Chapter 5

Experiment

In this chapter, the experimental results are presented for evaluating the efficiency of the proposed algorithm in Chapter 4. First, the experiment preparation process is presented. Next, the experiment results are presented, they are categorized to 4 parts to evaluate an effect of value of *k*, an effect of value of *e*, an effect of the number of partitions, and an effect of cardinality of an incremental dataset ($|\Delta D|$)

5.1 Experiment Preparation

The experiments are conducted using the Adult dataset from the UCI repository [29], which has been used to evaluate the (*k*, *e*)-Anonymous approach [10] and an incremental breach prevention algorithm in [11]. The dataset contains 14 attributes over 48,000 tuples. The "capital-loss" attribute is selected to be the sensitive values. Eight of the attributes are selected to be the quasi-identifier, as in [10]. Thus, the remaining number of tuples in the dataset is 1427 tuples. The range of the "capital-loss" values is 155 to 3900, with 89 distinct values. For 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. For the incremental part of a dataset, it will be divided further into ten equal parts, for appending to the dataset. Once all of the parts of the data are appended to the dataset, the execution time is subsequently reported. The efficiency is evaluated in terms of the execution time when the four parameters are changed, i.e. the *k* value, the *e* value, the size of the incremental dataset $|\Delta D|$, and the number of partitions of the static dataset. Additionally, the percentage of discarded rounds for the incremental privacy breach determination is reported to present its relationship with the efficiency. A high percentage reflects the high efficiency that the algorithm can obtain. The resulting numbers that are reported here are five times the average.

We compare our proposed algorithm to a naive re-applying algorithm that re-applies the existing static algorithm proposed in [10], to further prevent an incremental privacy breach. To recall, the idea of the algorithm is as follows. First, for each

considered partition of the dataset to be released D_n, the (k, e)-Anonymous condition is evaluated. If it satisfies the condition, then the partition is considered for any incremental privacy breach occurs against all of the released datasets. If such a partition has an incremental privacy breach with at least one of the released datasets, then the current partition can be discarded. For the incremental privacy breach determination, the algorithm must compute the difference and the intersection results between the current considered partition and each partition for each released dataset. For simplicity, suppose that the number of tuples in each released dataset is n, and the number of released datasets is m. The complexity of this computing as well as the whole algorithm is $O(mn^3)$.

The comparing characteristics of our proposed algorithm and the comparing algorithm in [10] are shown in Table 5.1.

Table 5.1: The characteristics of proposed algorithm and naïve re-applying algorithm

	Proposed algorithm	Naïve re-applying algorithm
Complexity	$O(n^3)$	$O(mn^3)$
Number of released datasets to be input	1	m
(k, e)-Anonymous protection	Satisfied k and e conditions	Satisfied k and e conditions
Incremental privacy breach protection	An output does not have an incremental privacy breach with all previous released dataset	An output does not have an incremental privacy breach with all previous released dataset

5.2 Effects of the value of *k*

In the first experiment, we evaluate the efficiency of the proposed algorithm when the value of k is varied. Such a value is varied from 3 to 15, to evaluate its effect. As in previous work, we set the value of e to be 20 and the incremental part to be 10 percent of the size of the static dataset.

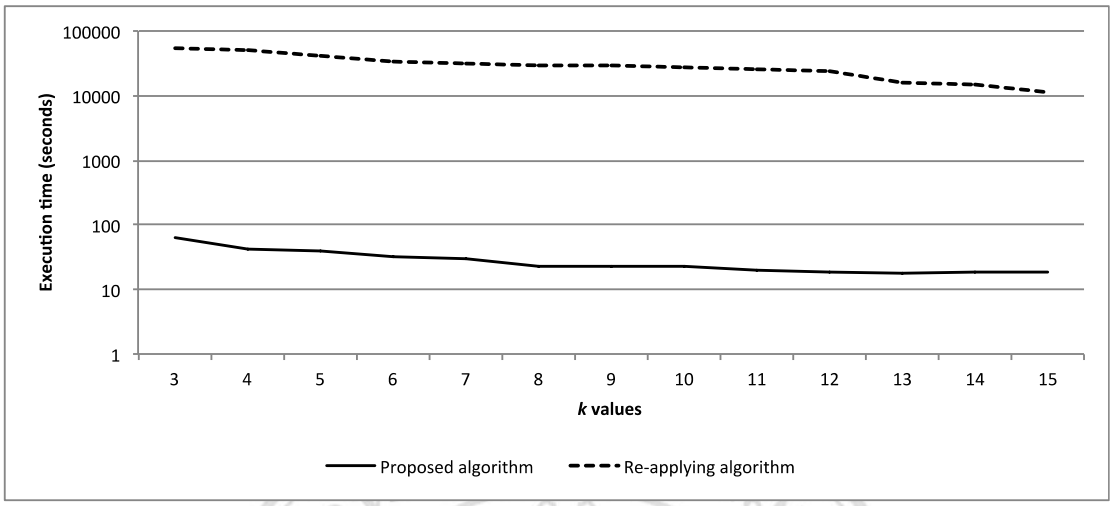

Figure 5.1: Execution time of the proposed algorithm and the naïve re-applying when the *k* value is varied

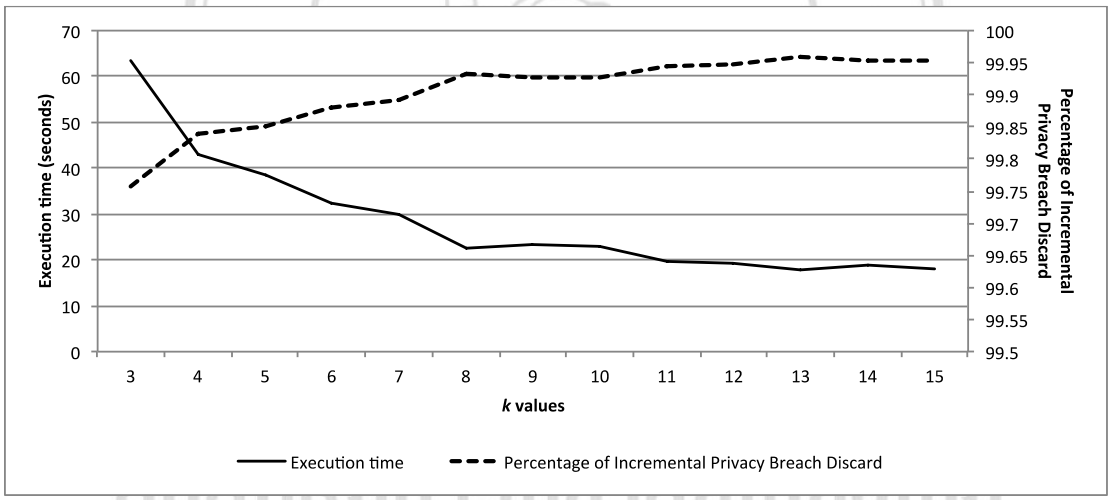

Figure 5.2: Proposed algorithm's execution time and the percentage of discard when the value of *k* is varied

The execution time of the proposed algorithm and the naive re-applying algorithm is presented in the Figure 5.1. Note that the result is shown on the logarithmic scale. The average execution time of the naive re-applying algorithm is 29,715 seconds. At the same time, the execution time of the proposed algorithm is in the range of 17 - 64 seconds. Clearly, the proposed algorithm is highly efficient because of the theoretical studies.

In the Figure 5.2, the experimental result of the proposed algorithm is presented in detail. It can be seen that the execution time of the proposed algorithm decreases when the values of *k* is increased. The reason behind the decrease is that all partitions in the previous dataset depend on the *k* value. The range of some partitions in the previous partitioned dataset can be expanded when the value of *k* is increased. Therefore, the proposed algorithm can discard an incremental privacy breach determination increasingly well.

5.3 Effects of the value of *e*

In this experiment, we evaluate the efficiency of the proposed algorithm when the value of *e* is varied. Such an *e* value is varied from 20 to 500, to evaluate its effect. We set the *k* value at 2, and the size of the incremental part is set at 10 percent of the size of the static dataset.

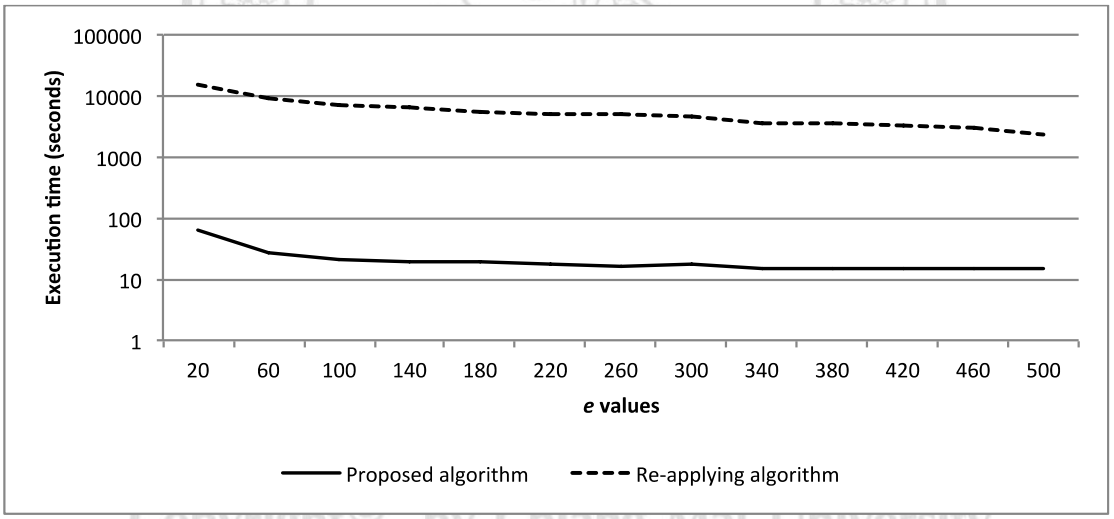

Figure 5.3: Execution time of the proposed algorithm and the naive re-applying algorithm when the *e* value is varied

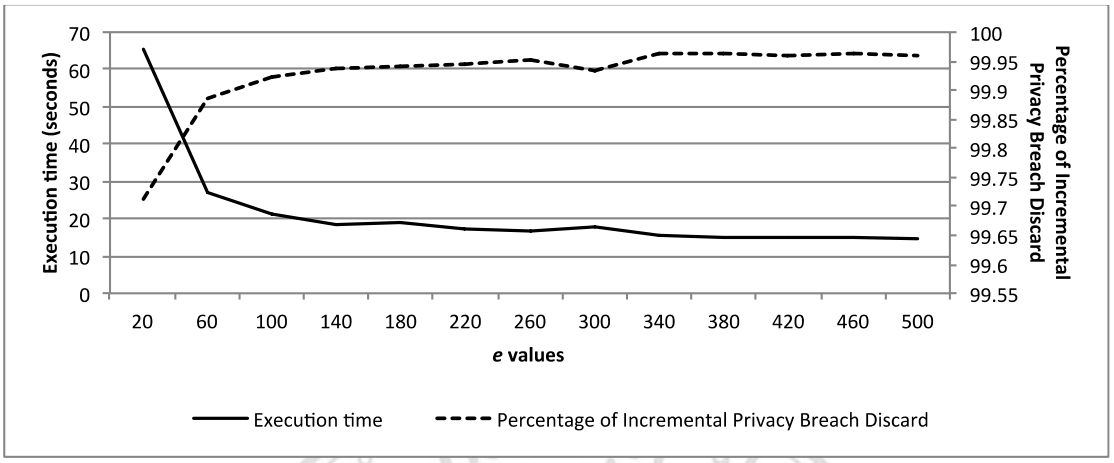

Figure 5.4: Proposed algorithm's execution time and percentage of discard when the *e* value is varied

The execution time of the proposed algorithm and the naive algorithm is presented in the Figure 5.3. The execution time of the naive algorithm is 5374 seconds. At the same time, the execution time of the proposed algorithm is in the range of 14 - 65 seconds. The result is similar to the previous experiment, i.e., the execution time of the proposed algorithm is much less than that of the naive algorithm.

The detailed experiment results when the *e* value is varied are presented in the Figure 5.4. Obviously, when the *e* value is increased, the execution time of the proposed algorithm decreases. The reason behind this decrease is that all of the partitions in the previous dataset depend on the value of *e*. The range of some of the partitions in the previous partitioned dataset can be expanded when the value of *e* is increased. Thus, the proposed algorithm can discard an incremental privacy breach determination better when the value of *e* is increased.

5.4 Effects of the Number of Partitions

In this experiment, we evaluate the efficiency of the proposed algorithm when the number of partitions in the static dataset is varied to see its effect. The variation is set at 3 to 22. We set the *k* value at 2, and the *e* value is changed to relate the numbers of the partitions.

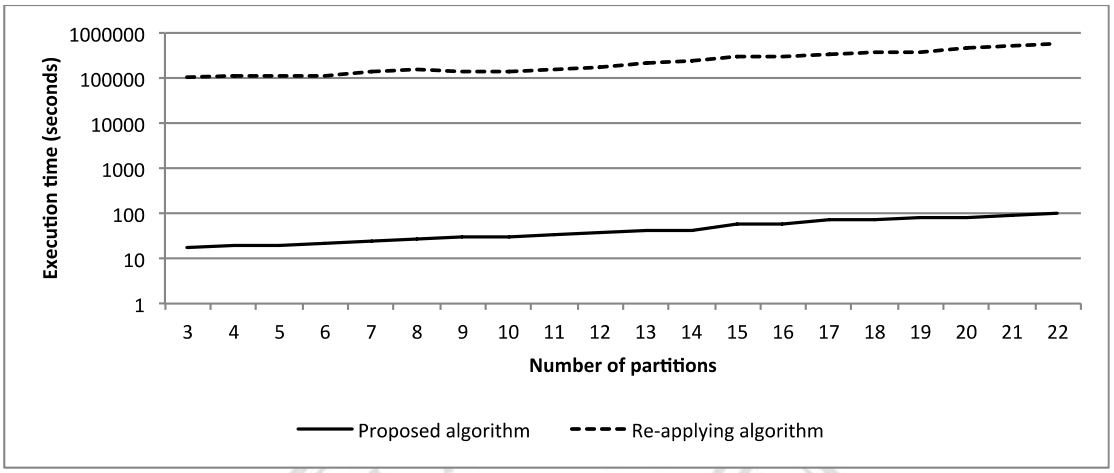

Figure 5.5: Execution time of the proposed algorithm and the naive re-applying algorithm when the number of partitions is varied

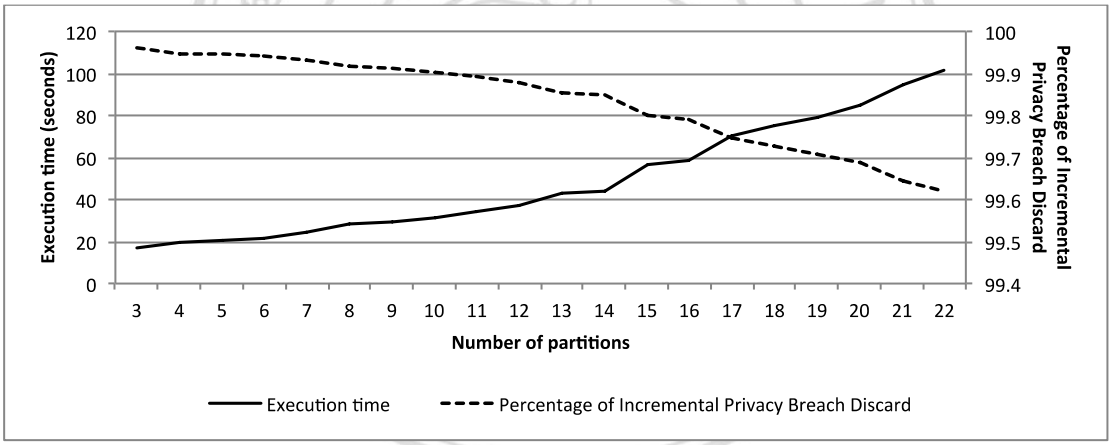

Figure 5.6: Proposed algorithm's execution time and the percentage of discard when the number of partitions is varied

From the results in the Figure 5.5 and the Figure 5.6, we can see that when the number of the partitions in the static part of the dataset is increased, the execution time of the proposed algorithm is also increased. The reason behind the increase is that the number of partitions is in reverse proportion to the parameters *k* and *e*. When the *k* and *e* values are decreased, the number of partitions is increased. Additionally, the ranges of some of the partitions in the previous partitioned dataset are reduced when the *k* and *e* values are decreased. Thus, the proposed algorithm can discard an incremental privacy

breach determination less when the number of the partitions is increased. However, compared with the naive algorithm, the proposed algorithm is still clearly efficient.

5.5 Effects of |Δ*D*|

In the last experiment, we evaluate the efficiency of the proposed algorithm when the size of the incremental dataset (|ΔD|) is varied. The variation in the set at the percentage of the whole data to be appended is 6, 8, 10, 12, and 14 percent of the size of the static dataset. We set the *k* value and the *e* value at 5 and 100, respectively.

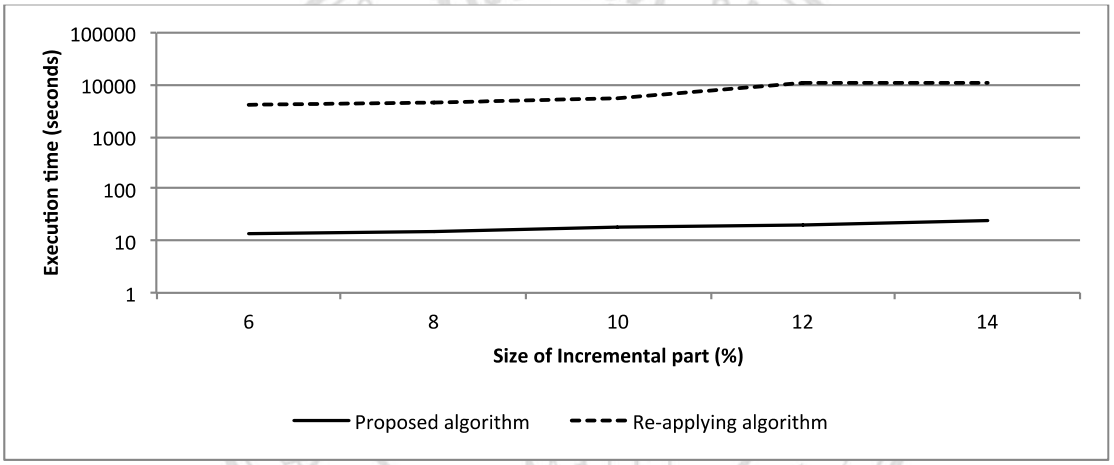

Figure 5.7: Execution time of the proposed algorithm and the naive re-applying algorithm when the size of the incremental part (%) value is varied

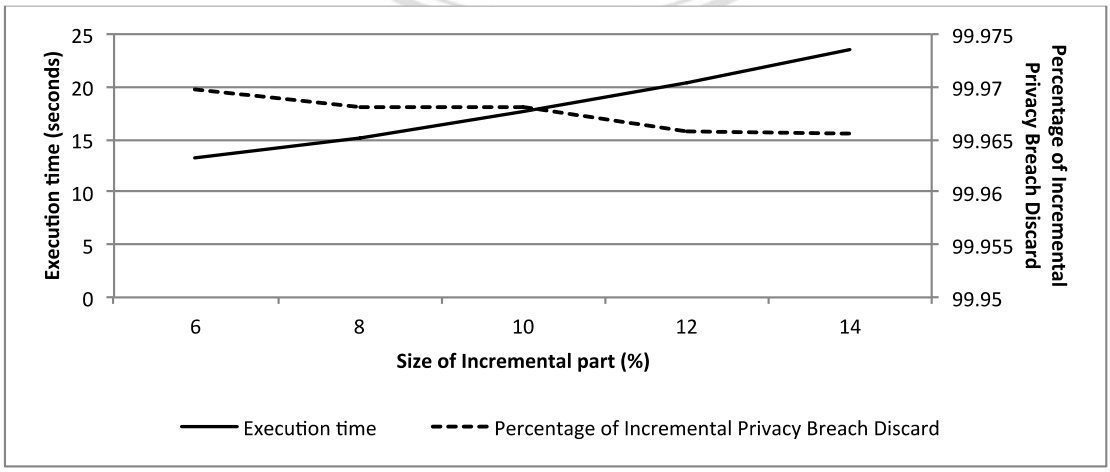

Figure 5.8: Proposed algorithm's execution time and the percentage of discard when the size of the incremental part (%) value is varied

The execution time of the proposed algorithm and the naive algorithm is presented in the Figure 5.7. The execution time of the proposed algorithm is in the range of 13 - 23 seconds. At the same time, the average execution time of the naive re-applying algorithm is 7190 seconds.

The Figure 5.8 shows the detailed execution time of the proposed algorithm in addition to the percentage of the incremental privacy breach determination discard. Obviously, when the size of the incremental part is increased, the execution time of the proposed algorithm increases. Additionally, the percentage of the discard is also decreased. This finding means that the proposed idea is less efficient when the size of the incremental part increases. However, considering that optimal solutions without any privacy breach can still be always guaranteed, and the high percentage of the discard of the privacy breach determination. It can still be claimed that the proposed algorithm is highly efficient and effective.

5.6 Summary

The efficiency of the proposed algorithm was presented in this chapter by the experiments that conducted to illustrate the real world situation. In all experiments, the proposed algorithm has more efficient than the naïve approach algorithm. The proposed algorithm can discard incremental privacy breach determination more than 99 percentages in the all of experimental cases. In the experiments of *k* and *e* values, the proposed algorithm can discard an incremental privacy breach determination better when the value of *k* and *e* are increased. In the experiments of the number of partition and the size of the incremental part, the proposed algorithm can discard an incremental privacy breach determination better when the value of the number of partition and the size of the incremental part are decreased.