Chapter 2

Related Work

The beginning of this chapter discusses about surveying the privacy problems in real world. Then the principle privacy preservation models will be discussed. Next, this thesis highlights the (*k*, *e*)-anonymous model [10] with the purpose of the minimum summation of error. Thus, the minimum summation of error algorithm is discussed. At last, some incremental privacy preservation problems and the solutions will be discussed.

2.1 Privacy Breach Problems.

From the time that the Internet has been invented, its usage has increased from time to time. In March 2013, the number of users is 2,749 million that is 38.8% of the world population [32]. Indeed, data privacy is one of the important topics to be addressed currently because the social network is growing up at vary fast speed with many activities are involved, such as, work collaboration, family communication, or even commercial advertiser. These activities may increase the quantity of the data collection and sharing of specific personal information for improving the convenience of an information management and access. However, the privacy problem has been increased because of the convenience for the data acquisition.

There are quite a few countries that have not concerned about privacy issues in the Internet environment as much as the other developed countries. Generally, in some developed countries, they have the act of legislation for a long time such as, Privacy Act of 1974 (US), Electronic Communications Privacy Act (US), Privacy Act 1988 (AUS), Data Protection Directive (EU), Data Protection Act 1998 (UK), and Data protection (privacy) laws in Russia.

However, those developed countries still have privacy problems. From the survey on anonymity, privacy, and security from the Pew Research Center's Internet Project Survey in America by Carnegie Mellon University in [33], there are some interesting statistics about the privacy problem such as 21% of emails or social network

accounts was controlled or accessed by someone else, 12% of online users was tracked or annoyed, 11% of important personal information was stolen online, and 6% of the internet users was cheated and lost some money. These statistics are shown that the privacy breach problems occur in the real world. Next, there were some interesting cases of privacy breaches from news that happen in the year 2009-2014 as follows.

The first case happened in Canada. In March 2014, a man living in a Saskatoon city was asked to register a medical card of another city that he had not been there before. Next, that man received social insurance information of a stranger. It seems that personal information was breached by the government. After the case, he started to protect of his data privacy and worry about the government operation [34].

The second case has occurred since December 2009. A laptop of a Florida-based health insurer (AvMed) was lost. The laptop contains the unencrypted patient information about tens of thousands AvMed customers. This case was discussed on the district court. After that, AvMed has agreed to be responsible for victims who lost their money involving the breach. Additionally, the firm promised to improve the security system such as new password protocols, install disk encryption and GPS tracking tools on its laptops [35].

The last example case occurred in Australia, the privacy of the customers of a telephone organization named Telstra was compromised. The personal information of 15,775 customers ware showed up with a Google search between February 2012 and May 2013. The disclosed information was the customer names, telephone numbers, and, in some cases, home and business addresses. Following the breach, Telstra agreed to replace the software platform for handling the privacy information, and they will change some contracts with third-parties that relate with this privacy breach [36].

2.2 Principle privacy preservation models

The previous section indicated that the privacy policy of the organizations must be managed carefully. However, the best practice in term of the policy might not be enough. Therefore, some technology should be introduced to overcome the issue.

Firstly, security view management [38, 40] could address this problem. Creating different views for different usages is one of approaches to overcome this issue. For example, when dissemination is required, not all the data are released, and only part which is safe from the privacy breaches.

When the security mechanism is concerned, the access-control methods [38, 40] are employed and have been proposed substantially, among database security research community. These methods are based on the authorization mechanisms that are positive authorization and negative authorization. For the positive authorization, database administrators require to specify which group of the users can access to the data, while the negative authorization demands the administrator to specify which group of the users can “not” access the data.

Statistical security-control [28, 38] is another approach for addressing the privacy problems. Before the data are to be published for any purpose, noise values are added into the original dataset to make the data 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.

A basic method to protect the privacy of information is removing the identifiers such as ID or name. However, when all identifiers have been removed, unfortunately, there could be other data from other source that can link or overlap together to identify individuals. The linking or overlapping can be established by common attributes between the two data. For example, consider the dataset in Table 2.1. Suppose that the dataset is published carefully by a hospital to be analyzed via a data analysis company. By considering this dataset alone could misjudge that the privacy of the individuals has been already preserved by the removal of the identifiers. However, suppose that there is

another dataset that is released publicly for voting purpose as shown in Table 2.2. If an adversary get private information about a man named “Somchai” who lives in an area with postal code “50200”, and his age is approximately 50 years-old. The opponent can merge the two datasets together using postal code and age attributes, subsequently, his medical condition will be disclosed.

Table 2.1: Linkable dataset

Postal Code	Age	Sex	Disease
50211	20	Female	Fever
50211	24	Female	Fever
50211	32	Female	Flu
50202	41	Male	Flu
50200	50	Male	Cancer

Table 2.2: Published dataset

Name	Postal Code	Age	Sex
Ratanapat	50211	20	Female
Ratchaneekorn	50211	24	Female
Wichaya	50211	32	Female
Chontipan	50202	41	Male
Somchai	50200	50	Male

So, we review some of the techniques with regard to this situation where the privacy can be compromised, that is *k*-Anonymity, *l*-diversity, *t*-closeness and (*k*, *e*)-anonymity [3, 4, 6, 7, 8, 10]. Each model has advantages and disadvantages for this issue. In addition to the conditions of such model and the transformation must consider the tradeoff between privacy and utility. So, each model will be presented with its utility, such as information lossy and lossless conditions [14], the data mining workload [2, 15, 16], classification [17, 18], and aggregation query answering [10].

***k*-Anonymity model [1, 7, 8]**

k-Anonymity [1, 7, 8] is first well-known privacy model which can prevent the disseminated data to be linked. A dataset is said to satisfy the anonymity, if for every tuple in the dataset, there are another *k*-1 tuples which are indistinguishable from the tuple for all “linkable” attributes. The examples of the linkable attributes in the previous case are postal code, age, and sex. If a dataset does not satisfy the standard, the dataset

can be generalized, using the attribute's generalization tree (called hierarchy or taxonomy for some works) to generalize, that dataset until the standard is reached. For example, from Table 2.1, the dataset can be generalized into a *2*-Anonymity dataset by generalizing the last digit of the postal code to be unreadable as shown in Table 2.3. Nevertheless, data quality issue in the transformation processes is required to be addressed, i.e. the generalized dataset is supposed to have enough "quality" to be used by the designate data processing.

Table 2.3: *2*-Anonymity dataset

Postal Code	Age	Sex	Disease
5021*	20-35	Female	Fever
5021*	20-35	Female	Fever
5021*	20-35	Female	Flu
5020*	36-50	Male	Flu
5020*	36-50	Male	Cancer

Sweeney [7] has defined first information theoretic metric for quantifying the data quality of the generalized datasets, i.e. the amount of distortion in a generalized dataset by using the ratio of the number of level of the generalized value in the dataset to the height of the attribute's hierarchy, which reports the amount of generalization and measures the value distortions. Precision of a generalized dataset is one minus by the sum of all values distortions that is normalized (divided) by the total number of values, named a precision metric. This metric has been applied to many works that use the *k*-Anonymous model.

Many metrics were developed for evaluating the quality of generalized dataset, such as the average *q**-block size metric [3, 4, 9] that evaluates the quality using the average number of size of the generalized quasi-identify block, and the discernibility metric is equals to the summation of squares of the *q**-block size. Some metrics were created for evaluating the utility of generalized dataset, such as the classification metric CM [14, 16] was defined for measuring the utility of the classification workload by using the summation of the individual penalties of majority class for each tuple in the generalized dataset normalized by the total number of tuples, and the C_{FCM} [18, 21] that was applied from CM for measuring the utility of associative classification computing.

***l*-diversity model [4]**

This model uses generalization for privacy preserving. The approach of this model is to generalize the tuple values of quasi-identifier attributes to the same value by partitioning a set of data tuples into partitions. Sensitive value in the partition also has minimum *l* distinct value. This model appropriates for the scenarios that the sensitive value is categorical.

Table 2.1: *3*-diverse dataset after generalization

Quasi-identifiers			Sensitive
Age	Zip code	Gender	Disease
30-40	271**	*	Gastritis ulcer
30-40	271**	*	Gastritis
30-40	271**	*	Stomach cancer
41-50	272**	*	Gastritis
41-50	272**	*	Flu
41-50	272**	*	Bronchitis
51-60	276**	*	Bronchitis
51-60	276**	*	Pneumonia
51-60	276**	*	Stomach cancer

A disadvantage of *l*-diversity moreover using generalization is when the sensitive attribute is skewed. If the sensitive value has 2 distinct values, one has large frequency and another one has low frequency. Amount of partition that is satisfied *2*-diversity will depend on the frequency of the low one. Thus, some partitions may have too many tuples and they can be over-generalized.

Table 2.5 shows the skewed sensitive values dataset. The dataset can derive that who has age between 30 and 40 and has his/her three-first character of zip code equal to 271, then he/she can have HIV at 7/9 percent. This is happened because a frequency of HIV tuples is very high.

Table 2.2: *l*-diversity skewed sensitive attribute values

Quasi-identifiers			Sensitive
Age	Zip code	Gender	Disease
30-60	271**	*	HIV
30-60	271**	*	HIV
30-60	271**	*	HIV
30-60	271**	*	HIV
30-60	271**	*	HIV
30-60	271**	*	HIV
30-60	271**	*	HIV
30-60	271**	*	Pneumonia
30-60	271**	*	Stomach cancer

(*c*, *l*)-diversity model [4]

This model had added *c* value to *l*-diversity model. The *c* value is used to control the frequency of the first sensitive value (after sort frequency decreasingly) that has maximum frequency of each partition. The condition is that the frequency of the first sensitive value must be equal or less than *c* time of the sum of another sensitive value frequency. Table 2.6 show the generalized dataset that satisfies (*0.5, 3*)-diversity. This dataset have 3 distinct values i.e. HIV, Flu, and Gastritis. And each distinct value have frequency at most 0.5, for example, frequency of "HIV" is 4/9, 4 is number of "HIV", 9 is number of total values.

Table 2.3: (*0.5*, *3*)-diversity dataset after generalization

Quasi-identifiers			Sensitive
Age	Zip code	Gender	Disease
30-40	271**	Man	HIV
30-40	271**	Man	HIV
30-40	271**	Man	HIV
30-40	271**	Man	HIV
30-40	271**	Man	Flu
30-40	271**	Man	Flu
30-40	271**	Man	Flu
30-40	271**	Man	Gastritis
30-40	271**	Man	Gastritis

***t*-closeness model [3]**

t-closeness is another model that can solve the skew of sensitive attribute problem. With this model, the distributions of a sensitive attribute in any quasi–

identifier group are required to be close to the distribution of the attribute in the whole dataset. The *t*-closeness model uses the Earth Mover Distance (EMD) function [3, 19] to measure the closeness between two distributions of sensitive values, and demands the closeness to be at most *t*.

Nevertheless, *t*-closeness has several limitations and weaknesses. First, it lacks of flexibility of specifying different protection levels for different sensitive values as in [3]. Second, the EMD function is not suitable for preventing attribute linkage on numerical sensitive attributes. Third, enforcing *t*-closeness would greatly degrade the data utility because the distribution of sensitive values is required to be the same in all QID partition [3, 19].

For the data quality after transformation, *l*-diversity, (*c*, *l*)-diversity, and the *t*-closeness model, use the same utility measurements that are the generalization height metric [7], average of *q**-block size [3, 4], and discernibility metric [3, 4].

(*k*, *e*)-anonymous model [10]

The overview of this model, which our work is based on it, is to disassociate the relationship between a quasi-identifier and a numerical sensitive attribute value by partitioning a set of data tuples into partitions and shuffling their sensitive values with each partition. By the *k* value is minimum distinct value of sensitive attribute in every partitions, the *e* value is minimum length (maximum – minimum) of sensitive values in every partition.

Table 2.7: An example dataset

	ID	Quasi-identifiers			Sensitive
Tuple ID	Name	Age	Zip code	Gender	Salary
1	Alex	35	27101	M	54000
2	Bob	38	27120	M	55000
3	Carol	40	27130	M	56000
4	Debra	41	27229	F	65000
5	Evan	43	27269	F	75000
6	Frank	47	27243	M	70000
7	Gary	52	27656	M	80000
8	Henry	53	27686	F	75000
9	Ina	58	27635	M	85000

Table 2.8: An example (*k*, *e*)-anonymous dataset after permutation

	Quasi-identifiers			Sensitive
Tuple ID	Age	Zip code	Gender	Salary
1	35	27101	M	55000
2	38	27120	M	55000
3	40	27130	M	54000
4	41	27229	F	65000
5	43	27269	F	70000
6	47	27243	M	75000
7	52	27656	M	75000
8	53	27686	F	80000
9	58	27635	M	85000

Table 2.7: An example in Table 2.7, Tuple ID that is key attribute, and Name is identifier attribute. In addition, age, Zip code and Gender are quasi-identifier attribute, and Salary is sensitive attribute. The figure shows 9 tuples in this dataset. The dataset was transformed before reveal to be a dataset in Table 2.8, which is an sample dataset transformed by permutation after removing Name attribute, and this dataset satisfies (*k*, *e*)-anonymity when *k* is 3 and *e* is 2000.

The Minimum Summation of Error Algorithm for (*k*, *e*)-anonymous model

Based on (*k*, *e*)-anonymous model, an aggregation query answering is the main utility. An acceptable result of aggregation query answering depends on the partitioning step, which can be seen in the Chapter 1. The partitioning with minimum of summation of error purpose is one from two proposed solutions in [10]. The minimum summation of error algorithm complexity is $O(n^2)$, and the algorithm gives the partitioned dataset that has relative error of range queries and error of arbitrary queries to close with another algorithm (minimum maximum of error algorithm). The another algorithm complexity, however, is $O(n^6)$. Hence, this study aims to advance the minimum summation of error algorithm for incremental privacy breach problem.

Input:
D: a dataset that is sorted by sensitive values
k: a minimum number of distinct values threshold
e: a minimum error of values threshold
Output:
partition: the partition information
error: the error information
Method:

```
1   error[0] = 0
2   partition[0] = 0
3   for i =1 to D.size
4     error[i] = infinity
5     partition[i] = partition[i - 1]
6     for j = 1 to i
7         if distinct({d_j,...,d_i}) ≥ k and (d_i – d_j) ≥ e then
8           current_error = (d_i – d_j)
9         else
10          current_error = infinity
11        end if
12        temp = error[j - 1] + current_error
13        if temp < error[i] then
14          error[i] = temp
15          partition[i] = j
16        end if
17    end for
18  end for
```

Figure 2.1: Minimum Summation of Error Algorithm

The algorithm is presented in Figure 2.1. In the method, let *D* be an input dataset that has been sorted by sensitive values increasingly. Let *k* be a minimum number of distinct values threshold. Let *e* be a minimum error of values threshold. Let *partition* be an integer array variable that contains information of partitioning index. Let *error* be an integer array variable that contains information of accumulated error values, error is result of maximum sensitive value decreased by minimum sensitive value in a partition. The dataset *D*, *e* value, and *k* value are the inputs of the algorithm. The *error* array and *partition* array are output of the algorithm. After the algorithm is applied to the dataset, the *partition* array will be used to partition the input dataset *D.*

The algorithm computes the optimal summation of error in each *i*-loop. The error value is maintained in the error variable. The *error*[*D.size*] contains the

summation error of the optimal solution. The algorithm applies a greedy approach which can be seen in the *j* loop. The current partition ($\{d_j,\ldots,d_i\}$) subjected to the *k* and *e* values will be compared to the best-known error for such index *i*. If the current error is less than the existing value, it is stored as the new error in *error*[*i*]. Also, the index *j* which led to the lower error is held in partition[*i*]. This comparison will contribute to the sub-loop $O(n)$ for each *i* loop. The complexity of the algorithm, thus, becomes $O(n^2)$.

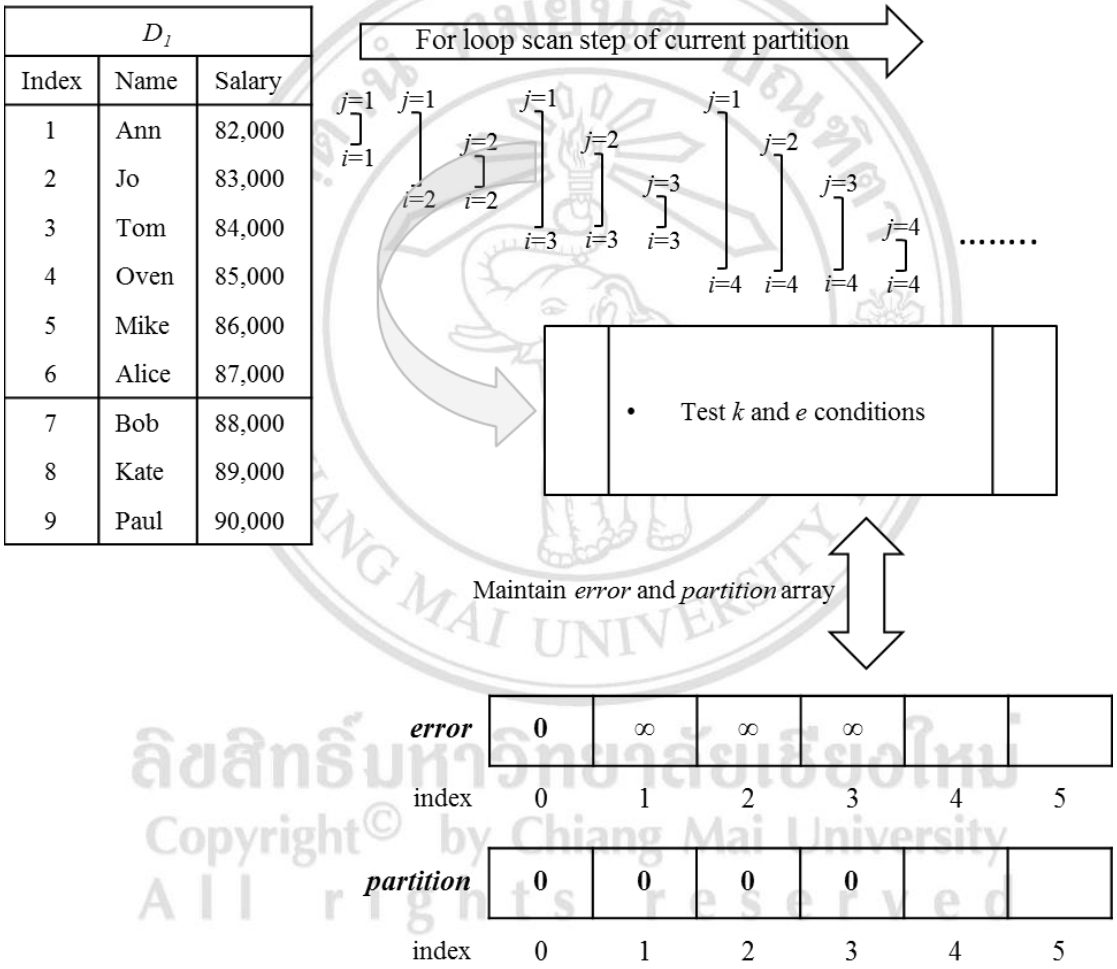

Figure 2.2: Illustration of the minimum summation of error algorithm

Figure 2.2 illustrates the minimum summation of error algorithm. The dataset D_1 are the original dataset. The algorithm scans all possible partition of D_1 one by one. Each single partition for consideration is called current partition. This is execution of

the for loop statements using variable *i* and *j* in the 3rd and 6th lines as shown in the algorithm in Figure.2.1. The steps of loop scanning are shown in the top of the example. Each step will consider the current partition, the current partition is the current partition is evaluated by *k* and *e* conditions. If the current partition does not pass the conditions, a variable *current_error* is set to be infinity by the statement in the 10th line. In case that the current partition pass the conditions, the *current_error* is set to be $d_i[D_n][S] - d_j[D_n][S]$ by the statement in the 8th line.

Next, the algorithm determines the summations of error at index *i*, and such value is stored in the *temp* variable as show in the statement in the 12th line of Figure 2.1. Then the algorithm maintains the minimum summation of error and partitioning index in *error* array and *partition* array by comparing *temp* and *error* at index *i* as show in the if statement in 13th line. If the *temp* value is less than the *error* at the index *i,* then the *error* value at index I is replaced by the *temp* value, and the *partition* value at index i is replaced by the *j* value as shown in the 14th and 15th lines respectively. After the *error* array and *partition* array are updated, the algorithm proceeds to the next partition until the end of the loops.

Table 2.9: An Example dataset to demonstrate minimum summation of error algorithm

D		*Error*	*Partition*
Index	Sensitive Value		
1	84,000	Infinity	0
2	86,000	Infinity	0
3	87,000	3,000	1
4	88,000	4,000	1
5	89,000	5,000	1
6	90,000	5,000	4

An example dataset to demonstrate the minimum summation of error algorithm is shown in Table 2.9. The quasi-identifier attribute is hidden. Suppose that the *k* and *e* values are set at 3 and 2000. After the algorithm is applied to the dataset, result of *error* and *partition* is shown in third and fourth columns of Table 2.9. In order to present the partitioning information, the *partition* array is to be scanned backward as shown in the

figure. The first partition can be determined by scanning at the last tuple. In the figure, the last tuple of D_1 has its index number equal to 6, and the value in the *partition* array of such index is 4 as shown in the right-most column in the figure. Therefore, the first partition is the set of tuples $\{d_6, d_5, d_4\}$. Then, the second partition can be determined by scanning the tuple of index number 3 (the most index number less than the previous partition, i.e. 4), it can be seen that the value in *partition* array of the tuple index is 1. Hence, the second partition is the set of tuples $\{d_3, d_2, d_1\}$.

2.3 Incremental privacy preservation problems and the solutions

An incremental data processing problem on the privacy preservation can be described as follows. In general, when the new data are added to the original data, the whole data should update with respect to the conditions of a privacy preservation model. The re-process of privacy reservation algorithm with whole data are one choice of the solution but this could be inefficiency. This section is going to introduce ideas and methods to solve efficiently with the privacy preservation in case of an incremental data issue.

Incremental processing and indexing for (*k*, *e*)-anonymization

Based on (*k*, *e*)-Anonymous model, after a dataset was transformed with respect to *k* and *e* conditions, and the minimum summation of error purpose. A new tuple may be added to the dataset. The added dataset can be re-processed by the minimum summation of error algorithm, but the re-processing is inefficient. Natwichai et al. [25] improved an efficient algorithm for reducing an execution time and the result still is optimal for the purpose. The method for observation is discussed as follows. First, all potential new tuples are defined, and then an impact of each added tuple is observed. The idea for improving an efficient algorithm comes from the conclusion of observations. The impacts indicate that the new tuple can be categorized into two cases.

For the first case, a new tuple can be added to the "in-range" of any partition. The Table 2.10 shows the example of adding in in-range case, i.e. the new tuple is added to a partition, inserted after the begin tuple and before the end tuple of a partition.

There might be two impacts after adding new tuple in the first case. Firstly, the borders of close partitions are unchanged as shown in Table 2.10. Secondly, the borders of previous partition are changed as shown in Table 2.11.

Table 2.10: An example dataset demonstrates a border unchanged impact of adding new tuple in an in-range case

D		*Error*	*Error'*	*Partition*	*Partition'*
Index	Sensitive Value				
1	84,000	Infinity	Infinity	0	0
2	85,000	Infinity	Infinity	0	0
3	86,000	2,000	2,000	1	1
4	88,000	4,000	4,000	1	1
5	89,000	5,000	5,000	1	1
new tuple	90,000	-	4,000	-	4
6	91,000	5,000	5,000	4	4
7	94,000	8,000	8,000	4	4
8	96,000	10,000	9,000	4	-
9	97,000	8,000	8,000	7	7

Table 2.10 demonstrates the impact of adding new tuple in an in-range case which the borders of closed partitions are unchanged. In the Table 2.10, *error'* and *Partition'* contain the information after re-processes the minimum summation of error algorithm. That can be seen, that the information before the added tuple is stable.

Table 2.11 demonstrates the impact of adding a new tuple in an in-range case which the borders of closed partitions are changed. However, after partitioning, the end border of first partition is move below.

For the second case, a new tuple is added at the border of any partition. The figure 2.6 illustrates an instance of such case, i.e. the new tuple is added at between two partitions. After the adding, the added tuple will be merged with nearest partition by decided on sensitive values of close tuples. If the added tuple merged with previous partition, there is no impact on the rest of partitions. In contrast, if the added tuple merged with next partition, the borders of further partitions may be impacted.

Table 2.11: An example dataset demonstrates a borders changed impact of adding new tuple in in-range case

D		*Error*	*Error'*	*Partition*	*Partition'*
Index	Sensitive Value				
1	83,000	Infinity	Infinity	0	0
2	84,000	Infinity	Infinity	0	0
3	85,000	2,000	2,000	1	1
4	87,000	4,000	5,000	1	1
5	90,000	5,000	7,000	1	1
new tuple	91,000	-	6,000	-	4
6	92,000	5,000	7,000	4	5
7	94,000	8,000	9,000	4	5
8	96,000	10,000	11,000	4	-
9	97,000	8,000	10,000	7	7

Table 2.12: An example dataset demonstrates a previous marge impact of adding new tuple in border case

D		*Error*	*Error'*	*Partition*	*Partition'*
Index	Sensitive Value				
1	83,000	Infinity	Infinity	0	0
2	84,000	Infinity	Infinity	0	0
3	85,000	2,000	2,000	1	1
new tuple	86,000	-	3,000	-	1
4	90,000	7,000	7,000	1	1
5	92,000	9,000	8,000	1	-
6	94,000	6,000	7,000	4	4
7	95,000	7,000	8,000	4	4
8	96,000	8,000	10,000	4	6
9	97,000	8,000	9,000	7	7

Table 2.12 demonstrates the impact of adding new tuple in a border case which new tuple is merged to the previous partition. That can be seen, that the information before the added tuple is stable and the borders of other partitions are stable.

Table 2.13: An example dataset demonstrates a next marge impact of adding new tuple in border case

D		*Error*	*Error'*	*Partition*	*Partition'*
Index	Sensitive Value				
1	83,000	Infinity	Infinity	0	0
2	84,000	Infinity	Infinity	0	0
3	85,000	2,000	2,000	1	1
new tuple	89,000	-	6,000	-	1
4	90,000	7,000	7,000	1	1
5	92,000	9,000	8,000	1	-
6	94,000	6,000	7,000	4	-
7	95,000	7,000	8,000	4	-
8	96,000	8,000	10,000	4	6
9	97,000	8,000	9,000	7	6

Table 2.13 demonstrates the impact of adding new tuple in a border case which new tuple is merged to the next partition. That can be seen, the information before the added tuple is stable. But the end border of merged partitions is moved up.

According to Minimum Summation of Error Algorithm from Figure 2.1, the algorithm improved the efficiency based on the observations. The algorithm starts the scanning not at the beginning, but it begins at the position of a new tuple. The algorithm then scans until the borders of further partitions are complete-alignment comparing with the original. Therefore, there is a possibility that algorithm may not scan until the end tuple.

Furthermore, this study introduced an index of the partitions for improving the efficiency of the algorithm. Each index composed of two components. The first component describes the range of the positions with respect to the key (sensitive data value) starting from the first tuple of the partition. The second component can be divided into 2 parts. First part shows the sensitive values that are sorted in sequence with respect to the key. The second part explains quantity of the values in the partition. For example, from Table 2.13, the index of first partition is <(1, 3), ((54, 55, 56), 3)>.

This index can be useful when comparing the new partition to the original partition for being complete-alignment condition.

Incremental privacy preservation for associative classification

Bowonsak and Juggapong [21] studied on preserving an incremental-data scenario problem of *k*-Anonymity model with regard to a data mining task, i.e. associative classification. Such classification model works on support and confidence scheme as association rules, but having a designate attribute as class label. The study focused on the characteristics of a proven heuristic algorithm, Minimum Classification Correction Rate Transformation (MCCRT) [12, 18] in the data incremental scenarios theoretically.

This study discovered a few techniques to reduce the computational complexity for the incremental problem comparing with the re-processing of MCCRT's algorithm. The proposed techniques not only reduce the complexity, but they also can generate exactly the same outputs as the MCCRT's. The introduced techniques generalized only additional tuples, and then added them into the original generalized dataset. After adding generalized tuples, the whole dataset might be adjusted generalization level. From now on, the MCCRT's algorithm is going to be discussed for the basic understanding of proposed technique.

First, the concept of Classification Correction Rate value (CCR) which is being used in MCCRT's algorithm, is described. The CCR of each attribute is determined as follows. For each attribute, the set of one-literal classification rules is derived for satisfying *minsup* and *minconf*. Subsequently, each tuple is classified into the predicted class label from the derived rules, comparing the predicted class label with the actual class label. Next, the ratio between the number of the correct prediction and the total number of the tuples is computed as the CCR of such attribute.

For the first step of MCCRT's, the algorithm calculates CCR of each attribute. Next, an attribute will be generalized by own generalization tree from the bottom level to the top level of the tree. The sequence of attributes to be generalized is ordered from

the minimum to maximum CCR values respectively. Generalizing process is going to be stopped when the dataset satisfied *k* condition.

It is supposed to be noted that when adding new tuples, CCR value of each attribute may be changed, so the order of generalizing attributes may be varied on changed CCR value for each attribute. After an impact of adding tuples that affected/unaffected CCR values is evaluated, there are two effects on the order of generalizing attributes. First effect, the order of generalizing attributes is unchanged. It can point out at two cases. For the first case, if the whole dataset does not satisfy *k* condition, generalization level will be increased with respect to MCCRT's algorithm. The latter case, if the whole dataset satisfies *k* condition, generalization level may be decreased. Second effect, the order of generalizing attributes is changed. If the order is changed in further part of the last generalized attribute, and in previous part of the last generalized attribute, there is no impact. However, the whole dataset has to be checked to satisfy *k* condition as same as first effect. In the other hand, if the order is changed not same as the above, the changed order is focused on first attribute having generalization level that is not at the top. Former generalization levels of further attributes cannot be used. Thus, the generalization levels are set to be bottom or zero level. Nevertheless, the whole dataset has to be checked to satisfy *k* condition as same as first effect.

An efficient quasi-identifier index based approach for privacy preservation over incremental datasets on cloud.

In [37], According to the technique based on *k*-Anonymity model, after a dataset is satisfied *k* condition, a group of tuples having same values is called "quasi-identifier group". Based on cloud computing, if the whole dataset is distributed randomly to each node in the cloud, some tuples are added into the system inefficiently because all nodes have to compute for constrains of the model. This study indicated that if the whole dataset is distributed properly to each node in the cloud, and creating the approach index, incremental process is more efficient because the computation may occur in some nodes; do not occur in all nodes.

The proposed technique of distributing can be done by each node in the cloud containing similar quasi-identifier group. By similar quasi-identifier group, it can be calculated by Equation 1.

$$dist(qid^x, qid^y) \triangleq \sqrt{\sum_{i=1}^{m} d^2(q_i^x, q_i^y)} \quad (1)$$

Where $dist(qid^x, qid^y)$ is the distance between q_i^x and q_i^y in the generalization tree of attribute i. The distance between q_i^x and q_i^y can be defined as the length of the path (number of edges) between two domain nodes with values q_i^x and q_i^y.

After the dataset is distributed, the new tuple is added into the cloud system. If the whole dataset in all nodes is re-generalized, this process is inefficient. The technique is using past generalization levels to generalize the additional tuples, and then assigning proper node to insert them.

The effect of adding tuples is that additional tuples may cause the whole dataset unsatisfying *k* condition. The technique for overcoming this issue is to consider which attribute is supposed to be generalized by comparing additional tuples with other tuples in the same node. An attribute to be generalized which increases generalization levels minimally, is selected for satisfying *k* condition.

Another effect of adding tuples is that additional tuples may lead to the whole dataset satisfying *k* condition. There is a possibility that generalization levels can be decreased and the whole dataset still satisfy *k* condition. If the case is occurred, the nodes (that new tuples are added) test to decrease generalization level one by one attribute. If all attributes cannot be decreased, the rest of nodes unnecessarily are computed. Meanwhile, if generalization level any attribute can be decreased, the rest node will test to decrease generalization level on same attribute. However, if any node unsatisfied *k* condition, the former generalization level is re-assigned to all nodes.

2.4 Summary

This chapter has shown some real world cases of the privacy problem, the privacy preservation models, the (*k*, *e*)-anonymous model and the algorithm that the thesis based on, and then some ideas and methods to solve the problem in cases of incremental data are discussed.

The basic definitions, problem definition, the observations, and algorithm that is conducted from observations, which is focused in this thesis, will be presented in next chapter.