k-Anonymity Based on Tuple Migration in Sharing Data

Anh T. Truong [1, 2]

[1] Faculty of Computer Science and Engineering, Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam
[2] Vietnam National University, Ho Chi Minh City, Linh Trung Ward, Thu Duc District, Ho Chi Minh City, Vietnam
E-mail: anhtt@hcmut.edu.vn

***Abstract*—Nowadays, the development of big data, cloud computing, and the internet of things has led to an increase in sharing data. Through the data mining process, some valuable information can be discovered from such shared data. However, most shared data contain personal sensitive information such as users' location information or disease status and attackers, by analysing such data, may also extract some private (sensitive) information of the user and this can result in threats against the user's privacy. Therefore, before sharing data or making it open we must apply privacy techniques to protect the sensitive information in the data. In this paper, we propose a new approach as well as a technique to guarantee k-anonymity, the most popular privacy protection technique, in the data. The main idea is to design an algorithm to organize tuples/records in the data into groups and then migrate tuples between the groups such that all the groups satisfy k-anonymity. Specifically, the proposed algorithm also maintains the significant association rules in the k-anonymity data so that the data mining process, based on association rule mining, can preserve valuable information as in original data. We perform experiments to evaluate the performance and data utility of our proposed technique in comparison with state-of-the-art anonymization techniques. The experimental results show that our technique outperforms such state-of-the-art ones.**

***Keywords*—k-anonymity, privacy preserving, privacy protection, sharing data, open data**

I. Introduction

Open data and sharing data have been trends for recent years [1]. Through the data mining process, some meaningful and valuable information can be discovered from such shared or open data. However, most shared data contain personal sensitive information such as users' location information or disease status and attackers, by analysing such data, may also extract some private (sensitive) information of the user and this can result in threats against the user's privacy. Therefore, some privacy techniques have been proposed to preserve the privacy of the data before sharing or opening it.

Randomization is a kind of such techniques to remove or modify personal information in the original data (i.e., the data before sharing). Assume that we have a dataset. $X = \{x_1, x_2, \ldots, x_N\}$. For every x_i, the randomization technique will add a noise based on the random distribution function $f_Y(y)$ with independent components $\{y_1, y_2, \ldots, y_N\}$. The transformed dataset $\{z_1, z_2, \ldots, z_N\} = \{x_1 + y_1, x_2 + y_2, \ldots, x_N + y_N\}$ satisfies the privacy requirement since the original sensitive information has been changed so it is impossible or difficult to recover. Besides randomization, k-anonymity [2–4] is one of the most popular techniques to preserve privacy. The main idea of k-anonymity is to transform the original dataset to a modified version in which for every record Ri, there are at least k−1 other records with the same values on *quasi attributes* (i.e., the attributes that can combine to uniquely reidentify a person). Most of the implementations of k-anonymity (i.e., k-anonymity algorithms) are based on generalization (a data value can be transformed to higher representation) or suppression (the data value is suppressed to a compact one) approaches. Incognito [5], OKA [6], KACA [7] are notable representatives in suppression approach while GCCG [8] is an algorithm based on clustering and generalizing the quasi-attribute values of the records in each cluster.

Clearly, the data mining process can help to extract some valuable information from shared or open data [9–12]. However, the results of the data mining process are only *significant* if the shared data is the same or "very close" to the original data (i.e., the valuable information in the original data is also maintained in the shared data). Unfortunately, most k-anonymity algorithms mentioned above only concentrate on general-purpose applications. Therefore, using such algorithms to anonymize the original data (to protect the privacy) and then share the anonymized data can lead to insignificant results in the data mining process later because the anonymized data may be "very different" to the original version. This demands the development of new algorithms for not only transforming datasets to achieve k-anonymity but also maintaining significant information for the data mining process.

Intuitively, generalization and suppression approaches may not be useful to preserve significant information because the information in the original data will be changed to general or compressed versions (and so the original values of the information also change).

Manuscript received September 5, 2022; revised September 28, 2022; accepted January 29, 2023; published July 11, 2023.

In Ref. [12−14], the authors proposed and demonstrated the novel idea and a general algorithm for an approach called tuple member migration. This approach is based on migrating records (or group tuple members) between groups and converting their quasi-attribute values to assure that all tuple members of a group have the same values on quasi attributes (to obtain k-anonymity). Moreover, we can improve this approach by keeping significant tuples and converting only less significant ones. By this, significant information is maintained in anonymized data and thus ready for the data mining process later.

In this paper, we follow the tuple member migration approach and propose an effective algorithm to obtain k-anonymity that outperforms state-of-the-art algorithms. Specifically, the proposed algorithm also maintains the significant association rules in the k-anonymity data so that the data mining process, based on association rule mining, can preserve valuable information as in original data. We also perform experiments to evaluate our proposed technique in comparison with state-of-the-art anonymization technique. The experimental results show that our technique outperforms such state-of-the-art ones.

The rest of the paper is organized as follows. Section II gives a brief survey of related works such as k-anonymity and the data mining. We also state the general problem that relates to privacy preserving in data sharing (anonymized and mined process) and propose the general structure for anonymized and mined technique. Section III presents the main algorithm of the technique that implements the tuple member migration idea to preserve the privacy in data. The experiment for evaluating the quality of data after the anonymization is shown in Section IV. Concluding remarks and future works are discussed in Section V.

II. K-ANONYMITY, TUPLE MEMBER MIGRATION METHOD, AND DATA MINING

The goal of k-anonymity is to transform the dataset to a new one which satisfies the requirement that for every record R_i, there are at least k−1 other records with the same values on Quasi-Identifier (QI) attributes (for short, quasi attributes). For convenience, with the purpose of more understanding about k-anonymity, we briefly provide the definitions and examples of k-anonymity in Tables I–IV as follows:

TABLE I. K-ANONYMITY, TYPES OF ATTRIBUTES IN A DATASET

Type of Attributes	Description	Example
Explicit identifier	Attributes explicitly identify people or objects.	ID, NAME
Quasi attributes	A collection of characteristics that can be used to identify people or objects.	AGE, SEX, ZIP
Sensitive attributes	Sensitive information that needs to be concealed	SALARY, DISEASE

Firstly, Table I introduces about three main types of attributes that a dataset might have with the descriptions of each type. In Table II, we have an example dataset D about salary information of employees in a company. In such data, ID is the explicit identifier. Normally, explicit identifiers will be deleted from the original dataset before sharing the dataset so that no one can know which record in the dataset belongs to whom. We assume salary is a sensitive attribute (i.e., the private information of one employee that he does not want to disclose). AGE, SEX, and ZIP are quasi attributes (i.e., the combination of the values of AGE, SEX, and ZIP may re-identify an exact employee with ID and SALARY). For example, the values 24, M, and 641015 of AGE, SEX, and ZIP will identify the user ID 1 (because only user 1 has such data) and then attacker can know the SALARY of user 1 (the sensitive data of user 1 is disclosed, i.e., the privacy is violated).

TABLE II. EXAMPLE OF DATASET D

ID	AGE	SEX	ZIP	SALARY
1	24	M	641015	78000
2	23	F	641254	45000
3	39	M	610002	85000
4	34	M	610410	20000
5	50	M	610410	50000

Assume that the data in Table II will be shared to certain people or organizations (i.e., data collectors). Clearly, if the original data in Table II are shared to the collectors (e.g., it is possible an attacker), the attacker can know about the sensitive data of the users as mentioned above. Thus, to preserve privacy, we need to modify the original data in Table II to hide the private information before sharing such data to collectors. Next, we show how to perform k-anonymity on the dataset D in Table II to hide the private data. Table III shows the output of Table II after applying a certain k-anonymity algorithm based on generalization and suppression such as OKA [6], KACA [7] and GCCG [8]. Because ID is the explicit identifier, it will be hidden after the anonymization. Salary is the sensitive attribute and intuitively, we can hide the salary data in the output Table III and so, the attacker cannot know the salary of any user (privacy is protected). However, in many real cases, the data in sensitive attributes contain valuable information for the data collector. Therefore, they must be kept as in the original data when sharing the data to collectors (e.g., because sensitive attributes are valuable data, so if we hide or delete them from the original data, the shared data becomes useless to the collectors). The data in each cell of the quasi attributes (AGE, SEX, and ZIP) are generalized to a more general one, e.g., the data 24 of AGE is modified to become 20–30. Now, based on Table III, the attacker can get difficulty in deciding the exact salary information of any user.

At this time, we are ready to share D’ (Table III) to the data collectors. By using some data mining techniques, the data collectors wish to extract some valuable information from D’ (for example, how many male employees whose income is over 70000). However, as you can see in the Table III, the data in quasi attributes are modified (generalized) to other values that are different from the original data, thus, when the collectors mine such data (using data mining techniques), the obtained results may not be valuable.

TABLE III. EXAMPLE OF 2-ANONYMITY D' OF D BASED ON GENERALIZATION AND SUPPRESSION

ID	AGE	SEX	ZIP	SALARY
*	20–30	ANY	641***	78000
*	20–30	ANY	641***	45000
*	31–50	ANY	610***	85000
*	31–50	ANY	610***	20000
*	31–50	ANY	610***	50000

In Table IV, we present an idea called tuple member mitigation to anonymize the Table II to obtain 2-anonymity: the data in tuples are not generalized as previous algorithm, they are transformed to existing values in the table (e.g., the value 641254 is transformed to existing value 641015). Thus, the collectors will receive the real data instead of "anonymized" data as in the generalization or suppression algorithms.

TABLE IV. EXAMPLE OF 2-ANONYMITY D' OF D BASED ON TUPLE MEMBER MIGRATION

ID	AGE	SEX	ZIP	SALARY
*	24	M	641015	78000
*	24	M	641015	45000
*	39	M	610002	85000
*	39	M	610002	20000
*	39	M	610002	50000

Association rule mining is one of the most popular techniques in data mining [12]. A rule is represented by ($lhs \rightarrow rhs$) where *r* is the identity of the rule, lhs and rhs are the sets of values of certain attributes in the data set. For example, $r_1(\{A, B\} \rightarrow \{C\})$ (where A, B, C are certain values of three attributes in the dataset) is a rule. The mining rules are characterized by two fundamental parameters: support and confidence [12]. Support of the rule r_1 is calculated by the percentage of records which include all items' values in r_1 (i.e., A, B, and C). Confidence of the rule r_1 is calculated by the percentage of records which support all items in r_1 (i.e., A, B, and C) with records supporting the left hand side of it (records which have A and B).

- Given:

– $frq(X, Y, \ldots)$: is a counting function, returns the number of records that contain both items X, Y, … in the dataset.

– N: Total number of records in the dataset

- Then:

$$-\ su(r_1) = \frac{frq(A, B, C)}{N}$$

$$-\ confidence\ (r_1) = \frac{frq(A, B, C)}{frq(A, B)}$$

In this paper, we focus on the association rule technique of the data mining. Now, the context of the sharing data is the following (see Fig. 1): the owner of the data (called O) has a data table (called D) that contains sensitive data, quasi data, and identifiers. Some other organizations or people (called C) request for the data for their purposes (e.g., to analyse and mine the data using the association rule technique to obtain valuable information). To preserve the privacy of the user having the sensitive data, O must use a k-anonymity algorithm to anonymize the original data D to obtain the D' table. Then, the table D' is shared to C. C will use the association rule technique to mine the table D' to get valuable information. As mentioned above, the association rule technique is based on association rules that have support and confidence values. Clearly, if all association rules in table D (especially high-support and high-confidence rules) are maintained in table D' after the anonymization, the data mining process on table D' will output the same result as on D. In the next Section, we propose our k-anonymity algorithm based on tuple member mitigation that tries to maintain *significant* association rules (i.e., rules with high support and confidence) in the original data. In the literature, we can use the two improvements of k-anonymity, called l-diversity and t-closeness, for privacy protection. Actually, in order to guarantee l-diversity and t-closeness, we need to obtain k-anonymity first and then use some specific algorithms to satisfy l-diversity or t-closeness [16, 17]. Thus, k-anonymity is a good point to start the privacy protection. The algorithms that we use to transform a dataset satisfying k-anonymity to l-diversity or t-closeness are considered as our future works.

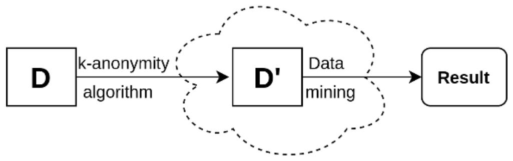


Figure 1. Anonymize and mine process.

III. THE PROPOSED ALGORITHM

As discussed above, ideally, the algorithm will try to retain the association rules in the original data while guaranteeing k-anonymity. However, it is difficult to retain all association rules because the number of the association rules may be very big. Normally, the data mining process (based on association rule mining technique) only considers association rules which occur frequently in the database. Therefore, the anonymization algorithm should focus on retaining these rules. We call such rules as significant rules. In our proposed algorithm, two thresholds min_sup and min_conf are provided to specify whether an association rule is significant or not. An association rule is significant if its support value is greater than min_sup and its confidence value is also greater than min_conf (Note that the values of support and confidence of a rule range from 0 to 1, i.e., 0% to

100%, see Section II). Conversely, the association rule is considered as insignificant.

We state the configuration for the proposed algorithm as follows:

- Input: D, R, k, min_sup, min_conf, and QI
- Output: D' that satisfies k-anonymity

where D is the original dataset, R is the set of association rules extracted from D, k is the parameter of the anonymization (k-anonymity), min_sup and min_conf are thresholds for support and confidence values, QI is the set of quasi-identifier attributes in D, D' is the output dataset. D' must satisfy k-anonymity and we need to maintain as many as possible the significant association rules in D' (i.e., minimize the loss of significant association rules when anonymizing D).

Next, we illustrate some definitions and notions that are used in our algorithm:

Definition: A group g is a set of tuples (or records) in the dataset. Moreover, all tuples in a group must have the same QI values. A group g satisfies k-anonymity if it has at least k tuples, i.e., $|g| \geq k$, or has no tuples in it (we call k-safe group). Otherwise, the group is k-unsafe (i.e., $|g| < k$).

Definition: A tuple migration between two groups g_i and g_j ($g_i \to g_j$) changes all QI values of some tuples in g_i to the correlative values in g_j. For example, assume that group g_i has two tuples with QI is (x1, y1, t1) and group g_j has three tuples with QI is (x2, y2, t2), the migration $g_i \to g_j$ will form group g_j which has five tuples. The additional tuples are from group g_i and their QI attributes are changed to (x2, y2, t2).

As discussed above, the algorithm should retain significant association rules in the original data set. Assume that the rule $r_1(\{A\}\to\{B\})$ is significant in the original data set. It means that the support and confidence values of this rule are greater than the thresholds min_sup and min_conf. When we perform a tuple migration, it is possible to alter some values of QI attributes of tuples supporting this significant rule. The result is that this tuple may no longer support the rule. Clearly, in case we alter too more tuples supporting the rule, it may become insignificant. Thus, for each rule, we need to calculate the maximal number of tuples which we can alter through tuple migrations so that the significant rule is still significant.

Another possible situation may happen when performing the migrations is that an insignificant association rule may become a significant one or a new significant rule may be generated. This may also lead to affecting the datamining result. Thus, we also need to calculate the maximum number of tuples which we can alter without generating new significant rules.

A. Budget Calculation

Assume that we have a significant association rule $r_1(\{A\}\to\{B\})$, s and c are the support and confident values of this rule, respectively. Since the rule is significant, we have: $s>=min\_sup$ and $c>=min\_conf$. That are:

$$s = \frac{frq(A,B)}{N} >= \min\_\sup \quad (1)$$

$$c = \frac{frq(A,B)}{frq(A)} >= \min\_conf \quad (2)$$

where N is the number of tuples in the data set. After performing the tuple migrations, we consider the following cases:

Case 1: The migrations change only A in some tuples

Assume n is the number of tuples which are anonymized by the migrations (that change A), s' and c' are the support and confident values of the rule after performing the migrations. For our approach, we need to preserve the significance of the rule, then $s'>=min\_sup$ and $c'>=min\_conf$:

$$c' = \frac{frq(A,B)-n}{frq(A)-n} >= \min\_conf \quad (3)$$

$$s' = \frac{frq(A,B)-n}{N} >= \min\_\sup \quad (4)$$

Thus, the maximal number of tuples, which can be anonymized to guarantee r_1 is still significant, is:

$$n = MIN\left(N\times(s-min\_sup)\left\lfloor\frac{s\times N\times(c-min\_conf)}{c\times(1-min\_conf)}\right\rfloor\right) \quad (5)$$

Case 2: The migrations change only A in some tuples: Similarly, we have:

$$c' = \frac{frq(A,B)-n}{frq(A)} > \min\_conf \quad (6)$$

$$s' = \frac{frq(A,B)-n}{N} > \min\_\sup \quad (7)$$

Thus, the maximal number of tuples, which can be anonymized to guarantee r_1 is still significant, is:

$$n = MIN\left(N\times(s-min\_sup)\left\lfloor\frac{s\times N\times(c-min\_conf)}{c}\right\rfloor\right) \quad (8)$$

Case 3: The migrations change both A and B in some tuples: We notice that this case is similar to the case 1.

Similarly, our proposed algorithm also need to guarantee that an insignificant association rule $r_2(\{A\}\to\{B\})$ will not become a significant one or a new significant rule is generated. Let s and c be the support and confidence values of this rule before performing the migrations, s' and c' are the corresponding values after performing the migrations. n is the number of additional tuples that support the rule r_2. Then, we have:

$$c = \frac{frq(A,B)}{frq(A)} < \min\_conf \quad (9)$$

$$s = \frac{frq(A,B)}{N} < \min\_\sup \quad (10)$$

We consider the following cases:

Case 1: The additional tuples support only A after the migrations

Then, we have:

$$c' = \frac{frq(A,B)}{frq(A)+n} \quad (11)$$

Clearly, c' is always smaller than c and s' is equivalent to s. Therefore, this rule cannot become a significant rule.

Case 2: The additional tuples support only B after the migrations. Similar to case 1, the rule cannot become a significant rule.

Case 3: The additional tuples support both A and B after the migrations.
Then, we have:

$$c' = \frac{frq(A,B)+n}{frq(A)+n} < \min\_conf \quad (12)$$

$$s' = \frac{frq(A,B)+n}{N} < \min\_\sup \quad (13)$$

Therefore, the maximal number of additional tuples, which can be anonymized to support r_2 without making r_2 to become significant, is:

$$n = MIN\left(N\times(min\_sup - s), \left\lfloor \frac{s\times N\times(min\_conf - c)}{c\times(1-min\_conf)} \right\rfloor\right) \quad (14)$$

Our proposed algorithm will use these maximal numbers to calculate the budget for each rule. Normally, the budget of a significant rule is the maximal number of tuples, which can be anonymized to guarantee the rule is still significant (see Eq. (8)). Similarly, the budget of an insignificant rule is the maximal number of additional tuples, which can be anonymized to support the rule without making the rule to become significant (see Eq. (14)).

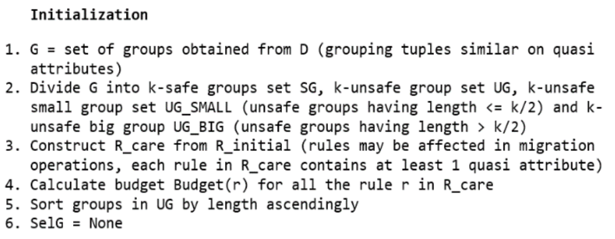
B. *The Algorithm*

Firstly, we introduce some notions and variables used in the algorithm:

- G: Set of groups generated from D by grouping tuples having same values on quasi attributes
- $|g|$: Length of group g, or number of tuples in g
- SG: Set of k-safe groups
- UG: Set of k-unsafe groups
- UM: Set of un-migrant groups, these groups will be dispersed in the last phase
- UG_SMALL: Set of groups having length less than or equal $k/2$
- UG_BIG: Set of groups having length greater than $k/2$
- $origin(g)$: Set of original tuples of group g
- $foreign(g)$: Set of received tuples in group g
- $R\_initial$: Set of rules initially mined on D at min_sup and min_conf
- R_care: Subset of $R\_initial$, set of rules that contain at least one quasi attribute
- $R\_affected$ (g_i, g_j): Set of rules affected by a tuple migration from g_i to g_j
- $r(lhs \rightarrow rhs)$: A rule with 2 fundamental parameters $support$ and $confidence$
- $B(r)$: $Budget$ which the rule r holds as mentioned above.
- $Risk$ (g): If group g is safe ($|g| \geq k$) then $Risk$ (g) = 0. Otherwise $Risk$ (g) = $k - |g|$.

A tuple migration operation between two groups g_i and g_j ($g_i \rightarrow g_j$) is valuable if it causes the sum $Risk$ (g_i) + $Risk$ (g_j) decreases. The optimal case is when this sum reaches 0 (i.e., the two groups become k-safe). Thus, in our algorithm, we need to select the best valuable migrations and this one is one of the most important criteria.

We present the pseudocode of the algorithm as in Figs. 2–4. We explain more details about each phase of the algorithm as follows.



```
Initialization

1. G = set of groups obtained from D (grouping tuples similar on quasi
   attributes)
2. Divide G into k-safe groups set SG, k-unsafe group set UG, k-unsafe
   small group set UG_SMALL (unsafe groups having length <= k/2) and k-
   unsafe big group UG_BIG (unsafe groups having length > k/2)
3. Construct R_care from R_initial (rules may be affected in migration
   operations, each rule in R_care contains at least 1 quasi attribute)
4. Calculate budget Budget(r) for all the rule r in R_care
5. Sort groups in UG by length ascendingly
6. SelG = None
```

Figure 2. The initial stage.

In the first phase, called the *initial stage* (Fig. 2), we initialize some groups of tuples for the later steps: *G, SG, UG, UG_BIG, UG_SMALL*, and *R_care*. The groups are formed and then categorized as k-safe groups (in SG) and k-unsafe groups (in *UG*). Groups in *UG* are then divided into *UG_BIG* (i.e., containing k-unsafe groups having more than k/2 tuples) and *UG_SMALL* (i.e., containing k-unsafe groups having less than or equal to k/2 tuples). Such groups are then sorted by their lengths (the number of tuples) ascendingly. Then, the algorithm forms the set of significant rules (R_initial) (i.e., the support and confidence values of the rules are greater than *min_sup* and *min_conf*, respectively). However, not all significant rules in R_initial are affected by tuple migrations. Only rules that contain at least one quasi attribute are affected by the migration. Thus, we need to extract such rules from R_initial, (called R_care). One important step in this phase is calculating the budgets for each rules in R_care. As mentioned above, budgets are based on Eq. (8) and Eq. (14).

The central idea of the second and also the main phase—Process stage (Fig. 3) is heuristically finding the most suitable group pairs and then performing tuple migration between them so that the tuple migration is the most valuable as mentioned above. The algorithm starts processing the unsafe groups then for each selected group $SelG$, it searches for a suitable group, in the set of all remaining ones, ($remaining\_groups$), so that the migration operation between this pair achieves an optimized result on risk reduction. When performing a migration operation of tuples from a group gi to a group g_j ($g_i \rightarrow g_j$), we need to deduct the budget of affected rules following the calculations in Section A.

```
Process

1.   while |UG| > 0 or SelG:
2.       if SelG is None:
3.           SelG = UG.pop(0)  // Pop the first unsafe group
4.           UG_BIG.remove(SelG); UG_SMALL.remove(SelG)
5.       if |SelG| <= k/2:
6.           remaining_groups = UG_BIG + UG_SMALL + SG
7.       else:
8.           remaining_groups = UG_SMALL + UG_BIG + SG
9.       Find group g to migrate with SelG in remaining_groups
         g = find_group_to_migrate(SelG, remaining_groups)
10.      if g is None:    // No result, continue next iteration
11.          UM.add(SelG); SelG = None    // SelG will be dispersed
12.      else:      // There is a result, consider the pair (SelG, g)
13.          If migrating n tuples from SelG to g:
14.              Update budget of every rule r in
                 R_affected(SelG, g): Budget(r) -= 1
15.          If migrating n tuples from g to SelG:
16.              Update budget of every rule r in
                 R_affected(g, SelG): Budget(r) -= 1
17.          If SelG is safe:
                 SG.add(SelG); UG.remove(SelG);
                 UG_BIG.remove(SelG); UG_SMALL.remove(SelG)
18.          If g is safe:
                 SG.add(g); UG.remove(g);
                 UG_BIG.remove(g); UG_SMALL.remove(g)
19.          if SelG is safe and g is safe: SelG = None
20.          else: SelG = unsafe group in pair {SelG, g}
     endwhile
21.  if |UM| > 0 for each um in UM: Pop um from UM; Disperse(um)
```

Figure 3. The process stage.

For the final phase—Disperse stage (Fig. 4), the algorithm handles remaining unsafe groups UM (i.e., the unsafe groups that the algorithm cannot find a valuable tuple migration for them) by dispersing them. Usually, the number of groups remaining at this step is small, so it does not require much computational effort. At the end, there is no more unsafe group and the data set achieves k-anonymity.

```
Disperse(um)

Begin

1.   For each tuple t of other groups that has migrated into SelG:
2.       Return t to its initial group; |g| += 1
3.       Update budget of every rule r in R_affected(g, SelG):
             Budget(r) += 1
4.       if |g| = 1 UM.add(g)
5.   For each remaining origin member t of SelG
6.       g = find_group_to_move_dispersing(SG)
7.       Update budget of every rule r in R_affected(SelG, g):
             Budget(r) -= 1
End
```

Figure 4. The disperse stage.

IV. Experiments and Analysis

We assessed the performance and data utility of our improved algorithm on Adult (UCI Machine Learning: http://archive.ics.uci.edu/ml/) dataset. First of all, we performed preprocessing by removing records with *null* or *unknown* values and retained only 9 attributes [*age, sex, marital—status, native—country, race, education, hours—per—week, capital—gain, workclass*]. The input dataset then has 32,169 records. Then we choose the first 6 attributes as *quasi attributes* while having chosen *min_sup* and *min_conf* both equal to 0.5. In general, *min_sup* and *min_conf* can be set to any value (in the range 0–1) in our proposed algorithm (see Section III). However, in this paper, for the purpose of comparison to the other algorithms that cannot set *min_sup* and *min_conf*, we choose 0.5 as the default value. We intend to evaluate the effect of changing *min_sup* and *min_conf* to the performance of our algorithm. Such work is left as our future work.

We implemented our proposed algorithm (U-M3AR) and 3 other state-of-the-art algorithms OKA [5], GCCG [7] and M3AR [10] in Python 3 and ran all the experiments on a *Windows 10, Intel Core I5 3570 3.4GHz, RAM 8GB* machine. Then we assessed and compared the performance and mining utility through five metrics [10]: Running time, Average Group Size (CAVG), Lost Rules Percentage (LRP), New Rules Percentage (NRP), Different Rules Percentage (DRP) (see Table V).

TABLE V. THE METRICS USED FOR THE EVALUATION

Metric	Description
Running time	The processing time of algorithms
Average Group Size (CAVG)	The average size of generated groups during the process. $CAVG = \frac{Total\ length\ of\ all\ groups}{Number\ of\ groups} \Big/ k$ High value of CAVG is good. It means that the generated groups have a high possibility to reach k.
Lost Rules Percentage (LRP)	The percentage of significant association rules lost after running algorithms. $LRP = \frac{\lvert R - R^{'} \rvert}{\lvert R \rvert}$ R is the rule set in D, R′ is the rule set in D′. Low value of LRP is good. It means that the number of significant rules lost is low.
New Rules Percentage (NRP)	The percentage of significant association rules lost after running algorithms. $NRP = \frac{R^{''}}{\lvert R \rvert}$ R" is the set of new association rules generated (in the set D′). after running algorithms. R is the rule set in D. Low value of NRP is good. It means that the number of rules generated that may affect the data quality is low.
Different Rules Percentage (DRP)	The percentage of difference in the set of rules before and after running algorithms $DRP = \frac{\lvert R - R^{'} \rvert R^{''}}{\lvert R \rvert}$ R" is the set of new association rules generated in the set D′. R′ is the rule set in D′. R is the rule set in D. Low value of DRP is good. It means that there is a little change in the set of rules.

According to the experiment results in Figs. 5–9, our proposed algorithm (U-M3AR) is more efficient than M3AR, concretely the running time is always lower while LRP always equals 0. Regarding OKA, the increase of *k* value leads to the decrease of the running time. Besides, this running time is also lower than M3AR and U-M3AR with k $\geq$ 20. The elapsed time of the GCCG algorithm always exceeds the others. We also took a look at the algorithms OKA and OCCG and found that for all the tuples that do not belong to any safe group (called unsafe tuples), the two algorithms transform them to a random safe group. Thus, the algorithms only need to select a safe group and transform all the tuples to such a group. Such steps do not need much processing time. That is why when k increases (this may lead to more and more unsafe tuples), the running time of the two algorithms decreases. However, this also decreases the quality of data because many significant association rules in the original data are lost (see Fig. 6). Conversely, our proposed algorithm will select the most "suitable" group to transform the unsafe tuples based on the heuristics so that the migration operation between this pair achieves an optimized result on risk reduction (see again Section III). Clearly, such steps require processing time but it is worth to maintain the quality of data.

In other metrics (Fig. 7), CAVG, our proposed algorithm outputs better results (higher values of CAVG) in comparison to the others. At every different value of k, the OKA gives the output of CAVG lower than both M3AR and U-M3AR, while GCCG always gives the lowest CAVG among the algorithms.

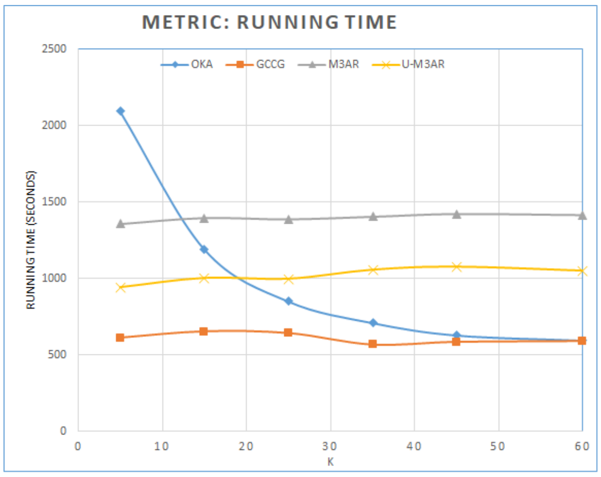


Figure 5. Results on metric running time.

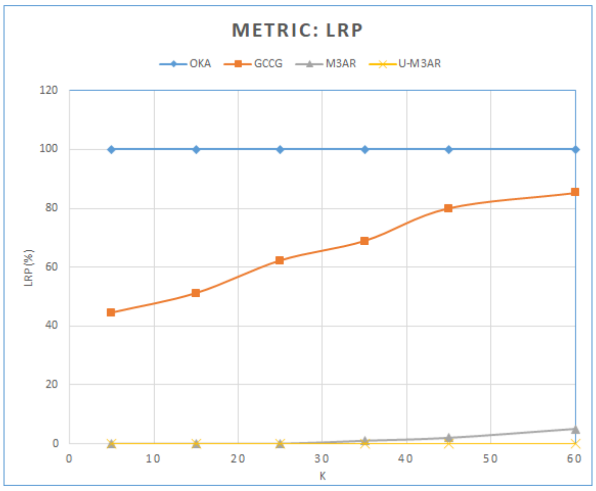


Figure 6. Results on metric Lost Rules Percentage (LRP).

With k $<=$ 30, NRP of our proposed algorithm also equals to 0 while with $k >$ 30, this metric is less than 20% (see Fig. 8). Although there are some new association rules generated, they do not affect to the quality of data because the support and confidence values of them do not exceed min_sup and min_conf, respectively. GCCG and OKA are algorithms based on local generalization and clustering, they are general so the data quality does not remain high as some significant association rules completely are lost.

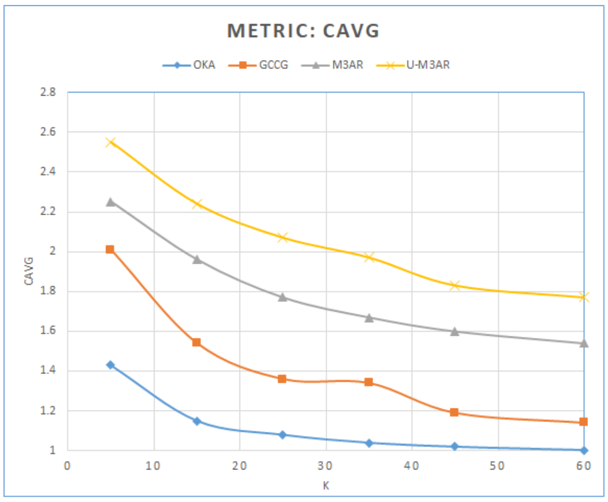


Figure 7. Results on metric Average Group Size (CAVG).

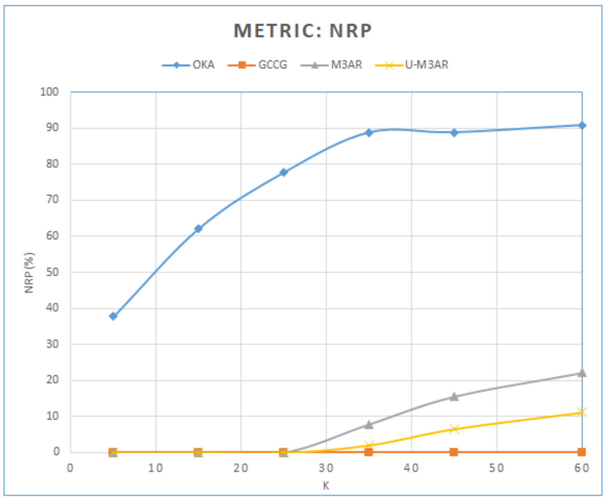


Figure 8. Results on metric New Rules Percentage (NRP).

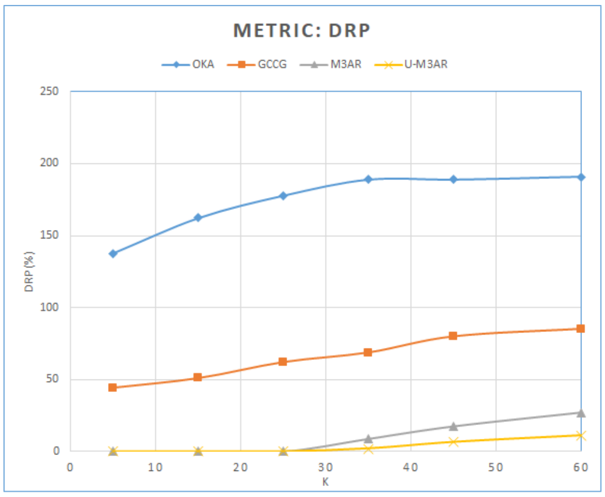


Figure 9. Results on metric Different Rules Percentage (DRP).

Finally, Fig. 9 shows the combination of the rules generated and the rules lost. Again, the low DRPs of our proposed algorithm confirm the effectiveness in maintaining the quality of data of the algorithm in comparison with OKA and OCCG.

V. Conclusions

In this paper, we propose an effective algorithm that preserves data privacy while retaining data mining quality. The main idea of the algorithm is to use the tuple member migration technique idea to select the most "suitable" safe group to transform the unsafe tuples. This is also one of the main advantages of our solution in comparison to state-of-the-art techniques and algorithms using generalization and suppression approaches. We perform experiments to evaluate the performance and data utility of our proposed technique in comparison with state-of-the-art anonymization techniques and the results prove the effectiveness of our proposed algorithm.

In the future, we can apply this technique to other mining techniques such as clustering or classification. Another direction is to extend the proposed technique to achieve l-diversity and t-closeness. We can also parallel some steps in the algorithm to improve the performance with the intention that it can be applied to big datasets.

Conflict of Interest

The authors declare no conflict of interest.

Acknowledgement

The author acknowledges Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

References

[1] E. Simperl, K. O'Hara, R. Gomer, "Open data and privacy," Report, University of Southampton, Electronics and Computer Science, Southampton: European Data Portal, 2016.

[2] L. Sweeney, "K-anonymity: A model for protecting privacy," *IEEE Security and Privacy*, vol. 10, pp. 1–14, 2002.

[3] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubra-maniam, "l-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data—TKDD*, vol. 1, p. 24, 2006.

[4] N. Li, T. Li, and S. Venkatasubramanian, "T-closeness: Privacy beyond k-anonymity and l-diversity," in *Proc. IEEE 23rd International Conference on Data Engineering (ICDE)*, 2007, vol. 2, pp. 106–115.

[5] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain k-anonymity," in *Proc. the ACM SIGMOD International Conference on Management of Data*, 2005, pp. 49–60.

[6] J.-L. Lin and M.-C. Wei, "An efficient clustering method for k-anonymization," in *Proc. the 2008 International Work-shop on Privacy and Anonymity in Information Society*, PAIS '08, (New York, NY, USA), Association for Computing Machinery, 2008, pp. 46–50.

[7] J. Li, R. C.-W. Wong, A. W.-C. Fu, and J. Pei, "Achieving k-anonymity by clustering in attribute hierarchical structures," in *Proc. the 8th International Conference on Data Ware-housing and Knowledge Discovery*, DaWaK'06, Springer-Verlag, 2006, pp. 405–416.

[8] S. Ni, M. Xie, and Q. Qian, "Clustering based k-anonymity algorithm for privacy preservation," *International Journal of Network Security*, vol. 19, pp. 1062–1071, 2017.

[9] Y. Yang and Y. Zhou, "A survey on privacy-preserving data mining methods," in *Proc. IOP Conference Series: Materials Science and Engineering*, 2020, vol. 782, Qingdao, China.

[10] S. Goel and S. Sharma, "Nonhomogenous anonymization approach using association rule mining for preserving privacy," in *Proc. the 2nd International Conference on Electronics, Communication and Aerospace Technology*, 2018.

[11] L. Zhang, W. Wang, and Y. Zhang, *Privacy Preserving Association Rule Mining: Taxonomy, Techniques, and Metrics*, 2019.

[12] A. Tiwari and M. Choudhary, "A review on k-anonymization techniques," *Scholars Journal of Engineering and Technology (SJET)*, pp. 238–245, 2017.

[13] T. Dang, J. Küng, and V. Q. P. Huynh, "Protecting privacy while discovering and maintaining association rules," in *Proc. 2011 4th IFIP International Conference on New Technologies, Mobility and Security*, 2011, pp. 1–5.

[14] T. Dang and A. Truong, "Anonymizing but deteriorating location databases," *Polibits*, vol. 46, pp. 73–81, 2012.

[15] A. Truong, T. Dang, and J. Küng, "On guaranteeing k-anonymity in location databases," *Database and Expert Systems Applications, Lecture Notes in Computer Science*, vol. 1, pp. 280–287, 2011.

[16] R. Mortazavi and S. Jalili, "Fast data-oriented microaggregation algorithm for large numerical datasets," *Knowledge-Based Systems*, vol. 67, 2014.

[17] X. Xiao and Y. Tao, "Anatomy: Simple and effective privacy preservation," in *Proc. the 32nd International Conference on Very Large Data Bases*, 2006, pp. 139–150.