Chapter 4

Observations and Proposed Algorithm

This chapter will present the observations of an incremental privacy breach scenarios. We try to add all possible new datasets and observe the impact of adding. Then, the observations are evaluated from the impact. All observations indicate to the theorem that uses to conduct the propose algorithm for finding an optimal result of an incremental privacy breach scenarios.

4.1 Observation of Incremental Privacy Breach Scenarios

In this section, an observation on a privacy breach from $P'[D_n]\rightarrow_{(k,e)}P'[D_{n-1}]$ where $n > 0$ is presented. These observations will help to develop an algorithm in the next section. Obviously, a naive approach to address the proposed problem is to verify a new version of the dataset against previous datasets to determine whether it leads to an incremental privacy breach, as discussed in the previous chapter. Evidently, such an approach can be costly as a result, the observations aim at reducing the computational cost while the result must be the same as in the naive approach.

Before we provide further discussion, we note here that the result of the privacy preservation based on the (k, e)-Anonymous in [10] is optimal. In other words, the result partitions always generate minimum sum of errors. Obviously, such a partition cannot be split further; otherwise, the result cannot satisfy the k and e conditions. Additionally, this approach does not lead to any incremental privacy breach because it does not have any previous released dataset.

Thus, we present an observation for the incremental privacy breaches from $P'[D_1]$ to $P'[D_0]$ as follows.

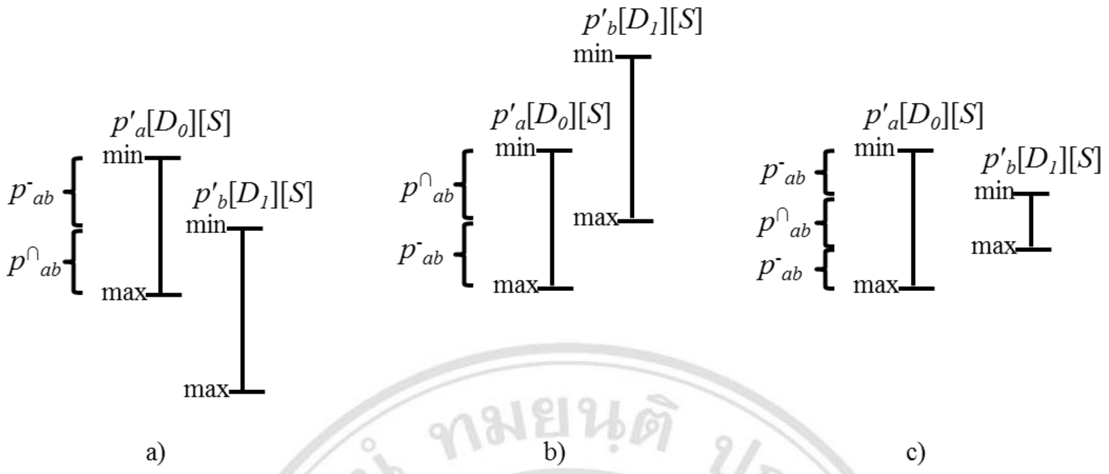


Figure 4.1: Overlapping between p_a and p_b

Lemma 1. An incremental privacy breach occurs from $P'[D_1]$ to $P'[D_0]$, or $P'[D_1] \rightarrow_{(k,e)} P'[D_0]$, if and only if any partition $p'_b[D_1]$ exists in $P'[D_1]$ and any partition $p'_a[D_0]$ exists in $P'[D_0]$ for which $p'_b[D_1][S]$ does not fully cover $p'_a[D_0][S]$, or $p'_b[D_1][S]$ overlaps with $p'_a[D_0][S]$.

Note that $p'_b[D_1]$ fully covers $p'_a[D_0]$ covers if $min(p'_b[D_1][S]) \leq min(p'_a[D_0][S]) \wedge max(p'_b[D_1][S]) \geq max(p'_a[D_0][S])$.

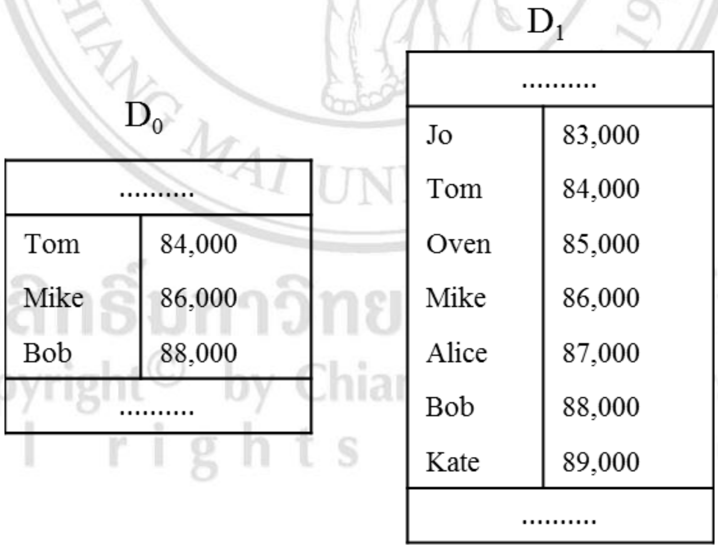
***Proof* 1**. Let $p^-_{ab} = p'_a[D_0] - p'_b[D_1]$, and $p^\cap_{ab} = p'_a[D_0] \cap p'_b[D_1]$. Suppose that the top part of $p'_b[D_1][S]$ is covered by some part of $p'_a[D_0][S]$, or $min(p'_b[D_1][S]) > min(p'_a[D_0][S]) \wedge min(p'_b[D_1][S]) < max(p'_a[D_0][S]) \wedge max(p'_b[D_1][S]) \geq max(p'_a[D_0][S])$ as shown in Figure 4.1 a). Then p^-_{ab} contains some tuples from $p'_a[D_0]$ such that the sensitive value of the tuples are less than $min(p'_b[D_1][S])$. The sensitive value of the covered tuples are at least $min(p'_b[D_1][S])$, and they can be obtained in $p^\cap_{ab}$. Thus, $P'[D_0] = p^-_{ab} \cup p^\cap_{ab}$ and $p^-_{ab} \cap p^\cap_{ab} = \emptyset$. Consider that there is a separation of $p'_a[D_0]$ into p^-_{ab} and $p^\cap_{ab}$ in which the incremental privacy breach has occurred because the original partition $p'_a[D_0]$ is a part of the optimal solution. Thus, the separated p^-_{ab} and $p^\cap_{ab}$ cannot satisfy the (k, e)-Anonymous condition. Therefore, $P'[D_1] \rightarrow_{(k,e)} P'[D_0]$.

In the case where the bottom of $p'_b[D_1][S]$ is covered by some part of $p'_a[D_0][S]$, or $max(p'_b[D_1][S]) < max(p'_a[D_0][S]) \wedge max(p'_b[D_1][S]) > min(p'_a[D_0][S]) \wedge$

min($p'_b[D_1][S]$) ≤ *min*($p'_a[D_0][S]$), it is obvious that the incremental privacy breach has occurred in the same way.

Last, in the case where the bottom of $p'_b[D_1][S]$ shrinks from $p'_a[D_0][S]$, or *max*($p'_b[D_1][S]$) < *max*($p'_a[D_0][S]$) ∧ *min*($p'_b[D_1][S]$) > *min*($p'_a[D_0][S]$), an incremental privacy breach has also occurred.

To illustrate the observation, let us consider Figure 4.2. This figure shows a part of D_0 and a part of D_1, and it can be seen that the part of D_0 has increased by some tuples to become the part of D_1. In Figure 4.3, the partitions p_a and p_b before shuffling are shown for the readability. Let k = 3 and e = 2000, then, it can be seen that the top part of $p_b[D_1][S]$ is covered by some part of $p_a[D_0][S]$. The result of the difference between $p_b[D_1][S]$ and $p_a[D_0][S]$, p^{-}_{ab}, is the partition containing {Tom}. Additionally, the result of the intersection between $p_b[D_1][S]$ and $p_a[D_0][S]$, $p^{\cap}_{ab}$, is the partition containing {Mike, Bob}. Both partitions cannot satisfy the (*3, 2000*)-Anonymous condition.



D₀

....	
Tom	84,000
Mike	86,000
Bob	88,000
....	

D₁

....	
Jo	83,000
Tom	84,000
Oven	85,000
Mike	86,000
Alice	87,000
Bob	88,000
Kate	89,000
....	

Figure 4.2: An example of two released datasets D_0 and D_1

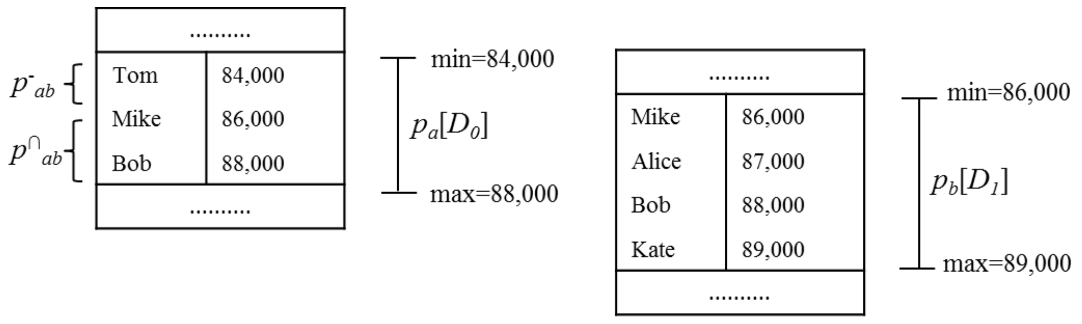


Figure 4.3: An example situation in which the top part of $p_b[D_1][S]$ is covered by some part of $p_a[D_0][S]$

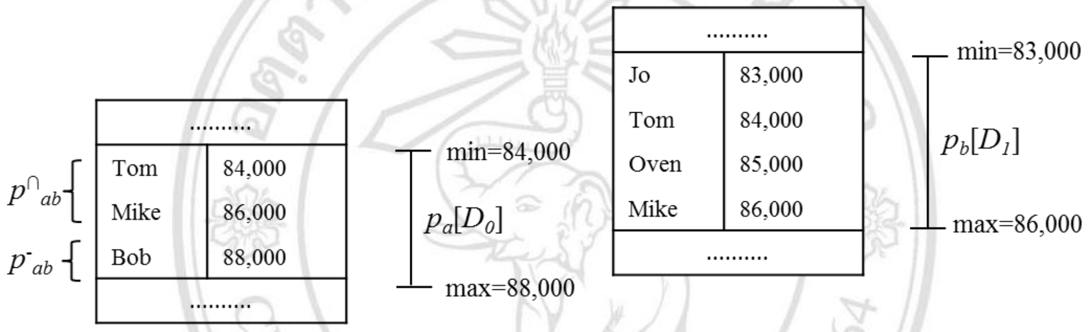


Figure 4.4: An example situation in which the bottom part of $p_b[D_1][S]$ is covered by some part of $p_a[D_0][S]$

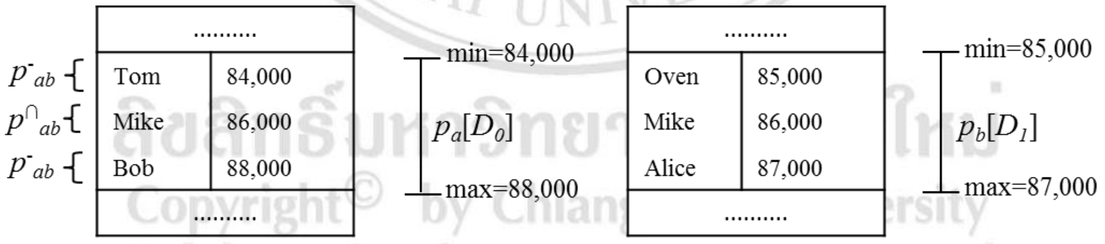


Figure 4.5: An example situation in which the $p_b[D_1][S]$ shrinks from $p_a[D_0][S]$

Another example is shown in Figure 4.4. Let k = 3 and e = 2000; then, it can be seen that the bottom part of $p_b[D_1][S]$ is covered by some part of $p_a[D_0][S]$. The result of the difference between $p_b[D_1][S]$ and $p_a[D_0][S]$, p^-_{ab}, is the partition that contains {Tom, Mike} and the result of the intersection between $p_b[D_1][S]$ and $p_a[D_0][S]$, $p^\cap_{ab}$, is the partition that contains {Bob}. It is not possible for both partitions to satisfy the (*3, 2000*)-Anonymous condition.

In the last case, from Figure 4.5, let $k = 3$ and $e = 2000$; then, it can be seen that the $p_b[D_1][S]$ shrinks from $p_a[D_0][S]$. The result of the difference between $p_b[D_1][S]$ and $p_a[D_0][S]$, p^{-}_{ab}, is the partition that contains {Tom, Bob} and the result of the intersection between $p_b[D_1][S]$ and $p_a[D_0][S]$, $p^{\cap}_{ab}$, is the partition that contains {Mike}. It is not possible for both partitions to satisfy the (*3, 2000*)-Anonymous condition.

Next, an observation on the incremental privacy breach from $P'[D_1] \rightarrow_{(k,e)} P'[D_0]$ in Lemma 1 is presented. Then, another observation for $P'[D_2]$ which is the foundation for $P'[D_n]$ subsequently will be presented.

Lemma 2. A partition $p'_c[D_2]$ does not lead to an incremental privacy breach from $P'[D_2] \rightarrow_{(k,e)} P'[D_0]$, if and only if $p'_c[D_2][S]$ fully covers $p'_b[D_1][S]$, $p'_b[D_1][S]$ fully covers $p'_a[D_0][S]$, and $p'_b[D_1]$ does not lead to an incremental privacy breach from $P'[D_1] \rightarrow_{(k,e)} P'[D_0]$.

***Proof* 2**. Suppose that $p'_c[D_2][S]$ fully covers $p'_b[D_1][S]$ and $p'_b[D_1]$ fully covers $p'_a[D_0]$, as shown in Figure 4.6, and that $p'_b[D_1]$ does not lead to an incremental privacy breach with $p'_a[D_0]$. First, consider the intersection and the difference between $p'_a[D_0]$ and $p'_b[D_1]$, it can be seen that $p'_a[D_0] = p^{\cap}_{ab}$, and $p'_b[D_1] = p^{\cap}_{ab} \cup p^{-}_{ba}$. Additionally, $p'_b[D_1] = p'_a \cup p^{-}_{ba}$. From the fact that $p'_b[D_1]$ does not lead to an incremental privacy breach, while considering $p'_a[D_0]$, p^{-}_{ba} satisfies the (k, e)-Anonymous condition.

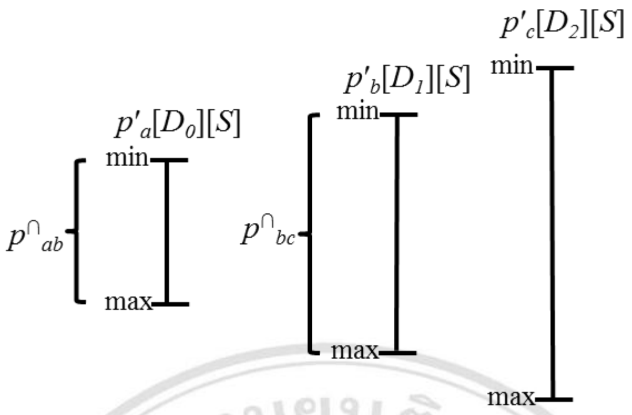


Figure 4.6: Fully covered situation

Next, consider the intersection and the difference between $p'_b[D_1]$ and $p'_c[D_2]$, it can be seen that $p'_b[D_1] = p^{\cap}_{bc}$, $p'_c[D_2] = p^{\cap}_{bc} \cup p^{-}_{cb}$. Additionally, $p'_c[D_2] = p'_b[D_1] \cup p^{-}_{cb}$. From $p'_b[D_1] = p'_a \cup p^{-}_{ba}$, therefore, $p'_c[D_2] = p'_a \cup p^{-}_{ba} \cup p^{-}_{cb}$.

Finally, consider the intersection and the difference between $p'_a[D_0]$ and $p'_c[D_2]$, it can be seen that $p'_a[D_0] = p^{\cap}_{ac}$, $p'_c[D_2] = p^{\cap}_{ac} \cup p^{-}_{ca}$. Additionally, $p'_c[D_2] = p'_a[D_0] \cup p^{-}_{cb}$. From $p'_c[D_2] = p'_a \cup p^{-}_{ba} \cup p^{-}_{cb}$, therefore, $p^{-}_{cb} = p^{-}_{ba} \cup p^{-}_{cb}$.

However, p^{-}_{ba} has already satisfied the (k, e)-Anonymous condition, therefore, $p^{-}_{ba} \cup p^{-}_{cb}$ or p^{-}_{cb} satisfies the (k, e)-Anonymous condition, also. Thus, the separation of $p'_c[D_2]$ into $p^{\cap}_{ac}$ and p^{-}_{cb} does not lead to an incremental privacy breach from $P'[D_2] \rightarrow_{(k,e)} P'[D_0]$.

In Figure 4.7, let $k = 3$ and $e = 2000$, it can be seen that $p_c[D_2][S]$ fully covers $p_b[D_1][S]$, $p_b[D_1]$ fully covers $p_a[D_0]$, and $p_b[D_1]$ does not lead to an incremental privacy breach with $p_a[D_0]$. The result of the difference between $p_c[D_2][S]$ and $p_a[D_0][S]$, p^{-}_{ab}, is an empty partition, and the result of the intersection between $p_c[D_2][S]$ and $p_a[D_0][S]$, $p^{\cap}_{ab}$, is the partition that contains {Tom, Mike, Bob}. This partition satisfies the (3, 2000)-Anonymous condition.

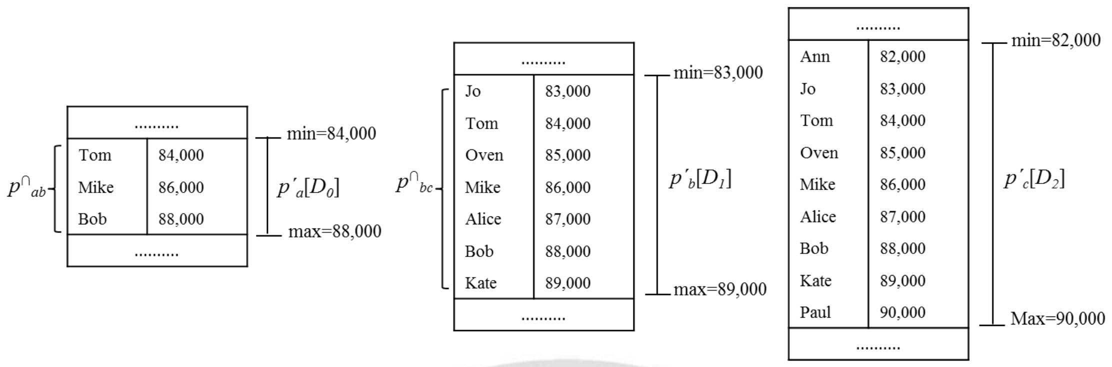


Figure 4.7: An example situation when the $p_b[D_1][S]$ is fully covered by $p_a[D_0][S]$

It can be seen in the Lemma 1 and Lemma 2, when the dataset D_n is partitioned, an incremental privacy breach can be discarded, from considering any partition that does not fully covers any partition in $P'[D_{n-1}]$. The cost of the intersection and the difference is more expensive than the cost of comparing the range between the partitions. Furthermore, if the partition fully covers any partition in $P'[D_{n-1}]$ and the partition does not lead to an incremental privacy breach from $P'[D_n]$ to $P'[D_{n-1}]$, then, there is no need to determine the incremental privacy breach for such a partition with the dataset $P'[D_{n-2}]$, $P'[D_{n-3}]$, ..., $P'[D_0]$. Subsequently, the execution time can be reduced.

Afterwards, the optimal solution $P'[D_n]$ is obtained. In some situations, there might exist a partition $p'_c[D_n]$ that fully covers a partition in the previous released dataset D_{n-1}, $p'_b[D_{n-1}]$ and such $p'_c[D_n]$ can be separated into many parts without leading to an incremental privacy breach from $P'[D_n]\rightarrow_{(k,e)}P'[D_{n-1}]$. However, the incremental privacy breach can still occur when considering the other previous dataset.

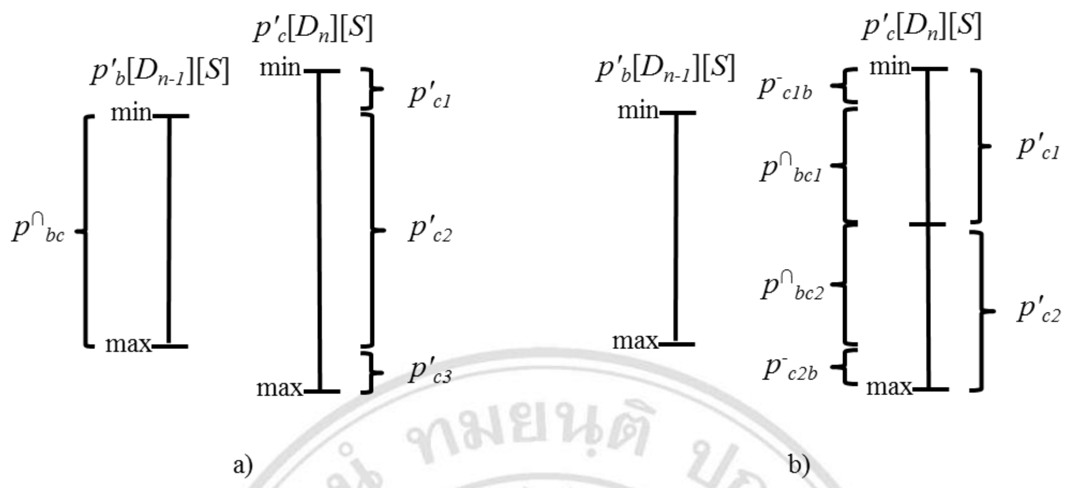


Figure 4.8: An example situation in which the $p_c[D_n][S]$ is separated into two parts

Lemma 3. The separation of any partition $p'_c[D_n]$ in $P'[D_n]$ leads to an incremental privacy breach from $P'[D_n]$ to at least one of previous partitioned dataset, if and only if the partition $p'_c[D_n]$ fully covers any partition $p'_b[D_{n-1}]$ in $P'[D_{n-1}]$, and $P'[D_n]$ is optimal.

***Proof* 3**. The separation of any partition $p'_c[D_n]$ in $P'[D_n]$ can be grouped into two situations, i.e. the partition $p'_c[D_n][S]$ fully covers any partition $p'_b[D_{n-1}][S]$ in $P'[D_{n-1}]$, and the partition $p'_c[D_n][S]$ is equal to any partition $p'_b[D_{n-1}][S]$ in $P'[D_{n-1}]$.

In the first situation, suppose that $p'_c[D_n][S]$ fully covers $p'_b[D_{n-1}][S]$, as shown in Figure 4.8 a), then, a partition $p'_c[D_n]$ can be considered in 3 parts, i.e., p'_{c1} , p'_{c2} , and p'_{c3} . Let p'_{c2} be equal to $p'_b[D_{n-1}]$, additionally, the partition p'_{c2} is equal to $p^{\cap}_{bc}$. Consider p'_{c1} and p'_{c3}, if both of them satisfy the (k, e)-Anonymous condition, then they can be separated without leading to an incremental privacy breach from $P'[D_n]\rightarrow_{(k,e)}P'[D_{n-1}]$. However, if $P'[D_n]$ is optimal, then at least one of them does not satisfy the (k, e)-Anonymous condition.

Suppose that $p'_c[D_n]$ is separated into two parts, p'_{c1} and p'_{c2}, as shown in Figure 4.8 b). Consider p'_{c1} and p'_{c2}, which differed by $p'_b[D_{n-1}]$, such results are p^{-}_{c1b} and p^{-}_{c2b}. From previous discussion, p^{-}_{c1b} and p^{-}_{c1b} are equal to p'_{c1} and p'_{c3} in Figure 4.8 a) respectively. Additionally, one of them does not satisfy the (k, e)-Anonymous condition, and the separation of $p'_c[D_n]$ leads to an incremental privacy breach from $P'[D_n]\rightarrow_{(k,e)}$ $P'[D_{n-1}]$.

In the second situation, suppose that $p'_c[D_n][S]$ is equal to $p'_b[D_{n-1}][S]$, as shown in Figure 4.9 a). Let $p'_c[D_n]$[S] be separated into two parts p'_{c1} and p'_{c2} where both satisfy the (k, e)-Anonymous condition. Therefore, the separation of $p'_c[D_n][S]$ does not lead to an incremental privacy breach from $P'[D_n]\rightarrow_{(k,e)}P'[D_{n-1}]$. Because the result of intersection of $p^{\cap}_{bc1}$ and $p^{\cap}_{bc2}$ are equal to p'_{c1} and p'_{c2} respectively, also, $p^{\cap}_{bc1}$ and $p^{\cap}_{bc2}$ do not lead to an incremental privacy breach from $P'[D_n]\rightarrow_{(k,e)}P'[D_{n-1}]$. However, $p'_b[D_{n-1}][S]$ may fully covers to previous partition in $P'[D_{n-2}]$, hence, separation of $p'_b[D_{n-1}][S]$ will lead to an incremental privacy breach from $P'[D_{n-1}]\rightarrow_{(k,e)}P'[D_{n-2}]$, that is similar to the first situation. The partition $p'_c[D_n][S]$ is equal to $p'_b[D_{n-1}][S]$, additionally, the separation of $p'_c[D_n][S]$ will lead to an incremental privacy breach from $P'[D_n]\rightarrow_{(k,e)}P'[D_{n-2}]$.

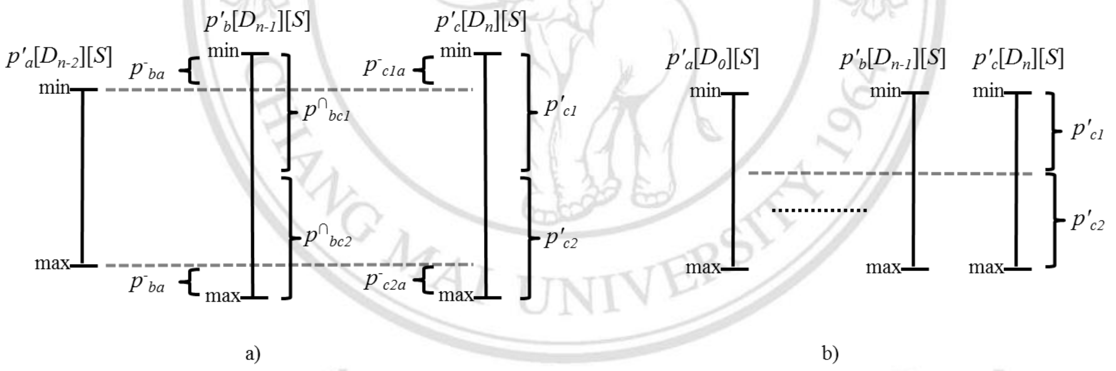


Figure 4.9: The examples of 2 situations on data scenarios when the $p_c[D_n]$[S] is separated into two parts

Suppose that all of partitions are equal as shown in Figure 4.9 b), $p'_a[D_0][S]$ to $p'_c[D_n][S]$. The separation of $p'_c[D_n]$ will lead to an incremental privacy breach from $P'[D_n]\rightarrow_{(k,e)}P'[D_0]$, as well as Lemma 1.

From the first and the second situations, the separation of partition $p'_c[D_n]$ leads to an incremental privacy breach from $P'[D_n]$ to at least one of the previously partitioned dataset, if $p'_c[D_n]$ fully covers $p'_b[D_{n-1}]$, and $P'[D_n]$ is optimal.

Obviously, from Lemma 1, Lemma 2 and Lemma 3, we can generalize to the cases in which $n > 2$ as in the following theorem. This theorem will be applied to the proposed algorithm in the next section, to further improve its efficiency.

Theorem 1. $p'_i[D_n]$ does not lead to an incremental privacy breach from $P'[D_n] \rightarrow_{(k,e)} P'[D_{n-2}]$, $P'[D_n] \rightarrow_{(k,e)} P'[D_{n-3}]$, . . ., $P'[D_n] \rightarrow_{(k,e)} P'[D_0]$, if $p'_i[D_n]$ fully covers to $p'_j[D_{n-1}]$ and $p'_j[D_{n-1}]$ does not lead to an incremental privacy breach from $P'[D_{n-1}] \rightarrow_{(k,e)} P'[D_{n-2}]$, $P'[D_{n-1}] \rightarrow_{(k,e)} P'[D_{n-3}]$, . . ., $P'[D_{n-1}] \rightarrow_{(k,e)} P'[D_0]$.

4.2 Proposed Algorithm

From the proposed theorem in the previous section, we can propose an algorithm that preserves the privacy in the incremental data scenarios as shown in Figure 4.10. This algorithm is based on the existing non-incremental privacy preservation proposed in [10]. However, not all previously released datasets are used as the input, only the previous version of the dataset. From Theorem 1, it can be seen that the current partition that is consideration, which is specified by indexes i and j, must fully cover a partition in the previous dataset $P'[D_{n-1}]$, or the current partition cannot overlap with all of the partitions in $P'[D_{n-1}]$. Thus, before an incremental privacy breach is to be determined for the current partition, an overlapping condition for such a partition can be considered against all of the partitions in $P'[D_{n-1}]$. If any overlap is found with any partition in $P'[D_{n-1}]$, then the current partition can be discarded. The overlap computing considers d_i and d_j (the border of the current partition) with the first tuple and the last tuple of each partition in $P'[D_{n-1}]$. Thus, the complexity of the overlap consideration is $O(p)$, where p is the number of partitions in $P'[D_{n-1}]$.

Input:

P' $[D_{n-1}]$: a partitioned dataset at time t_{n-1}

D_n: a dataset that t_n, sorted by sensitive values

k: a threshold for the minimum number of distinct values

e: a minimum *error* of the threshold values

Output:

$P'[D_n]$: the portioned dataset, that does not have an incremental privacy breach with $P'[D_0],\ldots, P'[D_{n-1}]$, and it has a minimum sum of *error*

partition: the partition information

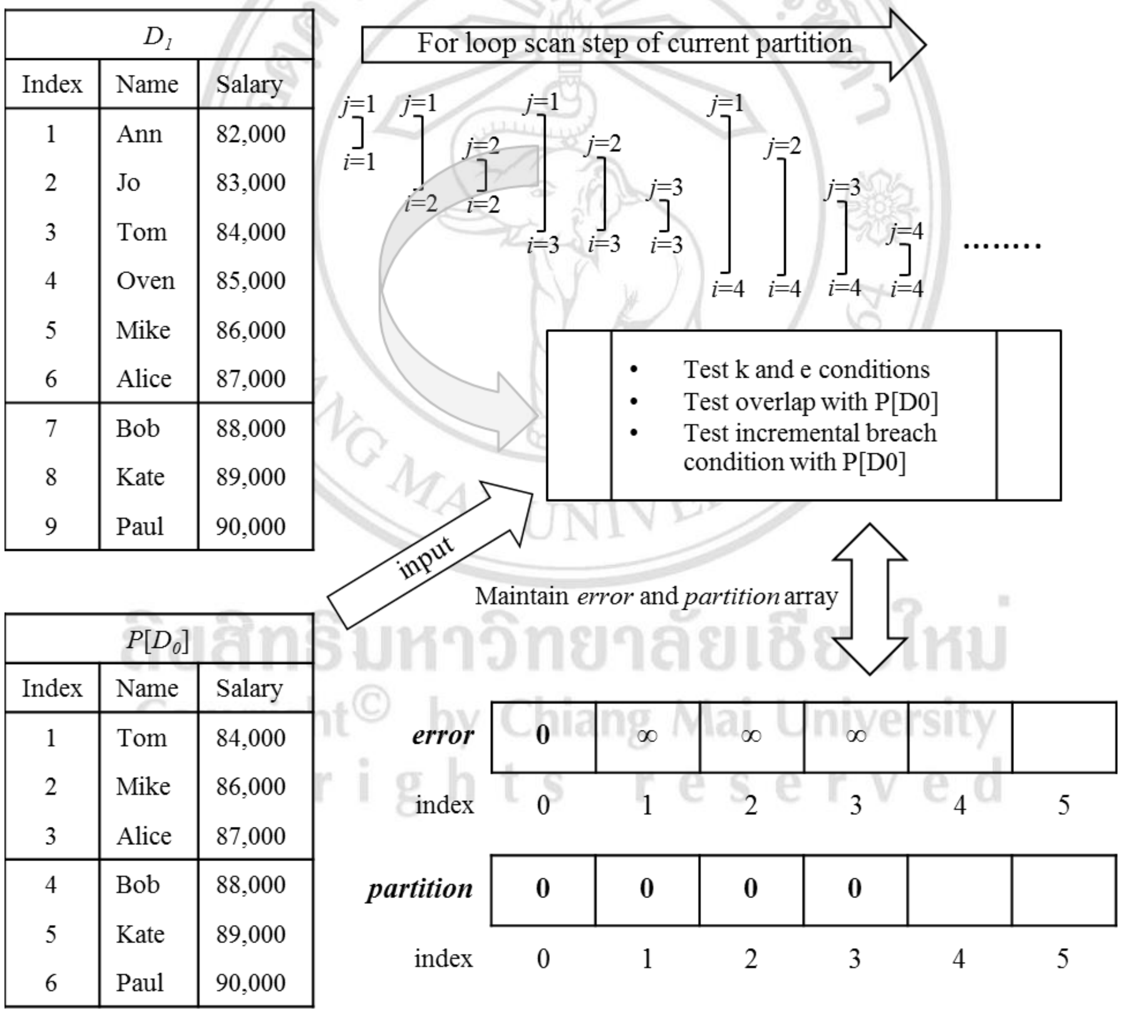
error: the *error* information

Method:

```
 error[0] = 0
partition[0] = 0
for $I$ =1 to $D_n$.size
   error[$i$] = infinity
   partition[$i$] = partition[$I - 1$]
   for $j$ = 1 to $i$
     if {$d_i[D_n][S],\ldots, d_j[D_n][S]$} satisfy $k$ and $e$
       if {$d_i[D_n][S],\ldots, d_j[D_n][S]$} not overlaps with all of partitions in $P'[D_{n-1}]$
         if {$d_i[D_n][S],\ldots, d_j[D_n][S]$} doesn't has an incremental breach with $P'[D_{n-1}]$
           current_error = $d_i[D_n][S] - d_j[D_n][S]$
         else
           current_error = infinity
         end if
       else
         current_error = infinity
       end if
     else
       current_error = infinity
     end if
     temp = error[$j - 1$] + current_error
     if temp < error[$i$]
       error[$i$] = temp
       partition[$i$] = $j$
     end if
   end for
end for
```

Figure 4.10: The Proposed Algorithm

Obviously, if the current partition does not lead to an incremental privacy breach that is subjected to $P'[D_{n-1}]$, then the current partition does not lead to an incremental privacy breach with all of the previous released datasets as well. The determination of the incremental privacy breach searches only in $P'[D_{n-1}]$. Thus, the complexity of the incremental privacy breach detection becomes $O(n)$. In the worst case, the overlap determination cannot skip the incremental privacy breach determination within all of the current partitions, thus, the complexity of our proposed algorithm in the worst case becomes $O(n^3)$, and the complexity of our proposed algorithm in the best case becomes $O(pn^2)$.



For loop scan step of current partition
j=1 i=1
j=1 i=2
j=2 i=2
j=1 i=3
j=2 i=3
j=3 i=3
j=1 i=4
j=2 i=4
j=3 i=4
j=4 i=4
....
• Test k and e conditions
• Test overlap with P[D0]
• Test incremental breach condition with P[D0]
input
Maintain error and partition array

D_1		
Index	Name	Salary
1	Ann	82,000
2	Jo	83,000
3	Tom	84,000
4	Oven	85,000
5	Mike	86,000
6	Alice	87,000
7	Bob	88,000
8	Kate	89,000
9	Paul	90,000

$P[D_0]$		
Index	Name	Salary
1	Tom	84,000
2	Mike	86,000
3	Alice	87,000
4	Bob	88,000
5	Kate	89,000
6	Paul	90,000

error	0	∞	∞	∞		
index	0	1	2	3	4	5

partition	0	0	0	0		
index	0	1	2	3	4	5

Figure 4.11: Illustration of proposed algorithm

Figure 4.11 illustrates the proposed algorithm by example. Suppose that the dataset D_1 is the current dataset, the dataset $P[D_0]$ is the previously released dataset that is considered for an incremental privacy breach condition. The proposed algorithm scans all possible partitions 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 4.10. The steps of loop scanning are shown in the top of the example, Figure 4.11. Each step will consider the current partition, the current partition is evaluated by 3 conditions as show in the 7th, 8th, and 9th lines for the *k* and *e* condition, not overlapping of the current partition with all partitions in $P[D_0]$ condition, and an incremental breach with $P[D_0]$ condition respectively. If the current partition does not pass at least one of the conditions, a variable *current_error* is set to be infinity by the statement in the 12th, 15th, and 18th lines. In the case that the current partition pass all the conditions, the *current_error* is set to be $d_i[D_n][S] - d_j[D_n][S]$ by the statement in the 10th line.

Next, the algorithm determines the summation of error at index *i*, and such value is stored in the *temp* variable as show in the statement in the 20th line of Figure 4.10. 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 21st 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 22nd and 23rd 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 4.1: An example of the two datasets, to illustrate the algorithm

D_0			D_1			error		partition	
Index	Name	Salary	Index	Name	Salary	D_0	D_1	D_0	D_1
			1	Ann	82,000		Infinity		0
			2	Jo	83,000		Infinity		0
1	Tom	84,000	3	Tom	84,000	Infinity	Infinity	0	0
			4	Oven	85,000		Infinity		0
2	Mike	86,000	5	Mike	86,000	Infinity	Infinity	0	0
3	Alice	87,000	6	Alice	87,000	3000	5000	1	1
4	Bob	88,000	7	Bob	88,000	4000	6000	1	1
5	Kate	89,000	8	Kate	89,000	5000	7000	1	1
6	Paul	90,000	9	Paul	90,000	5000	7000	4	7

Let us consider the efficiency of our proposed algorithm by another example. Suppose that the datasets D_0 and D_1 are given as shown Figure 4.12. The algorithm begins with scanning all of the possible partitions in D_1. Let us discuss the situation when i = 5 and j = 1, evidently, the maximum value of the current partition (86,000 in D_1) is less than the maximum value of the first partition of D_0 (87,000). The current partition can be discarded for any further computation. Then, when i = 6 and j = 1, it can be seen that the current partition fully covers the first partition in D_0, thus the incremental privacy breach is to be determined. In this case, there is no the incremental privacy breach, so the algorithm continues to determine whether the current partition leads to the privacy breach when subjected to the k and e condition. On the other hand, when i = 7 and j = 3, the current partition also fully covers the first partition in D_0, however, an incremental privacy breach occurs. Thus, this consideration is discarded. In the end, the optimal solution without any privacy breach of D_1 is partitioned as show in the outline of D_1 in Figure 4.12.

In order to present the partitioning information, the *partition* array is to be scanned backward as shown in Figure 4.12. 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

9, and the value in the *partition* array of such index is 7 as shown in the right-most column in Figure 4.12. Therefore, the first partition is the set of tuples {d_9, d_8, d_7}. Then, the second partition can be determined by scanning the tuple of index number 6 (the most index number less than the previous partition, i.e. 7), 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_6, d_5, d_4, d_3, d_2, d_1}.

4.3 Summary

This chapter discusses the incremental privacy breach situations, the theorems that are conducted by consideration of the situations, and the proposed algorithm that is conducted by the theorems. The proposed algorithm considers only one version of previously released dataset and the current dataset as the inputs for finding an optimal solution dataset for releasing. The efficiency of the proposed algorithm will be presented in the next chapter.