be perturbed such that the difference between the perturbed and the original data values can guarantee the privacy of the data. The usability in this work is represented as the accuracy of the decision-tree which is discovered from the modified data set. When the perturbed data set is used for classification model building, the reconstruction procedure is continued to estimate the data distribution of the original data set. Subsequently, the new decision tree-based classification model is built with the comparable accuracy of the original data set.

In [2], the Expectation Maximization (EM) technique is applied to the problem for improving the distribution reconstruction. In [24], the approach to address the same problem with binary and category data sets are proposed respectively. In [72], the way for perturbation the data set is guided by the data mining algorithm aspect. The authors suggested that the proposed algebraic approach can provide data mining result more accurate, and also disclose less sensitive information. Note here that in these works, the data mining algorithms to be used with the released data need to accept the different input data, that

is, instead of using only released data sets to derive knowledge, the algorithms must be modified to use perturbed data sets plus data distributions from the estimation process.

In [51], the approach to preserve the individual privacy for data clustering is presented. The approach transforms the given data set into dissimilar matrix before the data sharing takes place then, the clustering is performed by a dissimilar matrix. The authors also proposed a dimensional reduction method, random projection to improve efficiency of the approach. Another work by the same authors in [50] presents a different approach. The rotation transformation [18], which is an isometric geometry transformation, is applied to address the problem. The transformation is performed in the data set as each selected attribute-value pair is transformed by the transformation matrix, such that the "variance" from the transformed values and the original values are higher than the minimal security thresholds.

In [16], Clifton addressed the individual privacy problem in classification computing. Given a data set, the author proposed the approach to sample the data such that the privacy of individual can be preserved. The approach can determine the size of the released data set.

Secure Multi-party Computing

When collaborative parties want to compute a global computation from the integrated data by preventing each party from learning the other's data, the problem can be formulated into secure multi-party computing (SMC) [8, 15, 31]. This is a individual privacy problem in the PPDM. This problem can be considered as the generalized problem of secure two-party computation [71] which was addressed in the cryptography research area at its early stage. There are many different computations which can be performed by solutions of this problem; for example, the problem of the global decision tree is covered in [40]. Since each party has a different part of the global data, the way each party holds their data can be categorized into two groups: horizontal and vertical partitioning. In [62], the global association rules are computed from the parties, where each party has different items to be considered. This work is referred to as the vertically-partitioned association rule mining. On the other hand, horizontally-partitioned association rules mining is presented in [36], in which each party has a different part of the data set, but the same items.

Inference Problem

Data downgrading problem [14] or inference problem [20] is another important problem for individual privacy. The scenario is as follows. A data owner wants to release "some part of a data set" to the public, that is, there are some records which have their values blocked for the privacy preservation reason. However, the released part of the data may be used to infer the sensitive part of the data. In [14], the formal analysis of such situations and the thermodynamic approach to balance privacy against usability are purposed. A relationship between sensitive information and the size of the released data sets is studied in [16].

Chapter 3

Data Increment on (k, e)-Anonymization

3.1 Problem Definition

In this chapter, we define the notations and terms, and subsequently the focused problem.

Definition 1 (Dataset) *Let a dataset* $DID = \{d_1, d_2, \ldots, d_n\}$ *be a collection of tuples defined on a schema* $\boldsymbol{A}$*, which consists of an identifier* ID*, a set of quasi-identifiers* $QI = \{qi_1, qi_2, \ldots, qi_k\}$*, and the numerical sensitive attribute* S*. For each* $d_i \in DID$*, the value of identifier attribute* ID*, a quasi-identifier attribute* qi_j*, as well as the sensitive attribute* S *is denoted as* $d_i.ID$*,* $d_i.qi_j$*, and* $d_i.S$ *respectively. For each attribute* $A_j \in \boldsymbol{A}$*, its domain is denoted as* $dom(A_j)$*.*

Suppose that the given schema is as Figure 3.1 is given. An example dataset

Figure 3.1: An example schema

Citizen ID	Postal code	Age	Sex	Salary

of such schema can be as in Table 3.1. The Citizen ID is obviously the identifier of the dataset. The set of quasi-identifiers is, QI = {Postal code, Age, Sex}. And, suppose that the sensitive value is Salary attribute.

Table 3.1: An example dataset

Citizen ID	Postal code	Age	Sex	Salary
357xx567	50200	35	Male	14,000
357xx111	50210	36	Male	15,000
357xx113	50230	40	Male	16,000
357xx565	50300	41	Female	25,000
357xx446	50310	43	Female	35,000
357xx448	50330	47	Male	30,000
257xx567	50300	53	Male	40,000
257xx568	50310	54	Female	35,000
257xx167	50330	58	Male	45,000

We consider to transform the datasets without identifiers which is defined as follows.

Definition 2 (De-identified Dataset) *The de-identified version of a dataset DID is a projection of it over the set of quasi-identifiers and the sensitive attribute, denoted as D.*

The de-identified dataset for the dataset in Table 3.1 is shown in Table 3.2. Such de-identified datasets can have an overlapped public dataset on some quasi-identifier attributes. The public dataset also contains the identifiers which lead to the privacy breach. Our goal here is to transform the de-identified datasets to break such link between the quasi-identifier attributes and the sensitive value such that one who owns the public datasets cannot match them to the trans-

Table 3.2: An example dataset

Postal Code	Age	Sex	Salary
50200	35	Male	14,000
50210	36	Male	15,000
50230	40	Male	16,000
50300	41	Female	25,000
50310	43	Female	35,000
50330	47	Male	30,000
50300	53	Male	40,000
50310	54	Female	35,000
50330	58	Male	45,000

Table 3.3: The partitioned dataset

Postal Code	Age	Sex	Salary
50200	35	Male	14,000
50210	36	Male	15,000
50230	40	Male	16,000
50300	41	Female	25,000
50310	43	Female	35,000
50330	47	Male	30,000
50300	53	Male	40,000
50310	54	Female	35,000
50330	58	Male	45,000

formed dataset, while the aggregate queries can still be effectively answered.

Definition 3 (Aggregate Queries) *A aggregate query q is an SQL-query which applies the aggregate functions including COUNT, MAX, MIN, SUM, and AVERAGE on the sensitive attribute of the de-identified datasets.*

Definition 4 ((k, e)-partition) *A de-identified dataset D is satisfied a (k, e)-partition iff for each partition $P_i \subseteq D$, the projection over the sensitive attribute S provides at least k distinct values and the range of these different values in P_i is at least e.*

Definition 5 ((k, e)-anonymity permutation) *Let p be a random permutation over the tuple identifier $\{1, 2, \ldots, n\}$. The permutation of a de-identified D denoted as $p(D, QI, S) = \{d^{i'} | \forall qi_j \in QI, d'_i.qi_j = d_i.qi_j$ and $d_{i'}.S = d_{p(i)}.S\}$ Given a random permutation p and a de-identified (k, e)-anonymity dataset D, the (k, e)-anonymity permutation of D is $p(D, QI, S)$.*

As mentioned before, there could be many (k, e)-anonymity permutation datasets, or (k, e)-anonymity for short, for a given data, the transformation algorithm should select the optimal-utility dataset, i.e. the dataset which has the least error. Such error can be defined in two approaches [54], i.e. summation-error and maximum-error. We focus on the former error definition defined as follows.

Definition 6 (Error) *Let a set of partitions $\{P_1, P_2, \ldots, P_m\}$ be a partitioning of a dataset D resulted from the (k, e)-permutation transformation. Let the $error(P_i)$ be the error occurs in a partition P_i, $error(P_i) = max(S_i) - min(S_i)$,*

Table 3.4: The partitioned dataset

Postal Code	Age	Sex	Salary
50200	35	Male	15,000
50210	36	Male	16,000
50230	40	Male	14,000
50300	41	Female	25,000
50310	43	Female	30,000
50330	47	Male	35,000
50300	53	Male	35,000
50310	54	Female	40,000
50330	58	Male	45,000

where max and min is the function to determine the maximum and minimum sensitive value from P_i respectively.

The ***summation-error*** *of the dataset D from the (k, e)-permutation is* $\sum_{i=1}^{m} error(P_i)$.

Precisely, the problem is to determine the $p(D, QI, S)$ which has minimal **summation-error** given the dataset D, the k, and e parameters.

3.2 Effect of Data Increment

First, the $O(n^2)$ for preserving the privacy with regard to the (k, e)-anonymization is presented in Figure 3.2. The algorithm will compute the optimal summation error in each i-loop. The error value is kept in the *error* variable. That is, the $error[D.size]$ will contain the summation error of the optimal solution. The approach of the algorithm is a greedy which it can be seen in the d loop. The algorithm will compare the "valid" current error considering the k and e values, to the best-known error for such d_i data tuple. If the current error is less than the exiting value, it is stored as the new error in $error[i]$ value. Also, the index of the data tuple d_j which cause the less error is stored in $partition[i]$ value. This comparison will contribute to the sub loop $O(n)$ for each i loop. And, thus the complexity of the algorithm becomes $O(n^2)$.

Table 3.5: An example sensitive dataset to demonstrate the $O(n^2)$

Id	Salary
1	54
2	55
3	56
4	65
5	70
6	75
7	75
8	80
9	85

Table 3.6: The *error* and *partition* of the example sensitive dataset to demonstrate the $O(n^2)$

Id	*error*	*partition*
1	Infinity	0
2	Infinity	0
3	2	1
4	11	1
5	16	1
6	12	4
7	12	4
8	17	4
9	22	7

Suppose that the sensitive values of a dataset is as shown in Table 3.5 with the k and e values are set at 3 and 2 respectively. When the algorithm in Figure 3.2 have been applied to the dataset its *error* and *partition* values are as shown in Table 3.6. The partition information can be obtained by scan the *partition* variable backward, thus the partition of data tuples represented the Id as {1, 2, 3}, {4, 5, 6}, and {7, 8, 9}.

Input:
D: a dataset
k: a minimum number of distinct values threshold
e: a minimum range of values threshold
Output:
(D': the output dataset, which satisfies the (k, e)-anonymization
partition: the partition information
error: the error information
Method:

```
error[0] =infinity
partition[0] =0
for i = 1 to D.size
  error[i] = infinity
  partition[i] = partition[i − 1]
  for j = 1 to i
    if distinct({d_j, ..., d_i}) ≥ and (d_i − d_j) ≥ e then
      current_error = (d_i − d_j)
    else
      current_error = infinity
    end if
    temp = f[j − 1] + current_error
    if temp ≤ error[i] then
      error[i] = temp
      partition[i] = j
    end if
  end for
end for
```

Figure 3.2: Original $O(n^2)$-Algorithm

Table 3.7: The impact when a data tuple with 67 as its sensitive value is inserted

Id	Data	*error*	*error*′	*partition*	*partition*′
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
4	65	11	11	1	1
-1	67	-	13	-	1
5	70	16	7	1	4
6	75	12	12	4	4
7	75	12	12	4	4
8	80	17	17	4	4
9	85	22	17	7	7

Table 3.8: The impact when a data tuple with 55 as its sensitive value is inserted

Id	Data	*error*	*error′*	*partition*	*partition′*
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
-1	55	-	Infinity	-	0
3	56	2	2	1	1
4	65	11	11	1	1
5	70	16	16	1	1
6	75	12	12	4	5
7	75	12	12	4	5
8	80	17	17	4	5
9	85	22	22	7	8

When an incremental data tuple $\triangle d$ is appended to the dataset, the naïve approach is to execute the algorithm again. However, we will address the incremental data transformation problem in a more efficient way.

Suppose that the data tuple with 67 as its sensitive value is inserted into the dataset in Table 3.5. Let's assume that the Id is minus one for the insertion, to be able to see its impact. First, we present the *error* and *partition* values in Table 3.7. Note that we present the data and the previous values to compare with the impact as well. Such increment will change the partition structure into {1, 2, 3}, {4, -1, 5}, and {6, 7, 8, 9} and the summation-error is decreased to 17.

It can be seen that the increment can change summation error as well as the partition structure, however it might be not always the case. We present another example where the increment will not change the error. Re-consider the dataset shown in Table 3.5, suppose that the data tuple with 55 as its sensitive value is inserted. The *error* and the *partition* values of such increment is shown in Table 3.8. In which, the summation-error is the same as the previous value. Also, the structure of the new partition is "the same" though inside the first partition will have more element.

Thus, we analyze the impact of the increment theoretically as follows. First, we categorize the changes into two main groups. The first group is when the increment data is inserted at the "in-range" of the existing partition. The second group is when the increment data is inserted at the border of the two existing partitions. We discuss the details of each group in the next sections. Subsequently, the approach to compute the impact of the increment is presented.

Table 3.9: The impact when a data tuple with 76 as its sensitive value is inserted

Id	Data	*error*	*error*′	*partition*	*partition*′
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
4	65	11	11	1	1
5	70	16	16	1	1
6	75	12	12	4	4
7	75	12	12	4	4
-1	76	-	13	-	4
8	80	17	17	4	7
9	85	22	21	7	8

3.2.1 In-range Increment

For the "in-range" increment, when an data tuple $\triangle d$ is inserted into a partition Pi, which is a partition in an optimal solution. An impact could be the partition Pi', the partition Pi after insertion identified by the first/last data tuple in it, can be broken into a complete partition satisfying the (k, e)-condition before the ending/beginning tuples of the old Pi. The left out data tuples will be merged and considered with the consecutive partition Pj, which is preceded by Pi or next to Pi. In this case, the optimal summation-error can be reduced, since the partition can be completed before the position it was.

For example, let us re-consider the dataset in Table 3.7. The data tuple with 67 as its sensitive tuple is inserted. The partition such tuple is inserted to is the second partition, it had previously {4, 5, 6} data tuples with the summation-error at 10. After the insertion, the partition is broken before the 6-data tuple is considered. The modified partition becomes {4, -1, 5} with the summation-error at 5. Consecutively, the 6-data tuple is merged with last partition, however it does not increase the summation-error since it is duplicate with a data tuple within the partition. Overall, the insertion causes the summation-error reduce from 22 to 17.

Table 3.10: The impact when a data tuple with 56 as its sensitive value is inserted

Id	Data	*error*	*error′*	*partition*	*partition′*
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
-1	56	-	2	-	1
3	57	3	3	1	1
4	65	11	11	1	1
5	70	16	15	1	4
6	75	13	13	4	5
7	75	13	13	4	5
8	80	18	18	4	5
9	85	23	23	7	8

Another example is shown in Table 3.9. The scenario is a data tuple with 76 as its sensitive value is inserted to the existing dataset. The partition structure has become {1, 2, 3}, {4, 5, 6, 7}, and {-1, 8, 9}. The summation-error reduce from 22 to 21, and the last partition is completed before the 7-data tuple is considered.

In some cases, the "in-range" increment can also leave the summation-error intact. For example, let us consider a dataset in Table 3.10. Note that the existing dataset is different from the previous examples. It can be seen that the insertion does not cause the ending data tuples (3-data tuple, in this case) to be merged with the consecutive partition. Because the merging can increase the summation-error with regard to the optimal solution. Instead, it keeps the beginning and the ending of the existing dataset without increasing the error.

From the general point of view, we can further consider the details of such impact as follows. The $O(n^2)$ optimal algorithm in Figure 3.2 computes the *error* of each data tuple d_i by considering its partition-break-point in greedy approach. For example, from the dataset in Table 3.10 (ignoring the increment) when the last data tuple (10-data tuple) is considered, its *current_error* is firstly set at $infinity$. Then, the partition of $\{1, 2, \ldots, 9\}$ is considered for its error, and its *current_error* is decreased to 31. Not all possible partitioning can satisfy the (k, e)-condition, but some of them, e.g. when a break point providing the partition $\{1, 2, -1\}$ and $\{4, 5, \ldots, 9\}$ as i is set at 10, and j is set at 4, the *current_error* is reduced to 28, while the summation-error is reduced to 30 (the previous optimal error is, $f[j-1]$, is 2). Last, when the j variable is set at 7, the *current_error* is reduced to 10, providing the summation-error at 23.

Table 3.11: The impact when a data tuple with 57 as its sensitive value is inserted

Id	Data	*error*	*error′*	*partition*	*partition′*
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
-1	57	-	3	-	1
4	65	11	11	1	1
5	70	16	15	1	4
6	75	12	13	4	5
7	75	12	13	4	5
8	80	17	18	4	5
9	85	22	23	7	8

When a new valid break-point satisfying the (k, e)-condition is found, not only the *error* is updated, but the *partition*. Such variable will be updated to be the valid break-point in the greedy manner. All break-point values might not be used to construct the partition structure. These are the values up to the current data tuples since the greedy processing, the last value will provide the actual structure. Thus, the partition structure can be obtained by scanning the *partition* backward.

3.2.2 Border Increment

For the "border" increment, when an data tuple $\triangle d$ is inserted between a preceding partition Pi and another consecutive Pj, both partitions obtained from an optimal solution. One of the possible impact is that partition Pi merges such $\triangle d$ into it. As a result the structure of partition Pj and the rest are not effected. In this case, the optimal summation-error is to be increased since the preceding partition is extended.

For example, consider the dataset in Table 3.11. It can be seen that a data tuple with 57 as its sensitive value is inserted between the first and the second partitions, i.e. {1, 2, 3} and {4, 5, 6} respectively. The optimal result after the insertion is that the data tuple is merged to the first partition. The other partitions have no impact from such insertion. As a result, we obtain the optimal partition as {1, 2, 3, -1}, {4, 5, 6}, and {7, 8, 9}. Also the summation-error is increased from 22 to 23.

Table 3.12: The impact when a data tuple with 64 as its sensitive value is inserted

Id	Data	*error*	*error′*	*partition*	*partition′*
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
-1	64	-	10	-	1
4	65	11	11	1	1
5	70	16	8	1	4
6	75	12	13	4	4
7	77	14	15	4	4
8	77	14	15	4	4
9	101	38	34	4	7
10	102	37	35	7	7
11	103	16	17	9	10

When an increment data tuple is inserted at the border, it can also be merged with the consecutive partition Pj, and the rest of the next partitions subjected to Pj are remained the same. This causes the increasing of the summation-error by considering the margin of the insertion data tuple and Pj as shown in Table 3.12. Note that the dataset in this example is different from the previous.

For the cases where the $\triangle d$ is merged with partition Pj, it can cause such partition breaks into a complete partition satisfying the (k, e)-condition before the ending tuples of the old Pj. Thee left out data tuples will be merged and considered with the consecutive partition of Pj. This will happen only when the cost of the break is less than merging and keeping the structure of Pj the same as before. In order to illustrate this case, consider another example in Table 4.2. From the example, it can be seen that Pj, in this case {4, 5, 6, 7 8}, is broken by merging $\triangle d$ to its front-end. Then the partitions become {-1, 4, 5} and {6, 7, 8} with less summation-error.

From the above observation, we can see that the impact of an increment is as follows. First, when the position of the insertion data tuple is determined. The *error* and the *partition* values priori to the position are always remained intact due to the greedy approach. Then, the *error* and the *partition* values of such insertion values can be determined. For example, the *error* and *partitioning* values of the dataset in Table 3.9 before the insertion position is remained the same. While the values at the insertion as well as the rest can be changed.

When considering the details of the partitioning, we can see that the insertion at the partition border can only merge with the preceding and the consecutive

Table 3.13: The impact when a data tuple with 64 as its sensitive value is inserted, and a new partition is found

Id	Data	*error*	*error*′	*partition*	*partition*′
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
-1	64	-	8	-	1
4	65	11	11	1	1
5	70	16	10	1	4
6	75	12	15	4	4
7	76	13	16	4	4
8	80	17	15	4	7
9	101	37	36	7	7
10	102	35	37	8	7
11	103	19	17	9	7
12	120	36	34	9	10
13	121	37	35	9	10
14	122	21	19	12	13

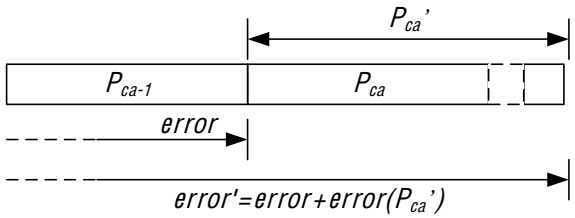
partition. If the data tuple is to be merged with the preceding partition, it is from the fact the error from extending the partition is less than the error causes by merging it with the consecutive partition. Though, merging $\triangle d$ data tuple to the consecutive partition can also causes such partition to meet the break-point before the previous since the (k, e)-condition is met. The structure-change of the insertion partition is depended on the *error* value of the data tuples which is processed in greedy manner.

3.3 Incremental Processing

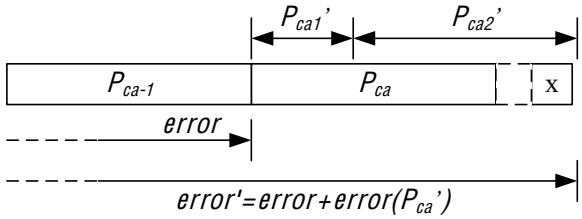
A naïve approach is that computing the *error* and the *partition* values right from the insertion position to the end. However, consider the cases where the insertion position is in the very beginning position of the existing dataset, such approach might not be appropriated. As such the complexity of the incremental algorithm in this case will be $O(n'^2)$ where n' is the number of the rest data tuples to be processed.

In this work, we propose a lemma which will be used to improve the incremental processing as follows.

Lemma 1 *For each incremented dataset D', after the completed-alignment partition, the partition in which the data tuples in it is the same as before the in-*



a) Extending partition



b) Breaking partition

Figure 3.3: Summation-error Computing for Incremental Processing

sertion, the remaining structure of the partition is the same as priori to the insertion.

Proof Let d_i be a data tuple after the completed-alignment partition p_{ca}, the partition in which the data tuples in it is the same as before the insertion. In order to obtain the optimal solution, the $error[i]$ of d_i can be only computed in two cases since partition p_{ca} has not been effected by the insertion. First, it is merged to p_{ca} or one of p_{ca} consecutive partitions, or it is combined with the other $d_{i-1}, d_{i-2}, \ldots d_{i-m}$ and forms a new partition. The both cases, $error[i]$ is the summation between the $error[partition[ca] - 1]$, and the difference between the sensitive value of d_i and the sensitive value of the first data tuple in $partition_{ca}$, or $error[i] = error[partition[ca] - 1] + (d[i] - d[partition[ca] - 1])$.

In Figure 3.3a), the illustration of the first cases is shown. It can be seen that the range of the partition p_{ca} which adds the new data tuple d_i is extended. Thus, its error is formed by the summation-error before p_{ca} and the incremented p_{ca}. The latter cases where a new partition is formed is illustrated in Figure 3.3b). Such error computation can be performed in the same approach.

Also, as the structure of the partition is not change after the complete-

Table 3.14: The impact when a data tuple with 64 as its sensitive value is inserted, and a new partition is found

Id	Data	*error*	*error*′	*partition*	*partition*′
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
-1	64	-	8	-	1
4	65	11	11	1	1
5	70	16	10	1	4
6	75	12	15	4	4
7	76	13	16	4	4
8	80	17	15	4	7
9	101	37	36	7	7
10	102	35	37	8	7
11	103	19	17	9	7
12	120	36	34	9	10
13	121	37	35	9	10
14	122	21	19	12	13

alignment partition, we can further update the *error* and *partition* variables as follows. First, the *partition* variable after such point is increasing by one. Since the positions of the remaining is shifted by one insertion. Then, the *error* partition can be computed as mentioned.

For example, suppose that the dataset in Table 3.14 is given, and the k value is set at 3, and the e value is set at 2. Suppose further that a data tuple with 64 as its sensitive value is appended. It can be seen that the complete-alignment partition is the partition with {9, 10, 11} data tuples. The remaining partition information of the remaining tuples is computed by adding one. Then, the $error[12]$ is the addition of the $error[partition[ca] - 1]$, i.e. 17, to $(120 - d[partition[ca] - 1])$, i.e. 120 - 103. Thus, such error is 34.

From such observation, we can develop an incremental algorithm which computes the *error* and the *partition* as the static algorithm until the complete alignment is found. Subsequently, the *partition* is computed with $O(n')$ where n' is the number of the remaining data tuples to process. As well as the *partition*, the *error* can be computed with the same cost. Such algorithm is to be presented in the next chapter.

Chapter 4

Incremental Algorithm

After we propose a lemma to improve the efficiency of the data incremental in the previous chapter. That is, if the partition in which the data tuples in it is the same as before the insertion is founded, the remaining structure of the partition is the same as priori to the insertion. In this chapter, we propose an incremental algorithm to preserve the privacy based on the (k, e)-anonymization based on such lemma as shown in Figure 4.1.

From the algorithm, it can be seen that the efficiency depends on how far the break-point position can be found. Let n be the number of data tuples, let *ins_pos* be the position of the insertion, and let n' be the number of the remaining data tuples after the break-point (In Line 20 of Figure 4.1). The computational complexity of the proposed algorithm is composed by the cost of the insertion determination, finding the optimal solution before the break-point, as well as *error* and *partition* determination after the breakpoint. Thus, the complexity is $O(ins\_pos)+O((n' - ins\_pos)^2)+O(n - n')$. Obviously, if the break-point is found at the end of the dataset, the incremental algorithm will have the same complexity as the non-incremental algorithm, otherwise it can be more efficient.

To clarify our algorithm, let us consider an example. Suppose that the dataset in Table 4.1 is given. When a data tuple with 64 as its sensitive value is inserted, the algorithm first determines that the position of the $\triangle d$, i.e. the fourth position as shown in Table 4.2. The $partition'$ and the $error'$ values of the preceding position are the same as their original values, i.e. 0, 0, 1, and Infinity, Infinity, 2 respectively. Subsequently, the i-loop is proceeded to determine the optimal solution for each d_i as the non-incremental algorithm.

Input:
D': a permuted dataset
k: a minimum number of distinct values threshold
e: a minimum range of values threshold
$\triangle d$: an additional tuple
partition: a partition information from the permuted dataset
error: an error information from the permuted dataset
Output:
$(D + \triangle D)'$: the output dataset, which satisfies the (k, e)-anonymization
partition$'$: the updated partition information
error$'$: the updated error information
Method:

```
1   insert △d into D' based on its sensitive value
2   determine the position of △d, ins_pos
3   copy partition and error into partition' and error' from the first to the position of the ins_pos
4   for i = ins_pos to (D + △D)'.size
5     error'[i] = infinity
6     partition'[i] = partition'[i − 1]
7     for j = 1 to i
8       if distinct({d_j, ..., d_i}) ≥ k and (d_i − d_j) ≥ e then
9         current_error = (d_i − d_j)
10      else
11        current_error = infinity
12      end if
13      temp = error'[j − 1] + current_error
14      if temp ≤ error'[i] then
15        error'[i] = temp
16        partition'[i] = j
17      end if
18    end for
19    if the previous partition comparing with the i position is a complete-alignment then
20      break from the i-loop
21    end if
22  end for
23  determine the remaining partition' and error'
```

Figure 4.1: Incremental Algorithm

Table 4.1: Original Dataset

Id	Data	*error*	*partition*
1	54	Infinity	0
2	55	Infinity	0
3	56	2	1
4	65	11	1
5	70	16	1
6	75	12	4
7	76	13	4
8	80	17	4
9	101	37	7
10	102	35	8
11	103	19	9
12	120	36	9
13	121	37	9
14	122	21	12

Table 4.2: The impact when a data tuple with 64 as its sensitive value is inserted, and a new partition is found

Id	Data	*error*	*error*′	*partition*	*partition*′
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
-1	64	-	8	-	1
4	65	11	11	1	1
5	70	16	10	1	4
6	75	12	15	4	4
7	76	13	16	4	4
8	80	17	15	4	7
9	101	37	36	7	7
10	102	35	37	8	7
11	103	19	17	9	7
12	120	36	34	9	10
13	121	37	35	9	10
14	122	21	19	12	13

The i-loop will continue until it finishes the processing for the d_{12} data tuple. At the point, the completed-alignment of the partition $\{d_9, d_{10}, d_{11}\}$ is found. Then, the i-loop is broken, and the remaining of the *partition* and the *error* values can be computed using Lemma 1 with $O(n - n')$ cost.

Chapter 5

Indexing

In order to further improve the efficiency of the proposed incremental algorithm, two indexes are proposed as follows.

5.1 Distinct Value Index

First, we propose an index on the distinct sensitive values. Because one of the most high-computation operations in the algorithm is to decide whether the given range of data tuples has enough distinct value to be a valid partition subjected to the k-condition. Additionally, the index can improve the efficiency of the complete-alignment identification. Since, the incremental algorithm relies on the break-point determination such that, it can stop the $O((n' - ins\_pos)^2)$ processing, and begin the $O(n - n')$ processing. If the cost of the alignment identification is lower, a high efficiency can be obtained.

After the algorithm is proposed, in order to further improve the efficiency of the incremental algorithm, there are a few possible indexes can be created.

Thus, we create an index which its key is the range of the positions. The value of the index is the sequence of the sensitive value of the data tuples in the partition belonged to the key, as well as their number of distinct values. Such index will be in a form of flat-file index, in which each entry is in the form $< (p_i, p_j), ((s_1, s_2, \ldots, s_n), no\_distinct) >$, where p_i and p_j is the range, $(s_1, s_2, \ldots, s_n)$, is the sequence of the sensitive values in the range, and $no\_distinct$ is the number of distinct values in such range. For example, given the dataset in Table 5.1, an index entry for the d_1 data tuple to the d_3 data

Table 5.1: An example dataset

Id	Data
1	54
2	55
3	56
4	65
5	70
6	75
7	76
8	80
9	101

Table 5.2: Index structure for the example dataset

Index Entry
$< (1,1), ((54),1) >$
$< (1,2), ((54,55),) >$
$< (1,3), ((54,55,56),3) >$
$< (1,4), ((54,55,56,65),4) >$
$< (1,5), ((54,55,56,65,70),5) >$
...
$< (5,6), ((70,75),2) >$
$< (5,7), ((70,75,76),3) >$
$< (5,8), ((70,75,76,80),4) >$
$< (8,9), ((80,101),2) >$
$< (9,9), ((101),1) >$

tuple is $< (1,3), ((54,55,56),3) >$. It can be seen that the sequence covers data tuples with 54, 55, and 56 as their sensitive values. And, the number of distinct values among them is 3. The index structure of this dataset is shown in Table 5.2.

5.2 Positioning Index

Second, given the fact that there could be a very large number of duplicates in real-life dataset. For example, consider the age attribute of the individuals, though the number of data tuples can be varied, age values can be limited to some certain ranges. Thus, we propose to apply a tree-based index, i.e. B^+-tree, which its key is the sensitive value of a data tuple, the value of its is the position of the first value of the key. Formally, the index entry is in the form of

$< d_i.sensitive, p_i >$. With this index, we can identify whether the current i-data tuple is the distinct value comparing with the previous value when processing the i-loop in the algorithm.

Chapter 6

Experiment Results

After our incremental algorithm and indexing are proposed, in this chapter, we present the experiment results to evaluate our work.

6.1 Configuration

The experiments were conducted on the Adult dataset from UCI Machine Learning Repository [12] which has been used to evaluate the (k, e)-anonymization [53, 54]. The dataset contains 14 attributes over 48,000 data tuples. The "capital-loss" attribute is selected as the sensitive values. Eight of the attributes are selected as the quasi-identifier as in [54], as well as the data cleansing processes. Thus, the remaining number of the dataset is 1427 tuples. The range of the capitol-loss values is 155 to 3900, with 89 distinct values.

The efficiency is evaluated in term of the execution time when the three parameters change, i.e. the k value, the e value, and the size of the incremental dataset $|\triangle D|$. In each experiment, the dataset will be divided into two equal parts, the first part is used as the static part of the data, while the latter will be used as the incremental data. The incremental part of the dataset will be appended to the static part in a one-by-one basis. Once, all the increment data are appended to the dataset, the execution time is reported. Additionally, the position of the break-point from the proposed incremental algorithm is reported to present the relationship between the position and the efficiency of the algorithm. Such positions are reported into the percentage of the skipped tuples relatively to the size of the dataset. The more the percentage means the more

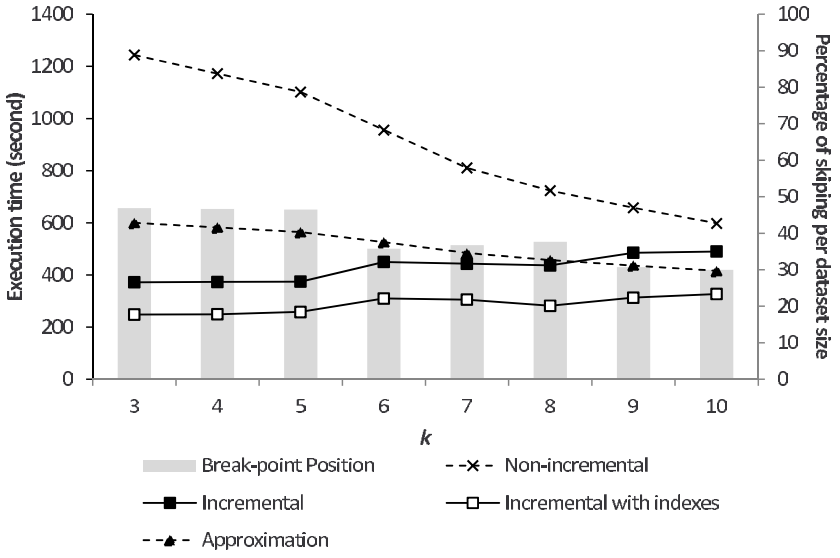


Figure 6.1: Effect of the k value

efficiency the algorithm can obtain. Note that the execution time of the incremental algorithm both with/without the proposed indexes will be reported to show their effects. Thus, the execution time reported for the incremental algorithm with the indexes consists of the execution time as well as the index update time. In each experiment, the proposed algorithm will be compared with the $O(n^2)$-algorithm in [54] and the 2-approximation algorithm with $O(n)$ complexity in [53]. Such 2-approximation algorithm is a greedy algorithm which scan the given dataset once for computing a cover of of the anonymous intervals based on the (k, e) condition. Note that the resulting numbers reported are five-time average.

6.2 Results and Discussion

6.2.1 Effects of k value

In the first experiment, we evaluate the efficiency of the incremental algorithm when the k value is varied. Such value is varied from 3 to 10 to evaluate its effect. The e value is fixed at 100. At each setting, a set of the increment data of size 10% of the existing dataset is inserted.

In Figure 6.1, the result of the experiment is presented. Obviously, when the

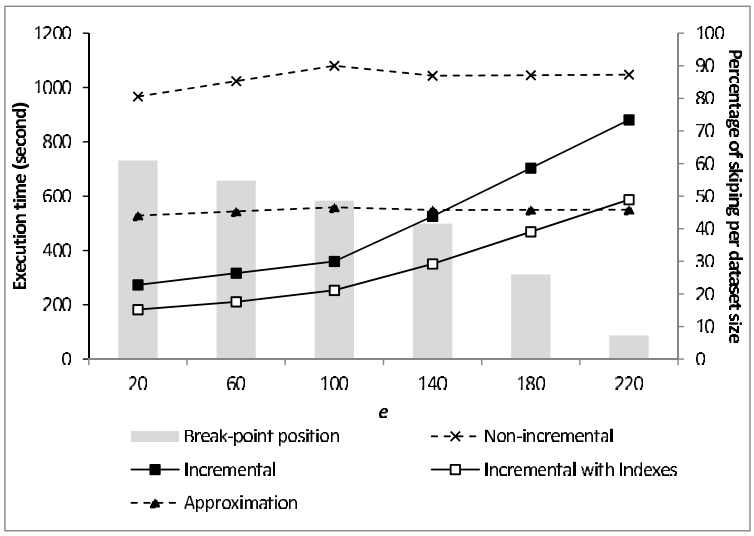


Figure 6.2: Effect of the e value

k value is increased, the execution time of the incremental algorithms are also increased. The rationale behind this is that when the k is higher, the break-point of the complete-alignment is more difficult to be found with regard to the k-condition. In the worst case, the algorithms need to compute the *error* and the *partition* values until the end of the dataset with $O((n' - ins\_pos)^2)$ cost. Thus, the efficiency is degraded. Comparing with the non-incremental algorithm, the fixed computational cost of $O(ins\_pos)$ of the incremental algorithms before the ins_pos-position is higher. We can see the effect of such cost when the k values is very high, the execution time of the incremental algorithm is even a bit higher. However, in [53], the authors suggested that the k value should be set at lower than 6 in general.

6.2.2 Effects of e value

In this experiment, we evaluate the efficiency of the incremental algorithm when the range of the tolerate error e is changed. The e will be increased from 20 to 220 to evaluate its effect. The k is fixed at 5, and $|\triangle D|$ at 10%. From the

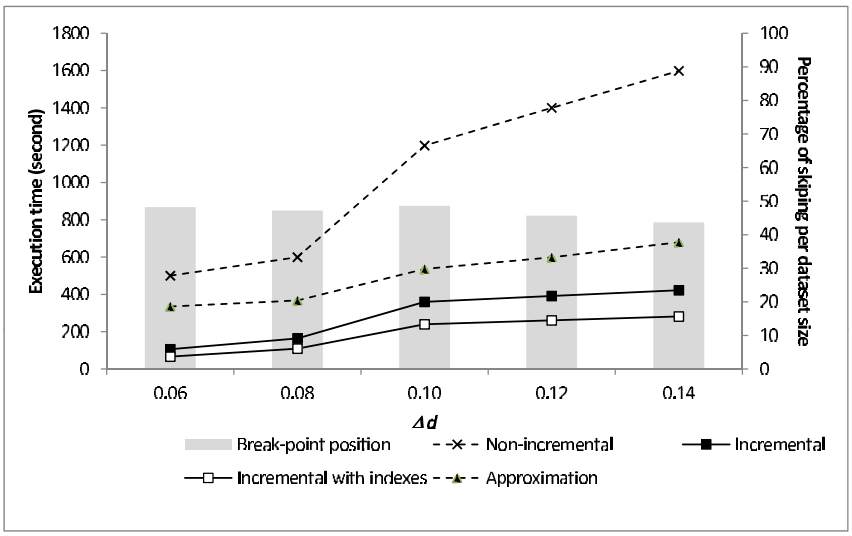


Figure 6.3: Effect of the $\triangle D$

result in Figure 6.2, it can be seen that the proposed incremental algorithm is more efficient than the non-incremental algorithm. These results are obvious particularly when the e value is set at 100 or lower in which the execution time of the non-incremental algorithm is more than 3-times of the proposed algorithm. Such gaps are caused by the discovery of the complete-alignment partition, and thus prevent the quadratic part of the algorithm to be executed till the end of the input. The rationale behind the efficiency degrade, after the e value is higher than 100, is each partition becomes larger. And, thus the complete-alignment is more difficult to be found as it can be seen the percentage of the skipped tuples from the break-point positions. Obviously, such percentage decreases when both the k and e parameters are increased because the mentioned reason.

6.2.3 Effects of $|\triangle D|$

In the last experiment, we evaluate the efficiency of the incremental algorithm when the size of the incremental dataset ($|\triangle D|$) is varied. The variation is set at the percentage of the whole data to be appended, i.e. 6, 8, 10, 12 and 14 %.

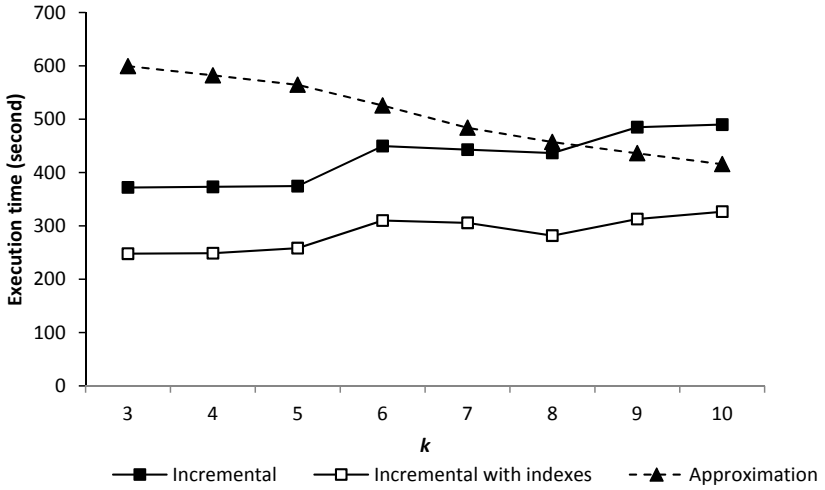


Figure 6.4: Effect of the k value considering indexes

We set the k and e values at 5 and 100 respectively. Figure 6.3 shows the result of this experiment. It can be seen that when the size of the appended data is increased, the gap between the execution time of both algorithms is increased. Obviously, the effect of the incremental processing is accumulated when the size of $\triangle D$ is increased. And, in all settings, both of the incremental algorithms outperform the approximation algorithm.

6.2.4 Indexing Performance

In this section, we report the performance of the indexes. The execution time of the proposed algorithm with/without indexes is reported to evaluate the performance. In addition, the execution time of the approximation algorithm is reported here to compare its efficiency with our proposed algorithm with indexes. Note that the execution time reported for the incremental algorithm with the indexes consists of the execution time as well as the index update time. The index performance when the k, e, and $|\triangle D|$ is varied is presented in Figure 6.4, 6.5, and 6.6 respectively.

From the figures, it can be seen that the incremental algorithms with the proposed indexes are more efficient than the without-indexes version of it sig-

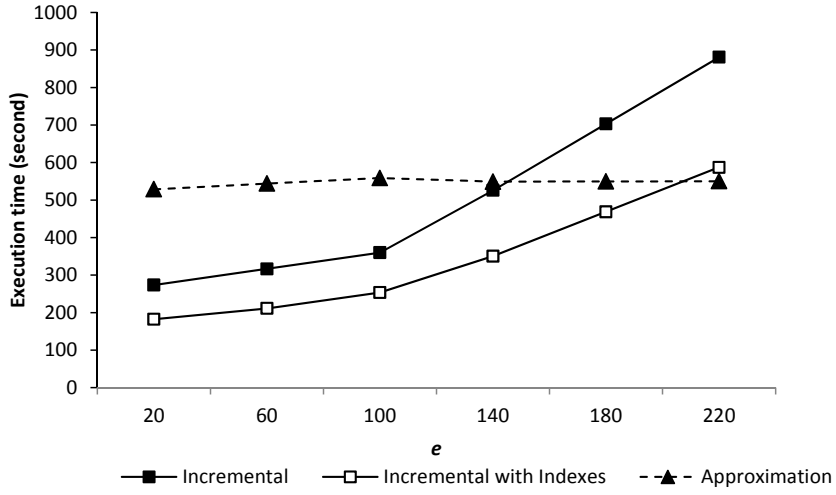


Figure 6.5: Effect of the e value considering indexes

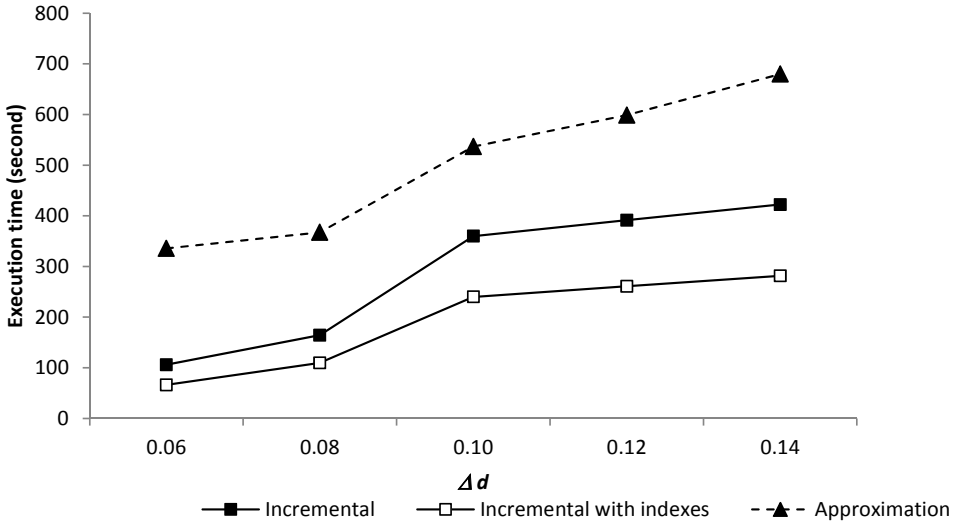


Figure 6.6: Effect of the $\triangle D$

nificantly. The algorithm with the indexes is even more efficient than the 2-approximation algorithm which has $O(n)$ complexity, meanwhile the optimal solutions can be obtained. Also, it can be seen that the incremental algorithm with indexes can outperform the 2-approximation algorithm when the e value is not extremely high. Given the fact that the optimal solutions can be obtained instead of the approximate results, our proposed algorithm is very efficient.

Chapter 7

Conclusion

In this work, we have proposed a solution to the focused problem. That is, an observation on the privacy preservation based on the (k, e)-anonymization when the data increment is considered. We have subsequently, proposed an incremental algorithm for the problem. Such algorithm can process the data incrementally while exactly the same solutions as the non-incremental algorithm can be guaranteed. The proposed algorithm can be more efficient than the non-incremental algorithm when the complete-alignment partition is discovered before the end of the data input is to be processed. Eventually, two indexes are proposed to resolve the efficiency issue. From the experiments, it has been found that the incremental algorithm with/without indexes is much more efficient than the non-incremental algorithm when the k or e value is not extremely high. When the size of the increment data is increased, the efficiency gap between the proposed algorithm and the non-incremental algorithm is larger. Also, the incremental algorithm with the proposed indexes can outperform the $O(n)$ approximation in most of the experiment settings. These can be summarized that the proposed algorithm can be very efficient in the real-world situations.

In our future work, we will investigate on a few interesting issues as follows. One of them is bulk-data insertion, i.e. to insert a relative-small/medium set of data into the existing dataset at once. For example, one may need to insert an additional dataset in Table 7.1 into the existing dataset in Table 7.2. The nave approach is to combine the datasets from the two datasets together, subsequently, re-process the privacy preservation. Or, a more efficient way would be applying the algorithm in [45] to individually insert the additional dataset.

The more efficient solution for such issue can help in the real-world situations

Table 7.1: An example additional dataset to be inserted

Postal Code	Age	Sex	Salary
50210	45	Male	16,000
50210	36	Female	20,000
50200	41	Male	15,000

Table 7.2: An example dataset

Postal Code	Age	Sex	Salary
50200	35	Male	14,000
50210	36	Male	15,000
50230	40	Male	16,000
50300	41	Female	25,000
50310	43	Female	35,000
50330	47	Male	30,000
50300	53	Male	40,000
50310	54	Female	35,000
50330	58	Male	45,000

where the data are collected in a fixed-time interval. For example, the Ministry of public health requires each hospital to submit the medical records collected in past months. Processing such bulk of records at once, rather than processing each record individually, will be more efficient and realistic. In order to apply the bulk insertion, there are a few sub-issues need to be addressed. That is, the efficiency of the insertion both in term of the computational and space cost should be considered. The bulk insertion of n tuples should be more efficient than insert each of the n tuples individually. The space requirement in order to perform the bulk insertion should not be the burden to the task. Additionally, the concurrency control of the indexes or the auxiliary data structures of the algorithm should be considered. Since the bulk insertion of large data might need to acquire the lock in order to perform the update operation. The nave approach is to preserve the lock until the bulk insertion is finished. However, the approach can degrade the efficiency of the indexes. Thus, the concurrency control in this case, should be carefully designed.

Bibliography

[1] C. C. Aggarwal and P. S. Yu. A condensation approach to privacy preserving data mining. In *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology*, pages 183–199. Springer-Verlag, 2004.

[2] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 247–255. ACM Press, 2001.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 439–450. ACM Press, 2000.

[4] M. Atallah, A. Elmagarmid, M. Ibrahim, E. Bertino, and V. Verykios. Disclosure limitation of sensitive rules. In *KDEX '99: Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange*, pages 45–52, Washington, DC, USA, 1999. IEEE Computer Society.

[5] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. Blocking anonymity threats raised by frequent itemset mining. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 561–564, Washington, DC, USA, 2005. IEEE Computer Society.

[6] M. Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi. k-anonymous patterns. In *Proceedings of the 9th PKDD European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 10–21. Springer-Verlag, 2005.

[7] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[8] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10, New York, NY, USA, 1988. ACM Press.

[9] S. Benferhat, R. E. Baida, and F. Cuppens. A stratification-based approach for handling conflicts in access control. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 189–195. ACM Press, 2003.

[10] E. Bertino, P. Samarati, and S. Jajodia. Authorizations in relational database management systems. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 130–139, New York, NY, USA, 1993. ACM Press.

[11] E. Bertino, P. Samarati, and S. Jajodia. An extended authorization model for relational databases. *IEEE Transactions on Knowledge and Data Engineering*, 9(1):85–101, 1997.

[12] C. Blake and C. Merz. UCI repository of machine learning databases, 1998.

[13] J.-W. Byun, T. Li, E. Bertino, N. Li, and Y. Sohn. Privacy-preserving incremental data dissemination. *Journal of Computer Security*, 17(1):43–68, 2009.

[14] L. Chang and I. S. Moskowitz. Parsimonious downgrading and decision trees applied to the inference problem. In *Workshop on New Security Paradigms*, pages 82–89, 1998.

[15] D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19, New York, NY, USA, 1988. ACM Press.

[16] C. Clifton. Using sample size to limit exposure to data mining. *Journal of Computer Security*, 8(4):281–307, 2000.

[17] C. Clifton and V. Estivill-Castro, editors. *CRPIT '14: Proceedings of the IEEE international conference on Privacy, security and data mining*. Australian Computer Society, Inc., Darlinghurst, Australia, 2002.

[18] H. T. Croft, H. J. Falconer, and R. K. Guy. *Unsolved Problems in Geometry*. Springer-Verlag, New York, NY, USA, 1991.

[19] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding association rules by using confidence and support. In *Proceedings of the 4th International Workshop on Information Hiding*, pages 369–383. Springer-Verlag, 2001.

[20] H. S. Delugach and T. H. Hinke. Wizard: A database inference analysis and detection system. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):56–66, 1996.

[21] D. E. Denning, S. G. Akl, M. Heckman, T. F. Lunt, M. Morgenstern, P. G. Neumann, and R. R. Schell. Views for multilevel database security. *IEEE Transactions on Software Engineering*, 13(2):129–140, 1987.

[22] J. Domingo-Ferrer and V. Torra, editors. *Privacy in Statistical Databases*, volume 3050 of *LNCS*. Springer, Berlin Heidelberg, 2004.

[23] J. Doyle. Pharmacy customers complain of privacy violations, delays. 2011.

[24] A. Evfimievski, R. Srikant, R. Agarwal, and J. Gehrke. Privacy preserving mining of association rules. *Information Systems*, 29(4):343–364, 2004.

[25] R. Fagin. On an authorization mechanism. *ACM Transactions on Database Systems*, 3(3):310–319, 1978.

[26] S. Farooq. Starbucks' foursquare check ins has its perks. 2011.

[27] P. Fule and J. F. Roddick. Detecting privacy and ethical sensitivity in data mining results. In *ACSC '04: Proceedings of the 27th Australasian conference on Computer science*, pages 159–166, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[28] B. C. Fung, K. Wang, and P. S. Yu. Anonymizing classification data for privacy preservation. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):711–725, 2007.

[29] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, June 2010.

[30] B. C. M. Fung, K. Wang, A. W.-C. Fu, and J. Pei. Anonymity for continuous data publishing. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, EDBT '08, pages 264–275, 2008.

[31] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229, New York, NY, USA, 1987. ACM Press.

[32] P. P. Griffiths and B. W. Wade. An authorization mechanism for a relational database system. *ACM Transactions on Database Systems*, 1(3):242–255, 1976.

[33] V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 279–288. ACM, 2002.

[34] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 474–485. ACM Press, 1997.

[35] R. J. B. Jr. and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st IEEE ICDE International Conference on Data Engineering*, pages 217–228. IEEE Computer Society, 2005.

[36] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Data and Knowledge Engineering*, 16(9):1026–1037, 2004.

[37] J. Li, R. C.-W. Wong, A. W.-C. Fu, and J. Pei. Achieving k-anonymity by clustering in attribute hierarchical structures. In *Proceedings of 8th International Conference on Data Warehousing and Knowledge Discovery*, Lecture Notes in Computer Science, pages 405–416. Springer, 2006.

[38] N. Li, T. Li, and S. Venkatasubramanian. Closeness: A new privacy measure for data publishing. *IEEE Transactions on Knowledge and Data Engineering*, 22(7):943–956, 2009.

[39] X.-B. Li and S. Sarkar. A tree-based data perturbation approach for privacy-preserving data mining. *IEEE Transactions on Knowledge and Data Engineering*, 18(9):1278–1283, 2006.

[40] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 36–54. Springer-Verlag, 2000.

[41] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. ℓ-diversity: Privacy beyond κ-anonymity. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 24, Washington, DC, USA, 2006. IEEE Computer Society.

[42] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[43] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 223–228. ACM, 2004.

[44] G. V. Moustakides and V. S. Verykios. A max-min approach for hiding frequent itemsets. In *Workshops Proceedings of the 6th IEEE ICDM nternational Conference on Data Mining*, pages 502–506. IEEE Computer Society, 2006.

[45] J. Natwichai, X. Li, and A. Kawtrakul. Incremental Processing and Indexing for (k, e)-Anonymization. *Unpublished manuscript*.

[46] S. News. Vodafone warned over customer privacy leaks. 2011.

[47] S. R. M. Oliveira and O. R. Zaïane. Privacy preserving frequent itemset mining. In *Proceedings of the IEEE international conference on Privacy, security and data mining*, pages 43–54. Australian Computer Society, Inc., 2002.

[48] S. R. M. Oliveira and O. R. Zaïane. Algorithms for balancing privacy and knowledge discovery in association rule mining. In *7th International Database Engineering and Applications Symposium*, pages 54–65. IEEE Computer Society, 2003.

[49] S. R. M. Oliveira and O. R. Zaïane. Protecting sensitive knowledge by data sanitization. In *Proceedings of the 3rd IEEE ICDM International Conference on Data Mining*, pages 613–616. IEEE Computer Society, 2003.

[50] S. R. M. Oliveira and O. R. Zaïane. Achieving privacy preservation when sharing data for clustering. In *Proceedings of the Workshop on Secure Data Management (SDM'04)*, pages 67–82. Springer, 2004.

[51] S. R. M. Oliveira and O. R. Zaïane. Privacy-preserving clustering by object similarity-based representation and dimensionality reduction transformation. In *Proceedings of the Workshop on Privacy and Security Aspects of Data Mining (PSDM'04) in conjunction with the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 21–30, 2004.

[52] J. Pei, J. Xu, Z. Wang, W. Wang, and K. Wang. Maintaining k-anonymity against incremental updates. In *Proceedings of the 19th International Conference on Scientific and Statistical Database Management*, SSDBM '07, pages 5–, 2007.

[53] C. M. Procopiuc and D. Srivastava. Efficient table anonymization for aggregate query answering. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 1291–1294, Washington, DC, USA, 2009. IEEE Computer Society.

[54] Z. Qing, N. Koudas, D. Srivastava, and Y. Ting. Aggregate query answering on anonymized tables. In *Proceedings of the 2007 IEEE International Conference on Data Engineering*, pages 116 – 125, Washington, DC, USA, 2007. IEEE Computer Society.

[55] J. Russell. Facebook surge puts thailand into worlds top 25. 2010.

[56] Y. Saygin, V. S. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Rec.*, 30(4):45–54, 2001.

[57] B. Seisungsittisunti and J. Natwichai. Incremental privacy preservation for associative classification. In *Proceeding of the ACM first international workshop on Privacy and anonymity for very large databases*, PAVLAD '09, pages 37–44, 2009.

[58] X. Sun and P. S. Yu. A border-based approach for hiding sensitive frequent itemsets. In *ICDM '05: Proceedings of the Fifth IEEE International*

Conference on Data Mining, pages 426–433, Washington, DC, USA, 2005. IEEE Computer Society.

[59] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–588, 2002.

[60] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

[61] T. M. Truta and A. Campan. K-anonymization incremental maintenance and optimization techniques. In *SAC '07: Proceedings of the 2007 ACM Symposium on Applied Computing*, pages 380–387, New York, NY, USA, 2007. ACM.

[62] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 639–644. ACM Press, 2002.

[63] J. Van Grove. Mayors of starbucks now get discounts nationwide with foursquare. 2010.

[64] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Rec.*, 33(1):50–57, 2004.

[65] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE Transactions on Data and Knowledge Engineering*, 16(4):434–447, 2004.

[66] K. Wang and B. C. M. Fung. Anonymizing sequential releases. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 414–423, New York, NY, USA, 2006. ACM Press.

[67] K. Wang, B. C. M. Fung, and P. S. Yu. Template-based privacy preservation in classification problems. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 466–473, Washington, DC, USA, 2005. IEEE Computer Society.

[68] R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 754–759, New York, NY, USA, 2006. ACM Press.

[69] Y.-H. Wu, C.-M. Chiang, and A. L. P. Chen. Hiding sensitive association rules with limited side effects. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):29–42, 2007.

[70] X. Xiao and Y. Tao. M-invariance: towards privacy preserving re-publication of dynamic datasets. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, SIGMOD '07, pages 689–700, 2007.

[71] A. C.-C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science (FOCS'86)*, pages 162–167, Toronto, Canada, 1986. IEEE Computer Society.

[72] N. Zhang, S. Wang, and W. Zhao. A new scheme on privacy-preserving data classification. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 374–383, New York, NY, USA, 2005. ACM Press.

[73] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, and Y. Jia. Continuous privacy preserving publishing of data streams. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 648–659, 2009.

Output จากโครงการวิจัยที่ได้รับทุนจาก สกว.

1. ได้ส่งผลงานเพื่อตีพิมพ์ในวารสารวิชาการนานาชาติ International Journal of Information and Computer Security โดย Juggapong Natwichai, Xue Li, and Asanee Kawtrakul โดยจะทราบผลภายในเดือน ก.ค. นี้
2. การนำผลงานวิจัยไปใช้ประโยชน์
   - เชิงวิชาการ
     - ได้นำผลงานไปใช้เป็นส่วนหนึ่งของวิชา Advanced Database System ณ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ โดยมุ่งเน้นประเด็นการตั้งข้อสังเกตกับอัลกอริทึมและการทดสอบประสิทธิภาพ
     - ได้ดัดแปลงปัญหาที่แก้ในงานวิจัยนี้ ไปเป็นปัญหาวิจัยในวิทยานิพนธ์ของนักศึกษาระดับปริญญาเอก หลักสูตรปรัชญาศาตรดุษฎีบัณฑิต (วิศวกรรมคอมพิวเตอร์) คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ โดยแปลงให้เป็นการรักษาความเป็นส่วนตัวในลักษณะข้อมูลหลายเวอร์ชัน
3. อื่นๆ -

Appendix

Incremental Processing and Indexing for (k, e)-Anonymization

May 27, 2012

Abstract

The emerging of the internet-based services poses a privacy threat to the individuals. Data transformation to meet a privacy standard becomes a requirement for typical data processing for the services. (k, e)-anonymization is one of the most promising data transformation approaches, since it can provide high-accuracy aggregate query results. Though, the computational cost of the algorithm providing optimal solutions for such approach is not very high, i.e. $O(n^2)$. In certain environments, the data to be processed can be appended at any time. In this paper, we address an efficiency issue of the incremental privacy preservation using (k, e)-anonymization approach. The impact of the increment is observed theoretically. We propose an incremental algorithm based on such observation. The algorithm can replace the quadratic-complexity processing by a linear function on some part of the dataset, while the optimal results are guaranteed. Additionally, a few indexes are proposed to further improve the efficiency of the proposed algorithm. The experiments have been conducted to validate our work. From the results, it can be seen that the proposed work is highly efficient comparing with the non-incremental algorithm and an approximation algorithm.

Keywords: Incremental Processing; Privacy; Anonymization; Indexing.

1 Introduction

Privacy is one of the most important issues for any data processing these days. Since the organisations have more opportunity to collect the data from the individuals. Such examples can be social network applications, or the location-based services which have been utilised widely. The collected data can be used to improve the quality of services by data analysing tools and algorithms, e.g. user-clustering can help improving campaign targeting. The results will benefit both the organisations and the individuals. However, the privacy of the individuals should not been compromised.

Consequently, the privacy preservation becomes one of the most active research areas. One of the first few privacy preservation approaches which have been proposed, is k-anonymization Samarati & Sweeney (1998), Samarati (2001), Sweeney (2002*b*). Its fundamental is to hide an individual within a k-size group with identical information. Subsequently, the ℓ-diversity, which further protects the privacy by enforcing the minimum ℓ-distinct sensitive data in the group, has been proposed in Machanavajjhala et al. (2006). Such two mentioned approaches are based on the data transformation by generalisation, i.e. changing a privacy sensitive value into its more-generalised value e.g. from a postal code 50202 to 5020*. Furthermore, there are a few more effective approaches based on such work to preserve the privacy, for example, t-closeness Li et al. (2009) which extends the l-diversity approach by preserving the distribution of the sensitive values in each group.

One of the most important approaches for preserving the privacy is (k, e)-anonymization Qing et al. (2007), Procopiuc & Srivastava (2009). With this

Table 1: An example dataset

Postal code	Age	Sex	Salary
50200	35	Male	14,000
50210	36	Male	15,000
50230	40	Male	16,000
50300	41	Female	25,000
50310	43	Female	35,000
50330	47	Male	30,000
50300	53	Male	40,000
50310	54	Female	35,000
50330	58	Male	45,000

Table 2: The partitioned dataset

Postal code	Age	Sex	Salary
50200	35	Male	15,000
50210	36	Male	16,000
50230	40	Male	14,000
50300	41	Female	25,000
50310	43	Female	30,000
50330	47	Male	35,000
50300	53	Male	35,000
50310	54	Female	40,000
50330	58	Male	45,000

approach, the sensitive data are partitioned, in which the probability to identify the individuals in each partition is not exceeded the pre-specified thresholds, k and e. The k values use to control the number of distinct sensitive data in a group, while the e values use to control the range of the sensitive data. Subsequently, the sensitive data in each partition is swapped, or permuted, to protect the privacy. Not only that the privacy can be preserved effectively, but also a high data utility can be obtained by applying such approach. The utility in the (k, e)-anonymization is defined in term of the accuracy of the aggregate query results. In Qing et al. (2007), the authors reported that the error from the aggregate query result is very low (1% error approximately), i.e. the result is very close to the non-privacy preserved result.

An example dataset to illustrate the (k, e)-anonymization is shown in Table

1. In the table, a dataset is composed by three non-sensitive attributes, i.e. postal code, age, and sex, of the individuals. The sensitive attribute is the monthly-salary in Thai Baht (THB). If the k and the e values are set at 3 and 2,000 respectively, an optimal solution with regard to the minimum-sum-error objective (to be described in Section 3), is shown in Table 2.

From Table 2, the partition of the dataset is outlined, i.e. there are three partitions. It can be seen that there is at least 3 distinct salary values in each partition, as well as the range of the salary is at least 2,000 THB. The sensitive data have been permuted already. If an aggregate query "find the summation of the salary of the 40-48 years-old" is issued, the result from the permuted dataset is 104,000-106,000 THB (A data tuple from the first partition, all of the tuples from the second partition, and none from the last partition are involved in the query answering). Meanwhile the same query executed on the original dataset results in 106,000 THB which is very close to the result obtained from the privacy preserved dataset.

The complexity of the algorithm generated the optimal solutions is $O(n^2)$ where n is the number of data tuples in the given dataset Qing et al. (2007). Such algorithm composes of two phases, i.e. partitioning the given dataset according to (k, e)-anonymization, as well as permuting the sensitive values within each partition. The partitioning phase, which costs $O(n^2)$, is our focus in this paper. Though, such cost is not very high, the data to be processed might not be static. There are some situations that the data to be transformed can be appended all the time Truta & Campan (2007), Seisungsittisunti & Natwichai (2009). For example, the Ministry of public health in Thailand requires each hospital to submit the monthly medical records to the global data warehouse. Re-applying the $O(n^2)$-algorithm every time the monthly data are loaded might not be an efficient way to approach the problem in such environ-

ment.

In this paper, we address the incremental privacy preservation problem based on the (k, e)-anonymization. We start with making the observation on the data appending theoretically. Subsequently, an algorithm based on the observation is proposed. In which the quadratic-complexity processing on some part of the dataset can be replaced by the linear-complexity processing. Furthermore, two indexes are proposed to improve the efficiency of the algorithm. The experiments have been conducted to validate our work. From the results, the proposed work is highly efficient comparing with the non-incremental algorithm and an approximation algorithm, while the same results with re-applying the optimal non-incremental algorithm can be guaranteed.

The organisation of this paper is as follows. The related work is presented in the next section. The basic notation for defining the problem is presented in Section 3. Subsequently, the observations on the data increment and the incremental algorithm are proposed in Section 4. The indexes for further efficiency improvement are proposed in Section 5. In Section 6, the experiment results to evaluate our work are reported. Finally, Section 7 gives the conclusion and outline of our future work.

2 Related Work

In general, the data privacy preservation issues can be addressed by the traditional database techniques such as security view management Denning et al. (1987) or access-control methods Jajodia et al. (1997) which have been studied in the past decades. With these approaches, data privacy can be preserved by creating different views/rights for different users with different privilege levels. Also, the statistical security-control Domingo-Ferrer & Torra (2004) is another approach for addressing the issues. With this approach, noises are inserted into

the original dataset in order to differentiate the utility of the dataset. While some specific statistical values, for example, the mean or variance, relied on the noise-added data must be maintained as close to the original values as possible.

k-anonymity, which is a well-known privacy preservation model, was firstly proposed Samarati & Sweeney (1998). There has been a lot of work applied the k-anonymity because its simplicity and meaningful of the transformed data Machanavajjhala et al. (2006), Sweeney (2002a), Wong et al. (2006), Fung et al. (2007). However, when the data utility of the transformed datasets is concerned, the transformation problem is not trivial. In Meyerson & Williams (2004), the problem that requires an optimal anonymization, i.e. satisfying k while having minimal impact on data utility, has been proven as an NP-hard problem.

There have been a few privacy preservation models built based on the k-anonymity model as follows. In Machanavajjhala et al. (2006), the authors proposed another privacy preservation model, ℓ-diversity, which improves the k-anonymity model by preventing homogeneity attack. The attack occurs when the k-anonymized dataset has too less diversity though it satisfies the k-anonymity condition. In Li et al. (2009), the authors showed that there exist some situations which the ℓ-diversity may not secure enough to publish the data. Therefore, they proposed the t-closeness model in which the data distribution of a sensitive attribute in any group should be similar to the distribution of such attribute in the dataset in order to guarantee the privacy of the given data. Such model has been proven as an efficient privacy preservation

(k, e)-anonymization was first proposed in Qing et al. (2007). As opposed to the generalisation-based approach, which the data values are transformed into a more general form of them, the approach begins with data partitioning process subjected to the k distinct values, and the e range. Then, the data

within each partition are permuted to protect the privacy. The partitioning phase can help increasing the utility obtained from the data, since the data transformation is proceeded in the scope of the partition. Comparing with the t-closeness, the (k, e)-anonymization rather aims at optimising the range of the sensitive data. Also, the aggregate query can be answered effectively with less optimisation constraints.

For the incremental privacy preservation, a few studies have been reported Xiao & Tao (2007), Pei et al. (2007), Fung et al. (2008), Byun et al. (2009). Although the incremental algorithms without re-processing the whole datasets have been proposed, these algorithms deal with the situations where multiple versions of the datasets are released. The different versions may have different levels of anonymization e.g. different generalised or shuffling values from different privacy parameters. Unlike our work, we deal with only a single version of the dataset with a single level of privacy parameters (pre-specified by the responsible agencies e.g. the Ministry of public health in Thailand as mentioned above). It may be released publicly multiple times, however, joining them do not give the attacker any additional knowledge.

Additionally, an approach to preserve the privacy in data streams has been proposed in Zhou et al. (2009). The approach focuses on transaction streams where the data of an individual may have been stored multiple times in a dataset. To deal with the continuous nature of the data streams, the lists of equivalence classes are kept and maintained for the incremental stream processing.

3 Basic Definition

In this section, we present the definitions required for our proposed approach.

Definition 3.1 (Dataset) *Let a dataset* $DID = \{d_1, d_2, \ldots, d_n\}$ *be a collec-*

tion of tuples defined on a schema $\boldsymbol{A}$*, which consists of an identifier* ID*, a set of quasi-identifiers* $QI = \{qi_1, qi_2, \ldots, qi_k\}$*, and the numerical sensitive attribute* S*. Each* d_i *can be referred as the* i*-data tuple.*

We consider to transform the datasets without identifiers defined as follows.

Definition 3.2 (De-identified Dataset) *The de-identified version of a dataset* DID *is a projection of it over the set of quasi-identifiers and the sensitive attribute, denoted as* D*.*

Such de-identified datasets can have an overlapped public dataset on some quasi-identifier attributes. The public dataset also contains the identifiers which lead to the privacy breach. Our goal here is to transform the de-identified datasets to break such link between the quasi-identifier attributes and the sensitive value such that one who owns the public datasets cannot match them to the transformed dataset, while the aggregate queries can still be effectively answered.

Definition 3.3 (Aggregate Queries) *An aggregate query is an SQL-query which applies the aggregate functions including COUNT, MAX, MIN, SUM, and AVERAGE on the sensitive attributes of the de-identified datasets.*

Definition 3.4 ((k, e)**-anonymity)** *A de-identified dataset* D *is satisfied a* (k, e)*-anonymity iff for each partition* $P_i \subseteq D$*, the projection over the sensitive attribute* S *in each* P_i *provides at least* k *distinct values and the range of these different values in* P_i *is at least* e*.*

Definition 3.5 ((k, e)**-permutation)** *Let* p *be a random permutation over the tuple identifier* $\{1, 2, \ldots, n\}$*. The permutation of a de-identified* D *is denoted as* $p(D, QI, S) = \{d_{i'} | \forall qi_j \in QI, d'_i.qi_j = d_i.qi_j \text{ and } d_{i'}.S = d_{p(i)}.S\}$*. Given a random permutation* p *and a de-identified* (k, e)*-anonymity dataset* D*, the*

(k, e)-anonymity permutation of D is $p(D, QI, S)$, or the (k, e)-permutation for short.

Obviously, there could be many (k, e)-anonymity permutation datasets for a given data, the transformation algorithm should select the optimal-utility dataset, i.e. the dataset which has the least error. Such error can be defined in two approaches Qing et al. (2007), i.e. summation-error and maximum-error. We focus on the former error definition defined as follows.

Definition 3.6 (Error) *Let a set of partitions $\{P_1, P_2, \ldots, P_m\}$ be a partitioning of a dataset D resulted from the (k, e)-permutation transformation. Let the $error(P_i)$ be the error occurs in a partition P_i, $error(P_i) = max(S_i) - min(S_i)$, where max and min is the function to determine the maximum and minimum sensitive value from P_i respectively. The **summation-error** of the dataset D from the (k, e)-permutation is $\sum_{i=1}^{m} error(P_i)$.*

Thus, the (k, e)-anonymization problem is to determine the $p(D, QI, S)$ which has minimal **summation-error** given the dataset D, the k, and e parameters.

4 Incremental Processing

4.1 Effect of data increment

As we focus on the data-incremental environments, in this section we present the observation on the increment in which its results will be used to develop an incremental algorithm in the next subsection. First, the $O(n^2)$ for preserving the privacy proposed in Qing et al. (2007) with regard to the (k, e)-anonymization is presented in Figure 1. The algorithm computes the optimal summation error in each i-loop. The error value is maintained in the *error*

Input:
D: a dataset
k: a minimum number of distinct values threshold
e: a minimum range of values threshold
Output:
(D': the output dataset, which satisfies the (k, e)-anonymization
partition: the partition information
error: the error information
Method:

```
error[0] =infinity
partition[0] =0
for i = 1 to D.size
  error[i] = infinity
  partition[i] = partition[i − 1]
  for j = 1 to i
    if distinct({d_j, ..., d_i}) ≥ k and (d_i − d_j) ≥ e then
      current_error = (d_i − d_j)
    else
      current_error = infinity
    end if
    temp = error[j − 1] + current_error
    if temp ≤ error[i] then
      error[i] = temp
      partition[i] = j
    end if
  end for
end for
```

Figure 1: Original $O(n^2)$-Algorithm

variable. That is, the $error[D.size]$ contains the summation error of the optimal solution. The algorithm applies a greedy approach which it can be seen in the d loop. The algorithm will compare the "valid" current error subjected to the k and e values, to the best-known error for such d_i data tuple. If the current error is less than the existing value, it is stored as the new error in $error[i]$. Also, the index of the data tuple d_j which led to the lower error is stored in $partition[i]$. This comparison will contribute to the sub-loop $O(n)$ for each i loop. And, thus the complexity of the algorithm becomes $O(n^2)$.

Let us illustrate the execution of the algorithm by examples. Suppose that the sensitive values of a dataset is as shown in Table 3 with the k and e values

Table 3: An example sensitive dataset to demonstrate the $O(n^2)$

Id	data	*error*	*partition*
1	54	Infinity	0
2	55	Infinity	0
3	56	2	1
4	65	11	1
5	70	16	1
6	75	12	4
7	75	12	4
8	80	17	4
9	85	22	7

Table 4: The impact when a data tuple with 67 as its sensitive value is inserted

Id	data	*error*	*error′*	*partition*	*partition′*
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
4	65	11	11	1	1
-1	67	-	13	-	1
5	70	16	7	1	4
6	75	12	12	4	4
7	75	12	12	4	4
8	80	17	17	4	4
9	85	22	17	7	7

are set at 3 and 2 respectively (Note that the quasi-identifiers are omitted). When the algorithm is applied to the dataset, its *error* and *partition* values are shown in the third and fourth columns of Table 3. The partition structure can be obtained by scanning the *partition* variable backward, thus the partition results are $\{d_1, d_2, d_3\}$, $\{d_4, d_5, d_6\}$, and $\{d_7, d_8, d_9\}$ as outlined in the table.

When an incremental data tuple $\triangle d$ is inserted to the dataset, the naïve approach is re-applying the algorithm again. However, we aim at addressing the incremental data transformation problem in a more efficient way. Thus, we analyse the impact of the increment by re-applying the algorithm theoretically as follows. First, we categorise the increments into two main groups. The first group is when the increment data are inserted at the "in-range" of any existing

partition. The second group is when the increment data are inserted at the border of any two existing partitions. We discuss the details of each group in the following subsections. Subsequently, an approach to compute the impact of the increment is presented in the Section 4.2.

4.1.1 In-range Increment

For the "in-range" increment, when a data tuple $\triangle d$ is inserted into a partition Pi, which obtained from an optimal solution. The impact caused by re-applying the algorithm could be the partition Pi', the partition Pi after the insertion identified by the first/last data tuple in it, can be broken into a complete partition satisfying the (k, e)-condition before the ending/beginning tuples of the old Pi. The left-out data tuples will be merged and considered with the consecutive partition Pj, which is preceded by Pi or next to Pi. In this case, the optimal summation-error can be reduced, since the partition can be completed before the position it was.

For example, let us re-consider the dataset in Table 3. Suppose that the data tuple with 67 as its sensitive tuple is inserted as shown in Table 4. The partition, which the data tuple is inserted to, was $\{d_4, d_5, d_6\}$ with the summation-error at 10. After the insertion, if the optimal algorithm is re-applied, the partition is broken before the d_6 data tuple is considered. The modified partition becomes $\{d_4, d_{-1}, d_5\}$ with the summation-error at 5. Consecutively, the 6-data tuple is merged with last partition, however it does not increase the summation-error since the value is duplicate with a data tuple within the partition. Overall, the insertion causes the summation-error reduce from 22 to 17.

Another example is shown in Table 5. The scenario is a data tuple with 76 as its sensitive value is inserted to the existing dataset. The partition structure has become $\{d_1, d_2, d_3\}$, $\{d_4, d_5, d_6, d_7\}$, and $\{d_{-1}, d_8, d_9\}$. The summation-

Table 5: The impact when a data tuple with 76 as its sensitive value is inserted

Id	data	*error*	*error′*	*partition*	*partition′*
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
4	65	11	11	1	1
5	70	16	16	1	1
6	75	12	12	4	4
7	75	12	12	4	4
-1	76	-	13	-	4
8	80	17	17	4	7
9	85	22	21	7	8

Table 6: The impact when a data tuple with 56 as its sensitive value is inserted

Id	data	*error*	*error′*	*partition*	*partition′*
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
-1	56	-	2	-	1
3	57	3	3	1	1
4	65	11	11	1	1
5	70	16	15	1	4
6	75	13	13	4	5
7	75	13	13	4	5
8	80	18	18	4	5
9	85	23	23	7	8

error reduce from 22 to 21, and the last partition is completed before the d_7 data tuple is considered.

The summation-error is not always changed, the "in-range" increment can also leave the summation-error intact. For example, let us consider a dataset in Table 6. It can be seen that the insertion does not cause the ending data tuples (the d_3 data tuple, in this case) to be merged with the consecutive partition. Because the merging can increase the summation-error with regard to the optimal solution. Instead, the result from re-applying the optimal algorithm keeps the beginning and the ending of the existing partitions the same without increasing the error.

Table 7: The impact when a data tuple with 57 as its sensitive value is inserted

Id	data	*error*	*error′*	*partition*	*partition′*
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
-1	57	-	3	-	1
4	65	11	11	1	1
5	70	16	15	1	4
6	75	12	13	4	5
7	75	12	13	4	5
8	80	17	18	4	5
9	85	22	23	7	8

4.1.2 Border Increment

For the "border" increment, when a data tuple $\triangle d$ is inserted between a preceding partition Pi and another consecutive Pj, both obtained from an optimal solution. One of the possible impacts from the re-applying is that partition Pi merges such $\triangle d$ into it. As a result the structure of partition Pj and the rest are not effected. In this case, the optimal summation-error is to be increased since the preceding partition is extended to embrace $\triangle d$.

For example, consider the dataset in Table 7. It can be seen that a data tuple with 57 as its sensitive value is inserted between the first and the second partitions, i.e. $\{d_1, d_2, d_3\}$ and $\{d_4, d_5, d_6\}$ respectively. The optimal result by re-applying the algorithm after the insertion is that the data tuple is merged to the first partition. The other partitions have no impact from such insertion. As a result, we obtain the optimal partition from the re-applying as $\{d_1, d_2, d_3, d_{-1}\}$, $\{d_4, d_5, d_6\}$, and $\{d_7, d_8, d_9\}$. Then, the summation-error is increased from 22 to 23.

When an increment data tuple is inserted at the border, it can also be merged with the consecutive partition Pj, and the rest of the next partitions subjected to Pj are remained the same. This causes the increasing of the summation-error by considering the margin of the insertion data tuple and Pj

Table 8: The impact when a data tuple with 64 as its sensitive value is inserted

Id	data	*error*	*error*′	*partition*	*partition*′
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
-1	64	-	10	-	1
4	65	11	11	1	1
5	70	16	8	1	4
6	75	12	13	4	4
7	77	14	15	4	4
8	77	14	15	4	4
9	101	38	34	4	7
10	102	37	35	7	7
11	103	16	17	9	10

as shown in Table 8.

Another case is that the $\triangle d$ is merged with partition Pj after re-applying the algorithm, it can cause such partition breaks into a complete partition satisfying the (k, e)-condition before the ending tuples of the previous Pj. The left-out data tuples will be merged and considered with the consecutive partition of Pj. This will happen only when the cost of the partition breaking is less than the merging and keeping the structure of Pj the same as before subjected to the optimal result. In order to illustrate this case, consider another example in Table 9. From the example, it can be seen that Pj, in this case $\{d_4, d_5, d_6, d_7 d_8\}$, is broken by merging $\triangle d$ to its front-end. Then the partitions become $\{d_{-1}, d_4, d_5\}$ and $\{d_6, d_7, d_8\}$ with lower summation-error.

In summary, we can see that the insertion at the partition border can only merge with the preceding and the consecutive partition. If the data tuple is to be merged with the preceding partition, it is from the fact that the error from extending the partition is less than the error causes by merging it with the consecutive partition. Though, merging $\triangle d$ data tuple to the consecutive partition can also cause such partition to meet the break-point before the previous. Since the (k, e)-condition is met. The structure-change of the insertion

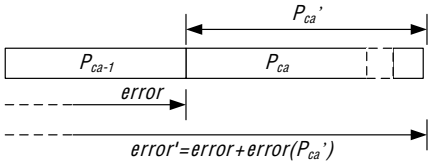
Table 9: The impact when a data tuple with 64 as its sensitive value is inserted, and a new partition is found

Id	data	*error*	*error*′	*partition*	*partition*′
1	54	Infinity	Infinity	0	0
2	55	Infinity	Infinity	0	0
3	56	2	2	1	1
-1	64	-	8	-	1
4	65	11	11	1	1
5	70	16	10	1	4
6	75	12	15	4	4
7	76	13	16	4	4
8	80	17	15	4	7
9	101	37	36	7	7
10	102	35	37	8	7
11	103	19	17	9	7
12	120	36	34	9	10
13	121	37	35	9	10
14	122	21	19	12	13

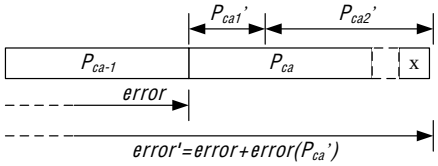
partition is depended on the *error* value of the data tuples, which is processed in greedy manner.

4.2 Incremental Processing

From the previous section, we observe that the impact of an increment by re-applying the algorithm is as follows. First, when the position of the insertion data tuple is determined. The *error* and the *partition* values priori to the position are always remained the same due to the greedy approach. Then, the *error* and the *partition* values of the insertion-position tuple can be determined. Thus, a naïve approach is to compute the *error* and the *partition* values right from the insertion position to the end of the dataset. However, consider the cases where the insertion position is in the very beginning position of the existing dataset, such approach might not be appropriated. As such the complexity of the incremental algorithm in this case will be $O(nn')$ where n' is the number of the remaining data tuples to be processed.



a) Extending partition



b) Breaking partition

Figure 2: Summation-error Computing for Incremental Processing

In this work, we propose a lemma from our observations which will be used to improve the incremental processing as follows.

Lemma 4.1 *For each incremented dataset D', after the completed-alignment partition, the partition in which the data tuples in it are the same as before the insertion, the remaining structure of the partition is the same as priori to the insertion.*

Proof Let d_i be a data tuple after the completed-alignment partition p_{ca}, the partition in which the data tuples in it are the same as before the insertion. In order to obtain the optimal solution, the $error[i]$ of d_i can be only computed in two cases since partition p_{ca} has not been effected by the insertion. First, it is merged to p_{ca} or one of p_{ca} consecutive partitions, or it is combined with the other $d_{i-1}, d_{i-2}, \ldots d_{i-m}$ and forms a new partition. In both cases, $error[i]$ can be determined from the summation between the $error[partition[ca]-1]$, and the difference between the sensitive value of d_i and the sensitive value of the first data tuple in $partition_{ca}$. That is, $error[i] = error[partition[ca]-1] + (d[i] - d[partition[ca]-1])$.

In Figure 2a), the illustration of the first case is shown. It can be seen

that the range of the partition p_{ca} which adds the new data tuple d_i is extended. Thus, its error is formed by the summation-error before p_{ca} and the incremented p_{ca}. The latter case, where a new partition is formed, is illustrated in Figure 2b). Such error computing can be performed in the same approach.

Also, as the structure of the partition is not changed after the complete-alignment partition, we can further update the *error* and *partition* variables as follows. First, the *partition* variable after such position is increasing by one. Since the positions of the remaining is shifted by one insertion. Then, the *error* partition can be computed as mentioned.

For example, in Table 9, it can be seen that the complete-alignment partition is the partition with $\{d_9,\ d_{10}, d_{11}\}$ data tuples. The remaining partition information of the remaining tuples is computed by adding one to it. Then, the $error[12]$ is the addition of the $error[partition[ca] - 1]$, i.e. 17, to $(120 - d[partition[ca] - 1])$, i.e. 120 - 103. Thus, such error is 34. For the rest of the data tuples, the same computing approach can be applied.

From such observation, we can develop an incremental algorithm which computes the *error* and the *partition* as the static algorithm until the complete-alignment is found. Subsequently, the *partition* is computed with $O(n')$ where n' is the number of the remaining data tuples to process. As well as the *partition*, the *error* can be computed with the same cost, while the same result as the re-applying can be guaranteed.

4.3 Algorithm

In this section, we propose an incremental algorithm to preserve the privacy based on the (k, e)-anonymization as shown in Figure 3.

From the algorithm, it can be seen that the efficiency depends on how far the break-point position can be found. Let n be the number of data tuples,

Input:
D': a permuted dataset
k: a minimum number of distinct values threshold
e: a minimum range of values threshold
$\triangle d$: an additional tuple
partition: a partition information from the permuted dataset
error: an error information from the permuted dataset
Output:
$(D + \triangle D)'$: the output dataset, which satisfies the (k, e)-anonymization
partition$'$: the updated partition information
error$'$: the updated error information
Method:

```
1   insert △d into D' based on its sensitive value
2   determine the position of △d, ins_pos
3   copy partition and error into partition' and error' from the first to the position of the ins_pos
4   for i = ins_pos to (D + △D)'.size
5     error'[i] = infinity
6     partition'[i] = partition'[i − 1]
7     for j = 1 to i
8       if distinct({d_j, ..., d_i}) ≥ k and (d_i − d_j) ≥ e then
9         current_error = (d_i − d_j)
10      else
11        current_error = infinity
12      end if
13      temp = error'[j − 1] + current_error
14      if temp ≤ error'[i] then
15        error'[i] = temp
16        partition'[i] = j
17      end if
18    end for
19    if the previous partition comparing with the i position is a complete-alignment then
20      break from the i-loop
21    end if
22  end for
23  determine the remaining partition' and error'
```

Figure 3: Incremental Algorithm

let ins_pos be the position of the insertion, and let n' be the number of the remaining data tuples after the break-point (In Line 20 of Figure 3). The computational complexity of the proposed algorithm is composed by the cost of the insertion determination, finding the optimal solution before the break-point, as well as *error* and *partition* determination after the breakpoint. Thus, the complexity is $O(ins\_pos)+O((n'-ins\_pos)^2)+O(n-n')$. Obviously, if the break-point is found at the end of the dataset, the incremental algorithm will have the same complexity as the non-incremental algorithm, otherwise it can be more efficient.

To clarify our algorithm, let us consider an example. Suppose that the dataset in Table 9 is given. The algorithm first determines that the position of the $\triangle d$, i.e. the fourth position. The $partition'$ and the $error'$ values of the preceding position are the same as their original values. Subsequently, the i-loop is proceeded to determine the optimal solution for each d_i as the non-incremental algorithm. The i-loop will continue until it finishes the processing for the d_{12} data tuple. At the point, the completed-alignment of the partition $\{d_9, d_{10}, d_{11}\}$ is found. Then, the i-loop is broken, and the remaining of the *partition* and the *error* values can be computed using Lemma 1 with $O(n-n')$ cost.

5 Indexing

In order to further improve the efficiency of the proposed incremental algorithm, two indexes are proposed as follows.

First, we propose an index on the distinct sensitive values. Because one of the most high-computation operations in the algorithm is to decide whether the given range of data tuples has enough distinct value to be a valid partition subjected to the k-condition. Additionally, the index can improve the efficiency of

the complete-alignment identification. Since, the incremental algorithm relies on the break-point determination such that, it can stop the $O((n' - ins\_pos)^2)$ processing, and begin the $O(n - n')$ processing. If the cost of the alignment identification is lower, a high efficiency can be obtained.

After the algorithm is proposed, in order to further improve the efficiency of the incremental algorithm, there are a few possible indexes can be created.

Thus, we create an index which its key is the range of the positions. The value of the index is the sequence of the sensitive value of the data tuples in the partition belonged to the key, as well as their number of distinct values. Such index will be in a form of flat-file index, in which each entry is in the form $< (p_i, p_j), ((s_1, s_2, \ldots, s_n), no\_distinct) >$, where p_i and p_j is the range, $(s_1, s_2, \ldots, s_n)$, is the sequence of the sensitive values in the range, and *no_distinct* is the number of distinct values in such range. For example, an index entry for the d_1 data tuple to the d_3 data tuple from Table 9 is $< (1, 3), ((54, 55, 56), 3) >$. It can be seen that the sequence covers data tuples with 54, 55, and 56 as their sensitive values. And, the number of distinct values among them is 3.

Second, given the fact that there could be a very large number of duplicates in real-life dataset. For example, consider the age attribute of the individuals, though the number of data tuples can be varied, age values can be limited to some certain ranges. Thus, we propose to apply a tree-based index, i.e. B^+-tree, which its key is the sensitive value of a data tuple, the value of its is the position of the first value of the key. Formally, the index entry is in the form of $< d_i.sensitive, p_i >$. With this index, we can identify whether the current i-data tuple is the distinct value comparing with the previous value when processing the i-loop in the algorithm.

6 Experiment Results

After our incremental algorithm and indexing are proposed, in this section, we present the experiment results to evaluate our work.

6.1 Configuration

The experiments were conducted on the Adult dataset from UCI Machine Learning Repository Blake & Merz (1998) which has been used to evaluate the (k, e)-anonymization Qing et al. (2007), Procopiuc & Srivastava (2009). The dataset contains 14 attributes over 48,000 data tuples. The "capital-loss" attribute is selected as the sensitive values. Eight of the attributes are selected as the quasi-identifier as in Qing et al. (2007), as well as the data cleansing processes. Thus, the remaining number of the dataset is 1427 tuples. The range of the capitol-loss values is 155 to 3900, with 89 distinct values.

The efficiency is evaluated in term of the execution time when the three parameters change, i.e. the k value, the e value, and the size of the incremental dataset $|\triangle D|$. In each experiment, the dataset will be divided into two equal parts, the first part is used as the static part of the data, while the latter will be used as the incremental data. The incremental part of the dataset will be appended to the static part in a one-by-one basis. Once, all the increment data are appended to the dataset, the execution time is reported. Additionally, the position of the break-point from the proposed incremental algorithm is reported to present the relationship between the position and the efficiency of the algorithm. Such positions are reported into the percentage of the skipped tuples relatively to the size of the dataset. The more the percentage means the more efficiency the algorithm can obtain. Note that the execution time of the incremental algorithm both with/without the proposed indexes will be reported to show their effects. Thus, the execution time reported for the incremental

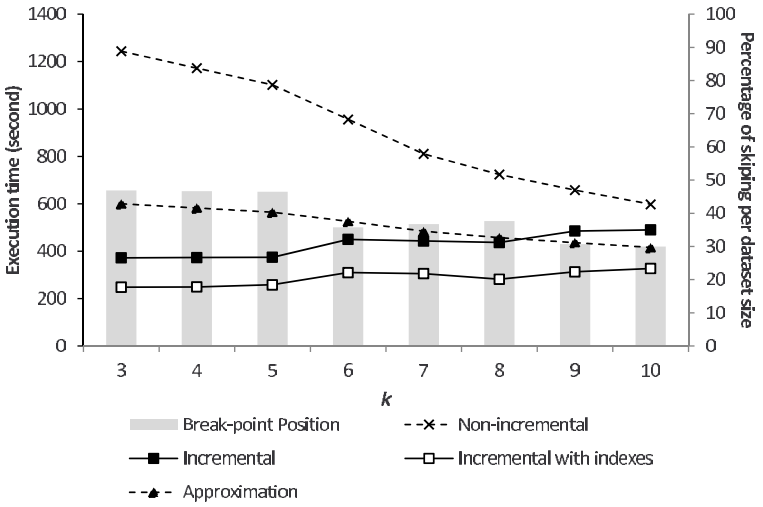


Figure 4: Effect of the k value

algorithm with the indexes consists of the execution time as well as the index update time. In each experiment, the proposed algorithm will be compared with the $O(n^2)$-algorithm in Qing et al. (2007) and the 2-approximation algorithm with $O(n)$ complexity in Procopiuc & Srivastava (2009). The resulting numbers reported are five-time average. Note that the execution time reported for the incremental algorithm with the indexes consists of the execution time as well as the index update time.

6.2 Results and Discussion

6.2.1 Effects of k value

In the first experiment, we evaluate the efficiency of the incremental algorithm when the k value is varied. Such value is varied from 3 to 10 to evaluate its effect. The e value is fixed at 100. At each setting, a set of the increment data of size 10% of the existing dataset is inserted.

In Figure 4, the result of the experiment is presented. Obviously, when the

k value is increased, the execution time of the incremental algorithms are also increased. The rationale behind this is that when the k is higher, the break-point of the complete-alignment is more difficult to be found with regard to the k-condition. In the worst case, the algorithms need to compute the *error* and the *partition* values until the end of the dataset with $O((n' - ins\_pos)^2)$ cost. Thus, the efficiency is degraded. Comparing with the non-incremental algorithm, the fixed computational cost of $O(ins\_pos)$ of the incremental algorithms before the ins_pos-position is higher. We can see the effect of such cost when the k values is very high, the execution time of the incremental algorithm is even a bit higher. However, in Procopiuc & Srivastava (2009), the authors suggested that the k value should be set at lower than 6 in general. Also, it can be seen that the incremental algorithm with the proposed indexes is more efficient than the without-indexes version of it significantly. The algorithm with the indexes is even more efficient than the 2-approximation algorithm which has $O(n)$ complexity, meanwhile the optimal solutions can be obtained.

6.2.2 Effects of e value

In this experiment, we evaluate the efficiency of the incremental algorithm when the range of the tolerate error e is changed. The e will be increased from 20 to 220 to evaluate its effect. The k is fixed at 5, and $|\triangle D|$ at 10%. From the result in Figure 5, it can be seen that the proposed incremental algorithm is more efficient than the non-incremental algorithm. These results are obvious particularly when the e value is set at 100 or lower in which the execution time of the non-incremental algorithm is more than 3-times of the proposed algorithm. Such gaps are caused by the discovery of the complete-alignment partition, and thus prevent the quadratic part of the algorithm to be executed till the end of the input. The rationale behind the efficiency degrade, after the e value is higher than 100, is each partition becomes larger. And, thus

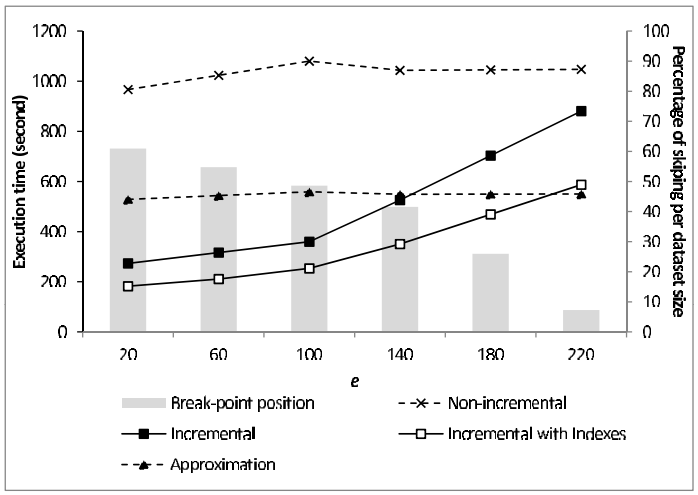


Figure 5: Effect of the e value

the complete-alignment is more difficult to be found as it can be seen the percentage of the skipped tuples from the break-point positions. Obviously, such percentage decreases when both the k and e parameters are increased because the mentioned reason. Also, it can be seen that the incremental algorithm with indexes can outperform the 2-approximation algorithm when the e value is not extremely high. Given the fact that the optimal solutions can be obtained instead of the approximate results, our proposed algorithm is very efficient.

6.2.3 Effects of $|\triangle D|$

In the last experiment, we evaluate the efficiency of the incremental algorithm when the size of the incremental dataset ($|\triangle D|$) is varied. The variation is set at the percentage of the whole data to be appended, i.e. 6, 8, 10, 12 and 14 %. We set the k and e values at 5 and 100 respectively. Figure 6 shows the result

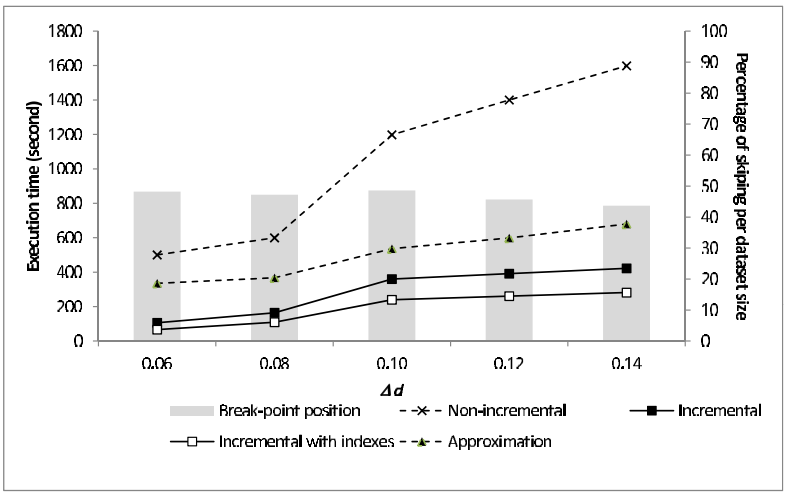


Figure 6: Effect of the $\triangle D$

of this experiment. It can be seen that when the size of the appended data is increased, the gap between the execution time of both algorithms is increased. Obviously, the effect of the incremental processing is accumulated when the size of $\triangle D$ is increased. Particularly, when we consider the incremental algorithm with the proposed indexes, the effect is even more obvious. And, in all settings, both of the incremental algorithms outperform the approximation algorithm.

7 Conclusion

In this paper, we have proposed an observation on the privacy preservation based on the (k, e)-anonymization when the data increment is considered. We have subsequently, proposed an incremental algorithm for the problem. Such algorithm can process the data incrementally while exactly the same solutions as the non-incremental algorithm can be guaranteed. The proposed algorithm

can be more efficient than the non-incremental algorithm when the complete-alignment partition is discovered before the end of the data input is to be processed. Additionally, two indexes are proposed to further improve the efficiency of the algorithm. From the experiments, it has been found that the incremental algorithm is much more efficient than the non-incremental algorithm when the k or e value is not extremely high. When the size of the increment data is increased, the efficiency gap between the proposed algorithm and the non-incremental algorithm is larger. Also, the incremental algorithm with the proposed indexes can outperform the $O(n)$ approximation in most of the experiment settings. These can be summarised that the proposed algorithm can be very efficient in the real-world situations. In our future work, we will further investigate on the multiple bulk-insertion scenarios. In which, not only the efficiency should be considered, but also the concurrency control issue.

References

Blake, C. & Merz, C. (1998), 'UCI repository of machine learning databases'. **URL:** *http://www.ics.uci.edu/∼mlearn/MLRepository.html*

Byun, J.-W., Li, T., Bertino, E., Li, N. & Sohn, Y. (2009), 'Privacy-preserving incremental data dissemination', *Journal of Computer Security* **17**(1), 43–68.

Denning, D. E., Akl, S. G., Heckman, M., Lunt, T. F., Morgenstern, M., Neumann, P. G. & Schell, R. R. (1987), 'Views for multilevel database security', *IEEE Transactions on Software Engineering* **13**(2), 129–140.

Domingo-Ferrer, J. & Torra, V., eds (2004), *Privacy in Statistical Databases*, Vol. 3050 of *LNCS*, Springer, Berlin Heidelberg.

Fung, B. C. M., Wang, K., Fu, A. W.-C. & Pei, J. (2008), Anonymity for continuous data publishing, *in* 'Proceedings of the 11th international conference on Extending database technology: Advances in database technology', EDBT '08, pp. 264–275.

Fung, B. C., Wang, K. & Yu, P. S. (2007), 'Anonymizing classification data for privacy preservation', *IEEE Transactions on Knowledge and Data Engineering* **19**(5), 711–725.

Jajodia, S., Samarati, P., Subrahmanian, V. S. & Bertino, E. (1997), A unified framework for enforcing multiple access control policies, *in* 'SIGMOD '97: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data', ACM Press, pp. 474–485.

Li, N., Li, T. & Venkatasubramanian, S. (2009), 'Closeness: A new privacy measure for data publishing', *IEEE Transactions on Knowledge and Data Engine'ering* **22**(7), 943–956.

Machanavajjhala, A., Gehrke, J., Kifer, D. & Venkitasubramaniam, M. (2006), ℓ-diversity: Privacy beyond κ-anonymity, *in* 'ICDE '06: Proceedings of the 22nd International Conference on Data Engineering', IEEE Computer Society, Washington, DC, USA, p. 24.

Meyerson, A. & Williams, R. (2004), On the complexity of optimal k-anonymity., *in* 'Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems', ACM, pp. 223–228.

Pei, J., Xu, J., Wang, Z., Wang, W. & Wang, K. (2007), Maintaining k-anonymity against incremental updates, *in* 'Proceedings of the 19th International Conference on Scientific and Statistical Database Management', SSDBM '07, pp. 5–.

Procopiuc, C. M. & Srivastava, D. (2009), Efficient table anonymization for aggregate query answering, *in* 'Proceedings of the 2009 IEEE International Conference on Data Engineering', IEEE Computer Society, Washington, DC, USA, pp. 1291–1294.

Qing, Z., Koudas, N., Srivastava, D. & Ting, Y. (2007), Aggregate query answering on anonymized tables, *in* 'Proceedings of the 2007 IEEE International Conference on Data Engineering', IEEE Computer Society, Washington, DC, USA, pp. 116 – 125.

Samarati, P. (2001), 'Protecting respondents' identities in microdata release', *IEEE Trans. on Knowl. and Data Eng.* **13**(6), 1010–1027.

Samarati, P. & Sweeney, L. (1998), Generalizing data to provide anonymity when disclosing information (abstract), *in* 'Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems', PODS '98, pp. 188–.

Seisungsittisunti, B. & Natwichai, J. (2009), Incremental privacy preservation for associative classification, *in* 'Proceeding of the ACM first international workshop on Privacy and anonymity for very large databases', PAVLAD '09, pp. 37–44.

Sweeney, L. (2002*a*), 'Achieving k-anonymity privacy protection using generalization and suppression', *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* **10**(5), 571–588.

Sweeney, L. (2002*b*), 'k-anonymity: a model for protecting privacy', *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems* **10**(5), 557–570.

Truta, T. M. & Campan, A. (2007), K-anonymization incremental maintenance and optimization techniques, *in* 'SAC '07: Proceedings of the 2007 ACM Symposium on Applied Computing', ACM, New York, NY, USA, pp. 380–387.

Wong, R. C.-W., Li, J., Fu, A. W.-C. & Wang, K. (2006), (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing, *in* 'KDD '06: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM Press, New York, NY, USA, pp. 754–759.

Xiao, X. & Tao, Y. (2007), M-invariance: towards privacy preserving re-publication of dynamic datasets, *in* 'Proceedings of the 2007 ACM SIGMOD international conference on Management of data', SIGMOD '07, pp. 689–700.

Zhou, B., Han, Y., Pei, J., Jiang, B., Tao, Y. & Jia, Y. (2009), Continuous privacy preserving publishing of data streams, *in* 'Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology', EDBT '09, pp. 648–659.