IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 30, 2019, accepted August 14, 2019, date of publication August 19, 2019, date of current version September 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936301

Efficient L-Diversity Algorithm for Preserving Privacy of Dynamically Published Datasets

ODSUREN TEMUUJIN[1], JINHYUN AHN[2], AND DONG-HYUK IM[1]
[1]Department of Computer and Information Engineering, Hoseo University, Asan 31499, South Korea
[2]Department of Management Information Systems, Jeju National University, Jeju 63243, South Korea

Corresponding author: Dong-Hyuk Im (dhim@hoseo.edu)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korean Government (MSIT) under Grant NRF-2017R1C1B1003600, in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program Supervised by the Institute for Information & Communications Technology Promotion (IITP), under Grant IITP-2019-2018-0-01417, in part by the IITP Grant funded by the Korean Government (MSIP) under Grant R0113-15-0005, in part by the Development of a Unified Data Engineering Technology for Large-scale Transaction Processing and Real-Time Complex Analytics, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2018R1D1A1B07048380.

ABSTRACT Although most conventional methods of preserving data privacy focus on static datasets, which remain unchanged after processing, real-world datasets may be dynamically modified often. Therefore, privacy-preservation methods must maintain data privacy after dataset modification. Re-anonymization of entire datasets is inefficient when large datasets are frequently modified. Although several previous studies have addressed data privacy for incremental data updates (i.e., record insertions), they have not adequately it for dynamic changes made to existing datasets (i.e., record updates and deletions). Therefore, we identified limitations of data-privacy preservation for dynamically evolving datasets and used anatomy instead of generalization and suppression to develop a more efficient l-diversity algorithm for preserving privacy of such datasets. We also used a Cuckoo filter, a new probabilistic data structure for approximate set-membership tests, to improve data-processing efficiency. Experimental results demonstrated that our proposed data-anonymization algorithm processed data more efficiently than other conventional algorithms, requiring much less running time than conventional re-anonymization of entire datasets. The Cuckoo-filtered algorithm was especially efficient, dramatically reducing operation execution times while maintaining privacy of dynamically evolving datasets.

INDEX TERMS Anonymization, Cuckoo filter, dynamic data publishing, l-diversity, privacy-preservation.

I. INTRODUCTION

Privacy-preservation issues have emerged with the increasing volumes of published data including sensitive, private information belonging to individuals and organizations. To address such issues, various methods of reducing risks associated with published data have been developed. Data anonymization is a widely used method of ensuring data privacy. Several anonymization models have also been developed for preserving privacy in data publishing. Existing anonymization models are commonly developed using tabular data structures (e.g., *k*-anonymity and *l*-diversity) wherein each row of the table is a tuple and each column is an attribute, which comes in several types such as: explicit identifiers, quasi-identifiers, sensitive attributes (SAs), and non-sensitive attributes. Explicit identifiers explicitly identify individuals (e.g., name and personal ID). Quasi-identifiers are sets of attributes that can potentially identify individuals. SAs describe some sensitive information about individuals. In the anonymity model, quasi-identifiers are anonymized for publishing because they cannot be distinguished. A set of tuples that cannot be distinguished by its quasi-identifiers, is called an equivalence class (EC).

The most well-known anonymity model is *k*-anonymity [1], which provides privacy protection by rendering data indistinguishable from at least *k*-1 other data. Numerous algorithms have been developed for *k*-anonymity and used in practice [2]–[7]. However, sensitive information anonymized using k-anonymity is not entirely secure and is vulnerable to some attacks. *L*-diversity [8] overcomes limitations of

The associate editor coordinating the review of this article and approving it for publication was Cristina Rottondi.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

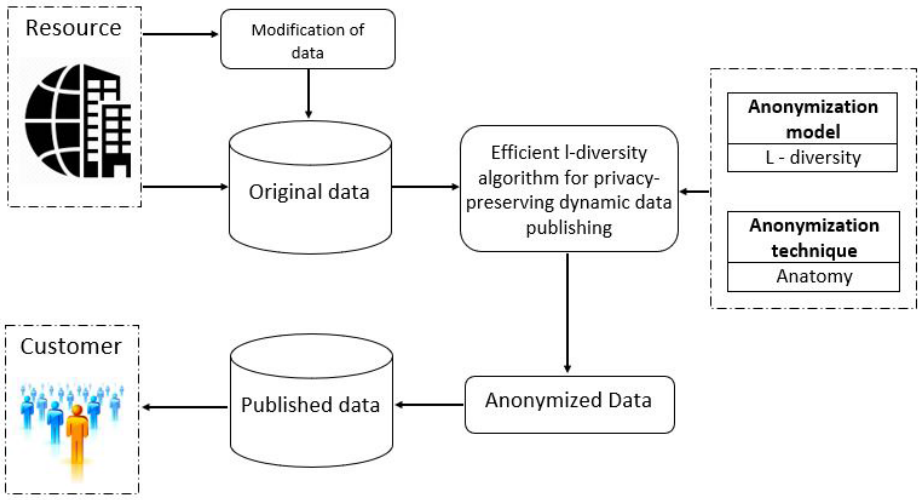


FIGURE 1. Proposed data-anonymization method.

k-anonymity and guarantees stronger privacy preservation than *k*-anonymity. An *l*-diverse dataset requires a least *l* distinct values for a SA in each EC. Suppression, which removes or replaces one or more values in a dataset with a special value, and generalization, which transforms values into more general ones, are widely used anonymization operations in the *k*-anonymity and *l*-diversity models.

Anatomy [9] is the most widely used privacy preserving algorithm for the *l*-diversity model because it can achieve anonymity without using generalization or suppression. Instead, it releases two separate quasi-identifier tables (QITs) and a sensitive-value table (ST), thereby preserving privacy because QITs do not indicate SAs of any tuple(s). However, most *l*-diverse anonymity algorithms are applied only to static datasets, which remain unchanged after initial processing.

In real-world applications, however, many datasets may be changed, meaning that new data may be added to datasets while old data may be updated and/or deleted. Therefore, guaranteeing privacy is reduced to the problem of determining how to preserve privacy after such modifications. The main operations resulting in such data-modification problems are insert, delete and update. Although the simplest solution to such problems would be to re-anonymize entire modified datasets, larger datasets are frequently modified; therefore, re-anonymizing entire datasets would be inefficient. Although several previous studies have addressing problems associated with dynamically changing datasets are described more detail in Section 2, most such studies only have addressed incremental data releases and have inadequately addressed data deletion and updates. To address these shortcomings, we developed an efficient *l*-diversity method of preserving privacy of dynamically evolving datasets. Our method is based on the *l*-diversity anonymization model, and we used anatomy to preserve privacy, as shown in Fig. 1. Therefore, our method does not consider information quality of released(anonymized) data. Furthermore, we used probabilistic data structures, which compactly store low-memory data and provide approximate answers to queries about stored data, to improve data-processing efficiency. Main contributions of this paper to the literature are summarized as follows:

- We identified limitations of privacy preservation for dynamically evolving datasets.
- We developed a method of preserving privacy of dynamically evolving datasets, which surpasses defined limitations and focuses on the *l*-diversity anonymization model.
- We also used a Cuckoo filter [10], a new probabilistic data structure for approximate set-membership tests, to improve data-processing efficiency.
- We compared data-processing times of our method to those of conventional ones. Experimental results demonstrated that our method showed superior efficiency.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 describes basic concepts such as the *l*-diversity model and algorithm, and limitations on privacy preservation of dynamically changing datasets. In Section 4, we propose our method of preserving privacy of dynamically evolving datasets. Section 5 presents experimental results comparing data-processing times of our method to those of conventional methods. Finally, Section 6 summarizes the current work and main findings and suggests a direction of future research based on the findings of this study.

II. RELATED WORK

This section provides a brief review of conventional privacy-preservation models for data publishing. We also review previous methods of preserving privacy of dynamically evolving datasets and data streams.

TABLE 1. Original table.

Quasi-identifier			Sensitive attribute
Age	Gender	Zip code	Disease
45	Male	35124	Pneumonia
23	Male	35113	Bronchitis
51	Female	35001	Flu
35	Female	35066	Gastritis
33	Female	35246	Dyspepsia
33	Male	35243	Pneumonia

TABLE 2. Anonymized table of table I.

Quasi-identifier			Sensitive attribute
Age	Gender	Zip code	Disease
[45-23]	Male	351**	Pneumonia
[45-23]	Male	351**	Bronchitis
[35-51]	Female	350**	Flu
[35-51]	Female	350**	Gastritis
33	*	3524*	Dyspepsia
33	*	3524*	Pneumonia

A. K-ANONYMITY

K-anonymity, originally proposed by Sweeney [1], has received the most research attention among models aimed at preserving data privacy and numerous algorithms have been developed for *k*-anonymity and are used in practice [2]–[7]. Furthermore, several studies have been performed by applying such algorithms and large data platforms such as Apache Hadoop and Apache Spark to preserve privacy of large datasets [11]–[14]. A dataset satisfies *k*-anonymity when every record in the dataset is indistinguishable from at least *k*-1 other records with respect to quasi-identifier attributes. Datasets that satisfy *k*-anonymity are referred to as *k*-anonymous datasets. For example, Table 2 shows a 2-anonymous table with respect quasi-identifiers. Suppression, which removes or replaces one or more values in a dataset with a special value, and generalization, which transforms values into more general ones, are widely used anonymization operations in the *k*-anonymity model. A set of tuples that cannot be distinguished by its quasi-identifiers, is called an EC. For example, Table 2 shows three distinct ECs.

B. L-DIVERSITY

Although *k*-anonymity has gained popularity for the data privacy because of its simplicity, and numerous algorithms have been developed and applied to real-world datasets, sensitive information anonymized using k-anonymity is not entirely secure and is vulnerable to some attacks, as shown by Machanavajjhala *et al.* [8] and proved by examples in their research. They defined a novel privacy preservation model called the *l*-diversity anonymity wherein anonymized *l*-diverse datasets require at least *l* distinct values for the SAs in each EC. For example, Table 2 is 2-diverse with respect to SAs. Likewise, *l*-diversity has been well studied in the literature [9], [15].

C. T-CLOSENESS

Although *l*-diversity guarantees stronger privacy preservation than *k*-anonymity, *l*-diversity alone is insufficient to prevent attribute disclosure because it does not protect against a number of attacks identified by Li *et al.* [16]. Therefore, the concept of *t*-closeness has been proposed to address limitations of *l*-diversity. An EC is said to have *t*-closeness if distance between distribution of a SA therein and that of the attribute in the whole dataset is no more than threshold *t*. A dataset is said to have *t*-closeness if all ECs have *t*-closeness.

Stronger privacy preservation of *t*-closeness is beyond the scope of the present study, which focuses instead on the *t*-diversity model.

D. PRIVACY-PRESERVATION OF DYNAMICALLY EVOLVING DATASETS AND DATA STREAMS

A few studies have been performed on preserving privacy of dynamically published data. The first such study [17] proposed an algorithm based on dynamically changing *k*-anonymous datasets, thereby guaranteeing *k*-anonymity in databases wherein data were frequently added, deleted, and updated. However, *k*-anonymity does not protect sensitive information vulnerable to some attacks. Wang and Wang [18] proposed an *l*-diversity algorithm for incremental data publishing by considering inference attacks, which destroy the privacy guarantee of *k*-anonymity. Unfortunately, their algorithm was applied only to incremental data publication and did not consider record deletions and updates. Likewise, Byun *et al.* [19] presented an anonymization method for dynamically published data. However, they also considered only insertion operations. Additionally, Sun *et al.* [20] investigated maintaining *l*-diversity for dynamic data publishing and proposed a solution for data additions and deletions. Their algorithm used a generalization operation to ensure anonymization and focused on information loss thereby maintaining efficiency. We describe their work in more detailed in Section 3 and compare results obtained in their study to those obtained in ours in Section 5. Furthermore, Xiao presented *m*-invariance [21], a generalization-based principle, Wei *et al.* proposed *e*-inclusion [22] using a permutation-based method and substitution to anonymize microdata.

Preserving privacy of data streams is one of the practical areas in data privacy, wherein significant research has been conducted. Kim *et al.* [23] presented a framework for preserving privacy of electronic health data streams. Although most prior studies on preserving data-stream privacy have generated accumulation delay. Their proposed method immediately anonymized input streams with counterfeit values. Cao *et al.* [24] presented a novel framework called SABRE[1] for microdata anonymization based on the t-closeness principle. They developed two instantiations: SABRE-AK and SABRE-KNN, focusing on efficiency and

[1] Sensitive Attribute Bucketization and REdistribution framework

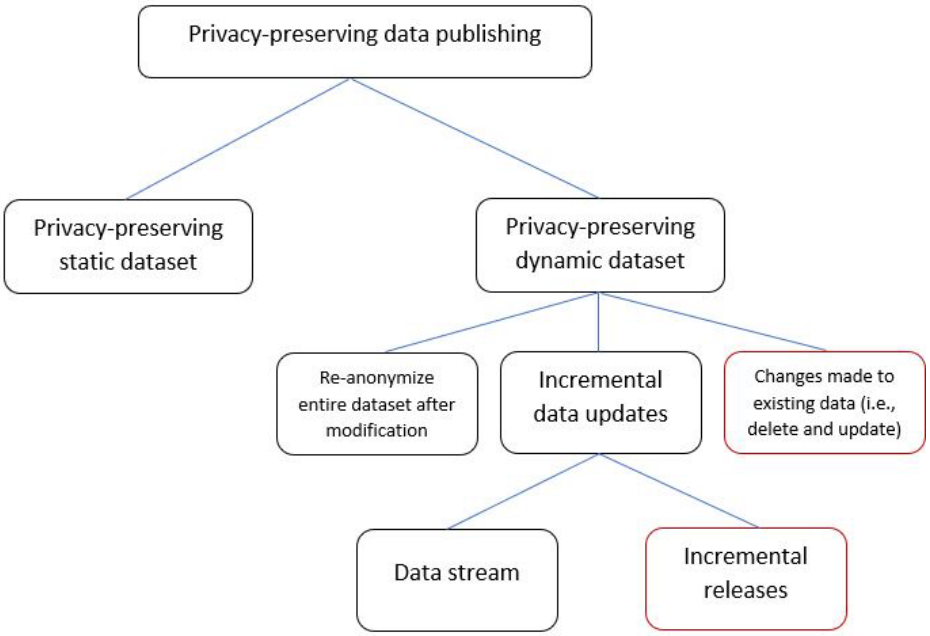


FIGURE 2. Classification of privacy-preservation data publishing.

information quality, respectively. They also extended SABRE to streaming data.

Classification of privacy-preservation data publishing is shown in Fig 2. The most attention-gaining anonymization models(i.e., *k*-anonymity, *l*-diversity, *t*-closeness) have been classified according to preserving privacy of released static data. This study considers classification of preserving privacy of incrementally released data dynamically updated or deleted as outlined in red in Fig 2.

TABLE 3. Equivalence classes corresponding to data in table I.

Quasi-identifier				Sensitive attribute
Group-Id	Age	Gender	Zip code	Disease
1	45	Male	35124	Pneumonia
1	23	Male	35113	Bronchitis
2	51	Female	35001	Flu
2	35	Female	35066	Gastritis
3	33	Female	35246	Dyspepsia
3	33	Male	35243	Pneumonia

III. BASIC CONCEPTS

Herein, we present an *l*-diversity anonymization model for data publishing and a corresponding *l*-diversity-based anatomy algorithm. We also describe various problems associated with preserving privacy in dynamically evolving datasets.

A. L-DIVERSITY

L-diversity is a privacy concept stronger than that introduced by Machanavajjhala *et al.* [8].

Definition 1 (L-Diversity Model): Let T be the initial micro-data table to be published. T contains d quasi-identifier (QI) attributes $QA_1, QA_2, \ldots, QA_d$ *and a sensitive attribute (SA). Although each* QA_i $(1 \leq i \leq d)$ *can be either numerical or categorical, SA should be only categorical, following the assumption of l-diversity [8].*

Definition 2 (L-Diversity Principle): An EC satisfies l-diversity when there are at least l "well-represented" values for SAs. A dataset satisfying l-diversity satisfies l-diversity for EC of the dataset.

To clarify, *"l well-represented"* means that each EC contains at least *l* different values for the SA.

Anatomy is one of the widely used *l*-diversity algorithms for preserving privacy in data publication, and was proposed by Xiao and Tao [9]. Anatomy can achieve anonymity without using generalization or suppression. Instead, it releases two separate QITs and an ST. First, anatomy computes *l*-diversity ECs of given datasets and assigns a unique ID, called an attribute Group-ID, to each EC. Then, it generates a QIT containing all QIs in the dataset along with the Group-ID. After that, anatomy generates an ST containing SAs of the dataset along with the Group-ID and a new attribute, Count, which the number of tuples in the QIT that have the same SA. For example, assume that we divide tuples in Table 1 into three ECs according to Group-ID, as shown in Table 3. Given these ECs, corresponding QIT and ST produced by anatomy are shown in Tables 4 and Table 5, respectively.

Anatomy preserves privacy because QIT does not indicate SAs of any tuples. For instance, suppose an adversary has information about a person aged 45 whose zip code 35124. Following this information, although the adversary knows

TABLE 4. Quasi-identifier table (QIT) corresponding to data in table III.

Age	Gender	Zip code	Group-Id
45	Male	35124	1
23	Male	35113	1
51	Female	35001	2
35	Female	35066	2
33	Female	35246	3
33	Male	35243	3

TABLE 5. Sensitive table (ST) corresponding to data in table III.

Group-Id	Disease	Count
1	Pneumonia	1
1	Bronchitis	1
2	Flu	1
2	Gastritis	1
3	Dyspepsia	1
3	Pneumonia	1

tuple 1 belongs to Table 4, it cannot obtain any sensitive information about the person belonging to tuple 1. Instead, the adversary can obtain only Group-ID of tuple 1. Using the obtained Group-ID, the adversary can know sensitive information contained in the current group. In this example, two different diseases are contained in the group belonging to tuple 1. Hence, the adversary cannot directly access sensitive information. The person belonging to tuple 1 could have *pneumonia* or *bronchitis* with 50% probability each.

B. CONVENTIONAL L-DIVERSITY-BASED ALGORITHM FOR PRESERVING PRIVACY OF DYNAMICALLY EVOLVING DATASETS

Herein, we describe the conventional l-diversity-based method of preserving privacy of dynamically evolving datasets. To the best of our knowledge, this method [20] is the only one described in the literature for preserving privacy of dynamically evolving datasets undergoing record, deletions and insertions. Thus, we compared experimental results obtained using our proposed method to results obtained for the published one. Byun *et al.* applied clustering [25] to preserve data privacy. Furthermore, their cluster-based method (i.e., ECs) was developed such that only minimal information was lost and total information loss of all clusters could be compute. They described two different problems in their work: how to efficiently update released clusters, thereby ensuring l-diversity when dataset ΔT^+ is added to or dataset ΔT^- is deleted from the original dataset. The solution to the first problem is that each tuple in ΔT^+ is added to a cluster, thereby increasing the minimum information loss to total information loss. If a cluster has more than l distinct tuples, it is split into two equal sub-clusters, as shown in Fig 3.

The solution to the second problem is that when each tuple in ΔT^- is deleted from it host cluster, if the resulting cluster has fewer than l remaining distinct tuples for SAs, all the remaining tuples are dispersed into other clusters, as shown in Fig 4.

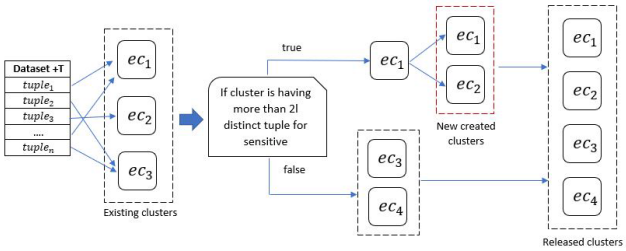


FIGURE 3. Record insertion by basic algorithm.

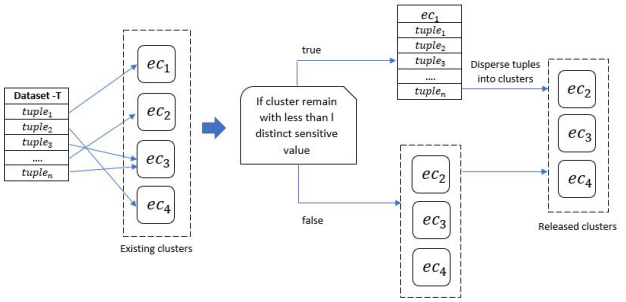


FIGURE 4. Record deletion by basic algorithm.

In this case, the algorithm finds a cluster wherein information loss is minimized after the tuple has been added to the dispersed cluster of tuples.

C. LIMITATIONS OF BASIC L-DIVERSITY ALGORITHM FOR PRESERVING PRIVACY OF DYNAMICALLY EVOLVING DATASETS

As previously mentioned, the basic method focuses on information loss. The algorithm calculates the minimum increase in total information loss when each tuple in dataset ΔT^+ (i.e., tuples added to the original dataset) has been added to a cluster. Furthermore, when records are deleted, each tuple in the cluster will be dispersed to another cluster if the remaining cluster has fewer than l distinct tuples for SAs, whereby the algorithm follows the same iterative process for calculating the minimum increase in information loss and total information loss for each tuple in dataset ΔT^+ and each tuple in the dispersed cluster. Therefore, the algorithm should determine information loss of all existing clusters in each iteration. If released anonymized datasets have many clusters and modified datasets consisting of numerous tuples, the algorithm can be inefficient. Furthermore, this study does not adequately describe how the algorithm was applied and does not include any specific information about algorithm implementation.

Considering these issues, we used anatomy instead of generalization and suppression to develop a more efficient algorithm for preserving privacy of dynamically evolving datasets. Therefore, we did not need to consider information loss and did not compare information quality in experimental evaluation.

IV. PRIVACY PRESERVATION FOR DYNAMICALLY EVOLVING DATASETS

As noted in Section 1, most *l*-diverse anonymity algorithms are only applied to static datasets. However, real-world datasets may be frequently modified by adding, updating, and deleting records. Preserving data privacy after such modifications is a significant issue, and these three operations in particular present corresponding unique problems. However, we assume we can directly solve problems associated with record additions and deletions, and we can solve the problem associated with record updates by combining record addition and deletion operations. To clarify, an update operation can be substituted by a combination of deletion and insertion ones. The following section describes how our proposed method performs insertion and deletion operations and how data privacy is preserved after each. It also introduces a probabilistic data structure for approximate set-membership tests and explains how we used the data structure in our method.

A. PROPOSED DATA-ANONYMIZATION METHOD

1) RECORD INSERTION

We first assume *l*-diversity ECs have been computed from the original dataset. Let the ECs be defined as follows:

$$ECs = \{ec_1, ec_2, \ldots ec_n\}$$

Definition 3: Let l be the specified anonymity requirement. A satisfied EC has more than l distinct SAs.

We also assume a set of tuples is added to the dataset.

Definition 4: Given the original dataset T_i let T_{i+1} be the revised dataset and $\Delta T^+ = \{t_1, t_2, \ldots t_n\}$ be the set of tuples added to T_i. Then T_{i+1} can be represented as follows:

$$T_{i+1} = T_i + \Delta T^+$$

An overview of maintaining data privacy after record insertion is shown in Fig. 5. If ΔT^+ contains more than l distinct tuples for an SA, we will compute new ECs from ΔT^+ and merge the *ECs*.

Theorem 1: If datasets A and B both satisfy the same l-diversity requirement, then the union of A and B ($A \cup B$) also satisfies l-diversity.

Proof: Let $SA_a = \{a_1, a_2, \ldots, a_n\}$ and $SA_b = \{b_1, b_2, \ldots b_n\}$ be sets of distinct tuples for an SA belonging to datasets A and B respectively, $|SA_a| = C_a$ and $|SA_b| = C_b$, where $|SA_i|$ represents cardinalities of SA_a and SA_b, respectively, and $C_a \geq l$, $C_b \geq l$. Let cardinality of union of the A and B be $|SA_a \cup SA_b| = C_U$.

- If the intersection of SA_a and SA_b (i.e., $SA_a \cap SA_b \neq \varnothing$) exists then $C_U \geq C_a$, hence; $C_U \geq l$.
- If the intersection of SA_a and SA_b (i.e., $SA_a \cap SA_b = \varnothing$) does not exist then $C_U = C_a + C_b$; thus, $C_U \geq l$.

□

If ΔT^+ does not contain l distinct tuples for an SA, each tuple t in T^+ is added to the smallest EC in the cluster of *ECs*.

Theorem 2: If every EC in a dataset satisfies the l-diversity requirement, then multiple record insertions into any EC also satisfy l-diversity.

Algorithm 1 Insertion

Input data: ***ECs***(*equivalence classes from original data*),
T+(*added tuples*)
l(*l-diversity value*)

1. $cnt = \textbf{\textit{distinct}}(T+)$
2. *//return count of distinct values for a sensitive value*
3. ***if*** $cnt >= l$
4. $\textbf{\textit{ECs}}_1 = \textbf{anatomy}(T+)$
5. *//return l-diverse equivalence classes*
6. $ECs = \textbf{\textit{merge}}\ (ECs_1)$
7. ***return*** Done
8. ***else***
9. ***for*** $i = 0; i < sizeOf\ (T+)\ ; i++$***do***
10. $tuple =$ get tuple from $T + [i]$
11. $EC =$ smallest sized equivalence class from *ECs*
12. add *tuple* to *EC*
13. ***return***Done

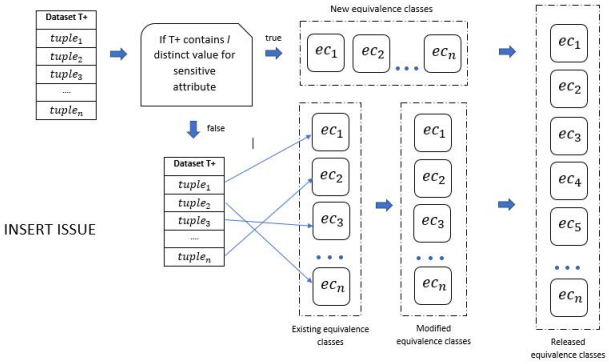


FIGURE 5. Maintaining data privacy after record insertion.

Proof: Let $SA = \{a_1, a_2, \ldots a_n\}$ be the set of distinct tuples for an SA of EC A such that $SA = \{a_1, a_2, \ldots a_n\} \geq l$. Furthermore, let multiple insertion $MI = \{m_1, m_2, \ldots m_n\}$ be the set of distinct tuples for an SA of MI. Then, there can be two cases of MI as follows:

- If $|MI| \geq l$, updated EC A satisfies l-diversity by Theorem 1.
- If $|MI| \leq l$, $|SA \cup MI| \geq l$ because $|SA| \geq l$. Therefore, updated EC A satisfies *l*-diversity.

□

Algorithm 1 guarantees that if any record is inserted into the original dataset, it can maintain anonymity by Theorems 1 and 2. The algorithm first maintains anonymity of an added dataset (ΔT^+) containing more than l distinct tuples for an SA (i.e., Lines 3-8 in Algorithm 1) and then maintains anonymity of an added dataset (ΔT^+) containing fewer than l distinct tuples for an SA (i.e., Lines 9-13 in Algorithm 1).

2) RECORD DELETION

We first assume that a set of tuples has been deleted from a dataset.

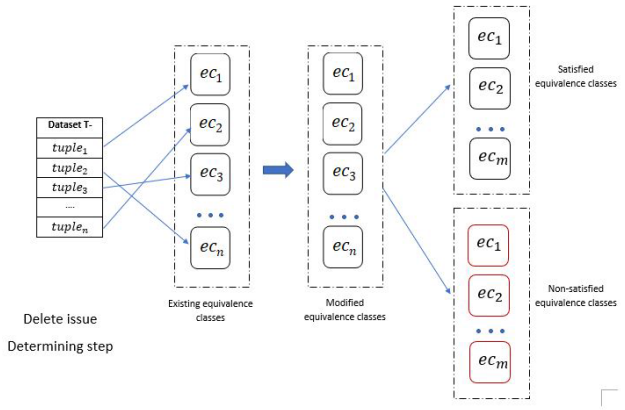


FIGURE 6. Maintaining data privacy after record deletion (determination step).

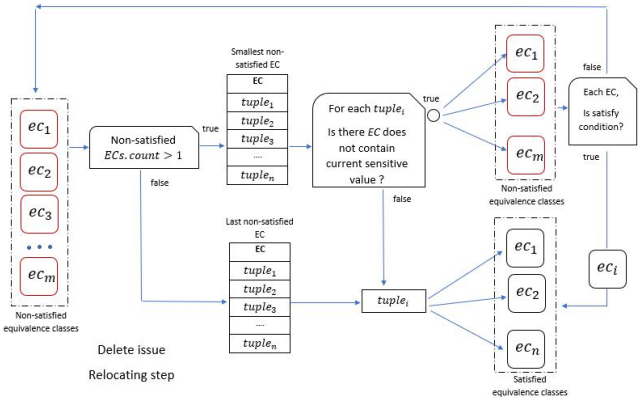


FIGURE 7. Maintaining data privacy after record deletion (relocation step).

Definition 5: Given the original dataset T_i, let T_{i+1} be the revised dataset and $\Delta T^- = \{t_1, t_2, \ldots t_n\}$ be the set of tuples deleted from T_i. Then T_{i+1} can be represented as follows

$$T_{i+1} = T_i - \Delta T^-$$

Each tuple t in ΔT^- will be deleted from its host EC. After record deletion, whether the revised EC still satisfies the specified anonymity requirement will be determined. An overview of maintaining data privacy after record deletion is shown in Figs 6 and 7.

Definition 6: Let l be the specified anonymity requirement. An EC having fewer than l distinct tuples remaining for the SA in the cluster is called a nonsatisfied EC (i.e., NT), which is defined as follows:

$$NT = \{ec_1, ec_2, \ldots ec_n\}$$

Theorem 3: If the entire dataset does not satisfy l-diversity, there exists at least one nonsatisfied EC in the dataset.

Proof by contrapositive: Assume that a dataset does not contain contain any nonsatisfied ECs. We prove that the entire dataset satisfies l-diversity.

Algorithm 2 Deletion

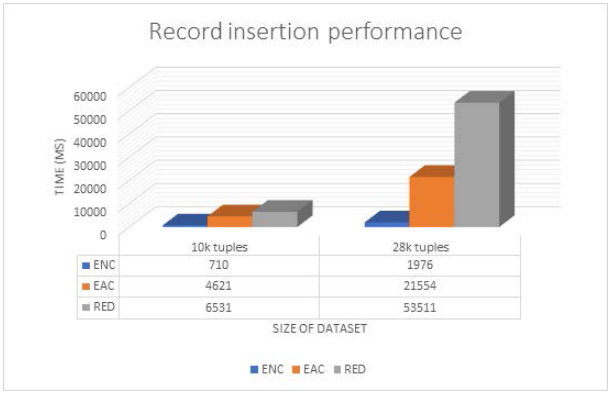
Input data: ***ECs***(*equivalence classes from original data*), ***T-***(*deleted tuples*) ***l***(*l-diversity value*)

1. $NT = \emptyset$
2. ***for*** $i = 0; i < sizeOf\ (T-)\ ; i++$ ***do***
3. $tuple$ = get tuple from $T - [i]$
4. EC = equivalence class currently containing $tuple$
5. *remove tuple* from EC
6. ***if*** EC having less than l remaining distinct tuple
7. *add EC* to NT
8. cnt = size of NT
9. *if* $cnt == 0$
10. ***return*** *Done*
11. ***while*** $cnt > 1$
12. EC = smallest equivalence class from NT
13. *remove EC* from NT and ECs
14. $cnt--$
15. ***for*** each $tuple$ in EC
16. ec = find equivalence class which is does not contain the sensitive value of $tuple$ from NT
17. ***if*** *is there possible ec*
18. add $tuple$ into ec
19. ***if*** ec satisfy $l-$diversity condition
20. remove ec from NT
21. $cnt--$
22. ***else***
23. EC = smallest equivalence class from ECs
24. *add tuple to EC*
25. EC = get last equivalence class from NT
26. *remove EC* from ECs
27. ***for*** $i = 0; i < sizeOf\ (EC)\ ; i++$ ***do***
28. $tuple$ = get tuple from $EC[i]$
29. EC = smallest equivalence class from ECs
30. *add tuple to EC*
31. ***return*** Done

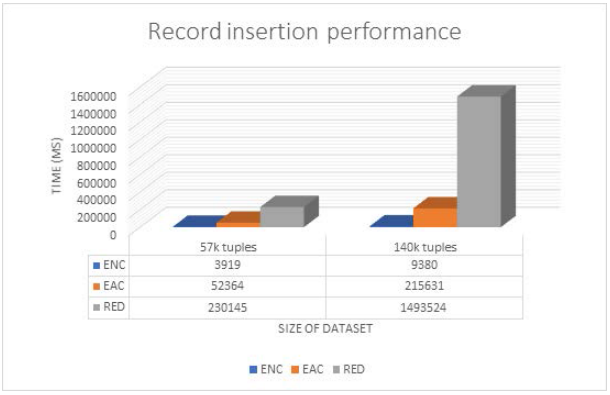
Let $EC_1, EC_2, \ldots EC_n$ be the EC and $k_1, k_2, \ldots k_n$ be the number of SAs. Because every $EC_i(k_i \geq l)$ has more than l distinct tuples for an SA, the entire dataset satisfies l-diversity by Theorem 1. □

Following Theorem 3, Algorithm 2 can maintain data anonymity when any record(s) is/are deleted from the original dataset. The algorithm first removes every tuple in ΔT^- from the host EC(Lines 2-5 in Algorithm 2), and then, determines nonsatisfied ECs from among all the existing ECs(i.e., Lines 6-7 in Algorithm 2).

If more than one nonsatisfied EC exists in NT, we select the smallest EC from NT and remove it from the ECs and NT (i.e., Lines 12-13 in Algorithm 2). For each tuple t in a selected EC, we find a new EC in NT that does not contain the SA of the current tuple (i.e., Line 16 in Algorithm 2). If an available EC is found, we insert tuple t into that EC (i.e., Line 18 in Algorithm 2).



(a) Record insertion (1)



(b) Record insertion (2)

FIGURE 8. Record insertion performance at four different datasets.

Otherwise, if there is no EC available in *NT*, we insert the tuple into the smallest EC of the *ECs*, which are not considered in *NT* (i.e., Lines 23 and 24 in Algorithm 2). This iteration repeats until at least one nonsatisfied EC remains in *NT*.

If there is only one nonsatisfied EC or one EC remaining in *NT*, each tuple in that class is added to the smallest EC in *ECs* (i.e., Lines 25-30 in Algorithm 2).

B. FILTERED DATA-ANONYMIZATION METHOD

In our data-anonymization method, whether a selected tuple of ΔT^+ or ΔT^- exist in an EC is frequently determined, which is an example of approximate set-membership test. Set-membership tests determine whether a given item belongs to a given set. Because probabilistic data structures pass set-membership tests, improve data-processing efficiency, and compactly store low-memory data, we used such structures to improve data-processing efficiency of our data-anonymization method method.

1) CUCKOO FILTER

A Cuckoo filter [10] is a space-efficient probabilistic data structure used for approximate set-membership tests. Probabilistic data structures are very useful, especially when processing large datasets. Numerous probabilistic data structures are described detail in the literature. Although Standard Bloom filters [26], which support record insertion and lookup operations, are well understood and are the most widely used in practice, they do not support record deletions. Thus, Counting Bloom filters extend standard ones to allow for record deletions. However, Counting Bloom filters requires 4x more space than standard ones in practice.

Cuckoo filters improve upon Bloom filters by supporting dynamic insertion and deletion of dataset records and better record lookup performance. Cuckoo filters also use less space than Bloom filters and can determine whether a given record is definitely not represented in a given dataset or might be represented in the dataset. That is, although negative responses are absolutely certain, positive ones are associated with a small false positive probability (FPP). Previous experimental results [10] have shown that Cuckoo filters outperform conventional data structures in both data-processing time and space requirements. Considering these beneficial properties, we choose a Cuckoo filter for our work.

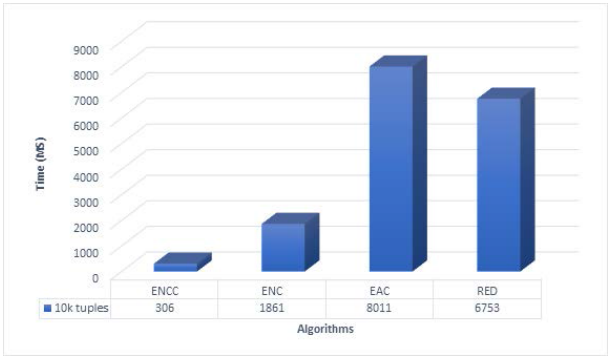
2) APPLICATION OF CUCKOO FILTERS TO OUR DATA-ANONYMIZATION METHOD

We used Cuckoo filters in two ways when a set of tuples was deleted from the original dataset.

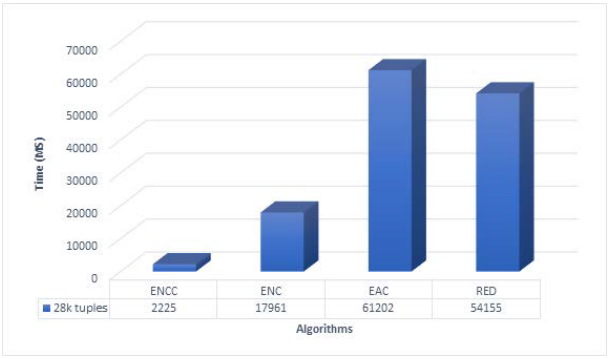
- Before each tuple t is deleted from the ΔT^- dataset, we must first determine its host EC from among all the *ECs* by comparing the current tuple to all the tuples in each EC. Assuming we have filters for each EC, we can simply the search for the current tuple from among all the filters to find the host EC. It meaning that when we determine whether the current tuple is in a given EC, we do not need to compare the tuple to all tuples in each EC; we just search for the tuple from among filters of each EC one at a time.
- After determining nonsatisfied ECs as *NT*, we collect all the tuples of the smallest EC from *NT* and relocate each tuple to a new candidate EC from *NT* that does not contain the SA of the current tuple. To do this, we compare each SA of each tuple to all SAs of all tuples in each EC in *NT*. Assuming we have already filtered SAs in each EC, we can simply search for the SA of the current tuple from among SA filters in each EC to find the candidate EC.

V. EXPERIMENTAL EVALUATION

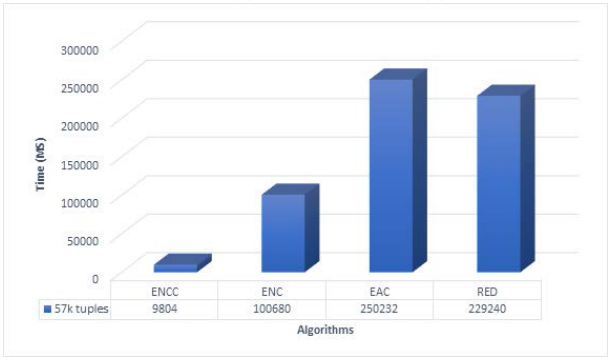
The experiment was designed to investigate record insertion and deletion performances and operation execution (data processing) efficiency of our data-anonymization method. In addition, we assume filters have already created with the initial released original dataset. We reemphasize that our method did not consider information loss



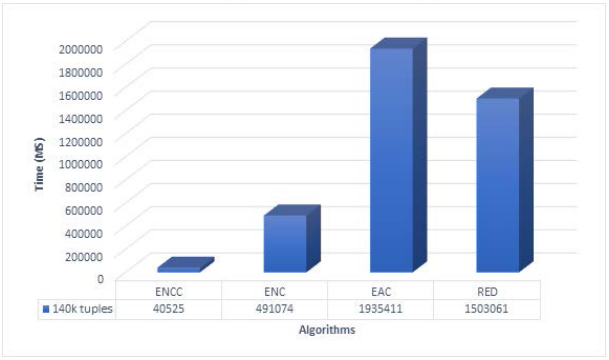
(a) Record deletion (1)



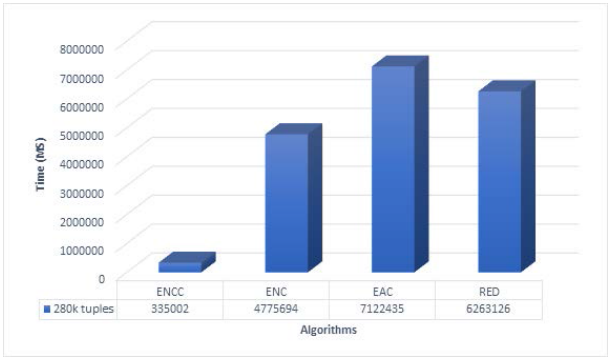
(b) Record deletion (2)



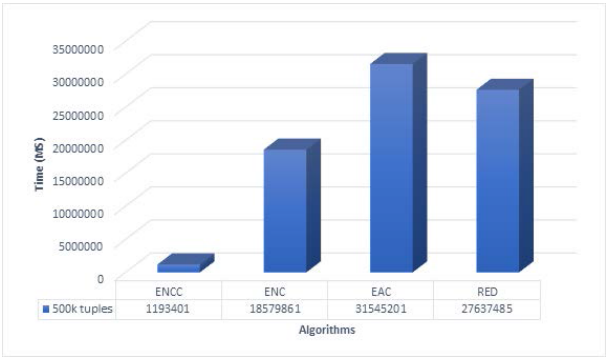
(c) Record deletion (3)



(d) Record deletion (4)



(e) Record deletion (5)



(f) Record deletion (6)

FIGURE 9. Record deletion performance at six different datasets.

because generalization or suppression was not used for data anonymization. Instead, we used anatomy to ensure data privacy.

A. EXPERIMENTAL SETTING

We utilized real census data from the Integrated Public Use Micro-data Series[1] (IPUMS, USA) containing personal information collected from the American population. We selected only distinct tuples by their quasi-identifier attributes and removed all the tuples containing unknown SAs from the dataset. The adjusted dataset consisted of 577 k tuples and 8 attributes. Table 6 summarizes attributes of the data used in the experiments as well as corresponding, type and number of distinct values for each attribute. In all the experiments, we considered Age a numerical quasi-identifier attribute, while Gender, Quarter of Birth, Marital status, Race, Birthplace, and Education were considered categorical quasi-identifier attributes. Furthermore, Occupation was considered the SA. All the experiments were performed using an Intel 2.7 GHz Core-i7 processor and 16 GB of random-access-memory(RAM).

[1]https://usa.ipums.org/usa/

B. RECORD INSERTION PERFORMANCE

We evaluated record insertion performances achieved using anatomy-based reanonymize-entire-dataset (RED) algorithm; the existing reference (EAC) algorithm [20], which examines all the clusters of released datasets and our

TABLE 6. Summary of data attributes.

	Attribute	Number of distinct values	Type of attribute
1	Gender	2	Categorical
2	Age	81	Numeric
3	Quarter of Birth	4	Categorical
4	Marital status	6	Categorical
5	Race	139	Categorical
6	Birthplace	219	Categorical
7	Education	24	Categorical
8	Occupation	479	Sensitive

proposed unfiltered data-anonymization (ENC) algorithm, which only examines nonsatisfied clusters. For record insertion, we did not use a Cuckoo filter with our proposed data-anonymization method. The experiment was conducted on four different volumes of data, reduced by 1, 5, 10, 25% of the entire dataset volume and *l*-diversity 10 was the anonymity requirement. Furthermore, inserted records were 10% of respective original datasets. Fig. 8 shows performance of record insertion executed on the different data volumes, where the *x*-axis represents the data volume, the *y*-axis represents execution time (ms), and chart column represents algorithms used in the experiment. Clearly, our proposed data-anonymization algorithm showed the best performance; RED the worst. Furthermore, because EAC considers information loss by examining all the clusters for each record insertion, it executes more slowly and could become inefficient with increasing volume of inserted data and plural clustered released data. As noted in Section 4, for record insertion, our proposed data-anonymization method does not examine existing clusters (i.e., equivalence classes (ECs)). If the number of distinct tuples for an SA exceeds *l*-diversity of an inserted dataset(ΔT^+), the method generates new ECs and merges the existing ones. Otherwise, each tuple in the inserted dataset (assuming fewer distinct tuples than *l*-diversity) is inserted into an existing EC, which does not require much data-processing time and so is very efficient.

C. RECORD DELETION PERFORMANCE

Herein, we compare record deletion performances achieved using the existing EAC reference algorithm [20], anatomy-based RED algorithm [9], our unfiltered ENC algorithm and our ENC algorithm with an applied Cuckoo filter (ENCC). The algorithms were executed on six different volumes of data. To perform the experiments, we reduced data volume by 1, 5, 10, 25, 50, and 100% containing approximately 10, 28, 57, 144, 288, and 577 k tuples, respectively, as shown in Table 7. To delete records, we randomly selected 10% of the tuples from respective datasets. Following data volume, deleted records were also different sizes. In this experiment, the *l*-diversity anonymity requirement was 10. Fig. 9 shows execution times (ms) of the four record-deletion algorithms obtained for the six different data volumes, where chart column represents the algorithms used, and the *y*-axis represents execution time (ms).

Clearly, execution times of our ENC and ENCC algorithms were dramatically shorter than those of RED and

TABLE 7. Dataset sizes.

Reduced size	1%	5%	10%	25%	50%	100
Number of tuples (k)	10	28	57	144	288	577

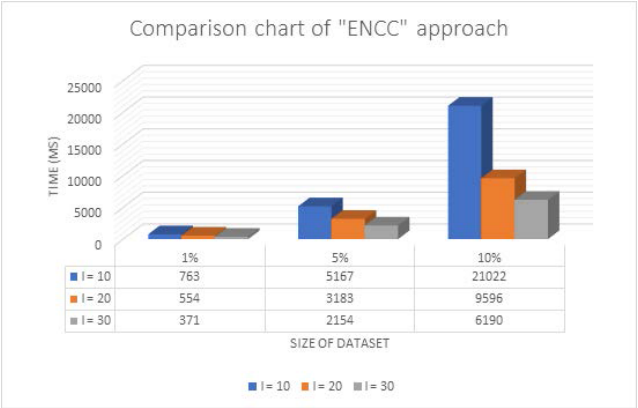


FIGURE 10. Execution times of "ENCC" approach for 3 different *l* value.

EAC reference algorithms. Although the execution time of ENCC algorithm (with the applied Cuckoo filter) includes the updating time of Cuckoo filter, it showed the highest data-processing efficiency. As previously noted in Section 4, probabilistic data structures such as Cuckoo filters can determine whether a given record is definitely not represented in a given dataset or might be represented in the dataset. Although negative responses are absolutely certain, positive ones are associated with a small false-positive probability (FPP). Therefore, there can be slight differences in probability between the ENCC algorithm and other algorithms. Because the anatomy-based RED algorithm [9] was applied, information loss of released anonymized datasets was not considered. As previously mentioned in Section 3, the EAC reference algorithm [20] focused on information loss; therefore, it required the most data-processing time in this experiment because total information loss was calculated in every step of the algorithm.

D. OVERALL ENCC PERFORMANCE

From results of the data-deletion performance experiments, the Cuckoo-filtered (ENCC) showed the best data-processing efficiency. Thus, we compared its data-processing efficiency for different *l*-diversity anonymity requirements. The experiment was conducted on three different volumes of data, reduced by 1, 5, and 10% of the entire dataset and *l*-diversity anonymity requirement of 10, 20, and 30. The results are shown in Fig 10, where chart of column represents *l*-diversity and *y*-axis represents execution time (ms). Clearly, for increasing the *l*-diversity the ENCC algorithm increasingly reduced execution time because, for increasing number of tuples in an EC, the ENCC method executed fewer data comparisons.

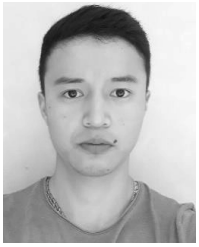
VI. CONCLUSION

Method of preserving privacy of dynamically evolving data sets are currently underdeveloped, especially for stronger privacy preservation requirements of l-diversity and t-closeness, and thus are not significantly addressed in the literature. Herein, we identified limitations of privacy-preservation methods for dynamically evolving datasets. We also developed *l*-diversity data-anonymization algorithm.

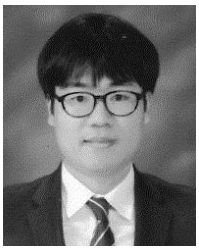
In addition, we implemented a Cuckoo filter, which is a probabilistic data structure that supports dynamic record addition and deletion, in the proposed algorithm to overcome aforementioned limitations and improve data-processing efficiency. Experimental results demonstrated that our proposed data-anonymization algorithm processed data more efficiently than other conventional algorithms. The Cuckoo-filtered algorithm was especially efficient, dramatically reducing operation execution times while maintaining privacy of dynamically evolving datasets. In future studies, we will focus on more stronger data-anonymization models, such as *t*-closeness.

REFERENCES

[1] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.
[2] K. Wang, S. Y. Philip, and S. Chakraborty, "Bottom-up generalization: A data mining solution to privacy protection," in *Proc. ICDM*, vol. 4, 2004, pp. 249–256.
[3] L. Sweeney, "Achieving K-anonymity privacy protection using generalization and suppression," *Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 571–588, 2002.
[4] B. C. Fung, K. Wang, and P. S. Yu, "Top-down specialization for information and privacy preservation," in *Proc. 21st Int. Conf. Data Eng. (ICDE)*, 2005, pp. 205–216.
[5] R. J. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *Proc. 21st Int. Conf. Data Eng. (ICDE)*, 2005, pp. 217–228.
[6] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain k-anonymity," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2005, pp. 49–60.
[7] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *Proc. ICDE*, vol. 6, 2006, p. 25.
[8] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," in *Proc. 22nd Int. Conf. Data Eng. (ICDE)*, 2006, p. 24.
[9] X. Xiao and Y. Tao, "Anatomy: Simple and effective privacy preservation," in *Proc. 32nd Int. Conf. Very Large Data Bases VLDB Endowment*, 2006, pp. 139–150.
[10] B. Fan, D. G. Andersen, M. Kaminsky, and M. D. Mitzenmacher, "Cuckoo filter: Practically better than bloom," in *Proc. 10th ACM Int. Conf. Emerg. Netw. Exp. Technol.*, 2014, pp. 75–88.
[11] Y. Sowmya and M. Nagaratna, "Parallelizing k-anonymity algorithm for privacy preserving knowledge discovery from big data," *Int. J. Appl. Eng. Res.*, vol. 11, no. 2, pp. 1314–1321, 2016.
[12] U. Sopaoglu and O. Abul, "A top-down k-anonymization implementation for apache spark," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 4513–4521.
[13] F. Radulovic, R. G. Castro, and A. Gómez-Pérez, "Towards the anonymisation of RDF data," in *Proc. 27th Int. Conf. Softw. Eng. Knowl. Eng.*, 2015, pp. 646–651.
[14] O. Temuujin, M. Jeon, K. Seo, J. Ahn, and D.-H. Im, "Spark-based partitioning algorithm for k-anonymization of large RDFs," in *Proc. FutureTech Conf.*, 2019, pp. 292–298.
[15] E. Elabd, H. Abdulkader, and A. Mubark, "L–diversity-based semantic anonymaztion for data publishing," in *Proc. IJ Inf. Technol. Comput. Sci. (IJITCS)*, vol. 10, 2015, pp. 1–7.
[16] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *Proc. IEEE 23rd Int. Conf. Data Eng.*, Apr. 2007, pp. 106–115.
[17] J. Salas and V. Torra, "A general algorithm for k-anonymity on dynamic databases," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Berlin, Germany: Springer, 2018, pp. 407–414.
[18] P. Wang and J. Wang, "L-diversity algorithm for incremental data release," *Appl. Math. Inf. Sci.*, vol. 7, no. 5, p. 2055, 2013.
[19] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li, "Secure anonymization for incremental datasets," in *Proc. Workshop Secure Data Manage.* Berlin, Germany: Springer, 2006, pp. 48–63.
[20] X. Sun, H. Wang, and J. Li, "*L*-diversity based dynamic update for large time-evolving microdata," in *Proc. Australas. Joint Conf. Artif. Intell.* Berlin, Germany: Springer, 2008, pp. 461–469.
[21] X. Xiao and Y. Tao, "M-invariance: Towards privacy preserving re-publication of dynamic datasets," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2007, pp. 689–700.
[22] Q. Wei, Y.-S. Lu, and L. Zou, "ε-inclusion: Privacy preserving re-publication of dynamic datasets," *J. Zhejiang Univ.-Sci. A*, vol. 9, no. 8, pp. 1124–1133, 2008.
[23] S. Kim, M. K. Sung, and Y. D. Chung, "A framework to preserve the privacy of electronic health data streams," *J. Biomed. Inf.*, vol. 50, pp. 95–106, Aug. 2014.
[24] J. Cao, P. Karras, P. Kalnis, and K.-L. Tan, "SABRE: A sensitive attribute bucketization and redistribution framework for t-closeness," *VLDB J.*, vol. 20, no. 1, pp. 59–81, 2011.
[25] J.-W. Byun, A. Kamra, E. Bertino, and N. Li, "Efficient *k*-anonymization using clustering techniques," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Berlin, Germany: Springer, 2007, pp. 188–200.
[26] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.



ODSUREN TEMUUJIN received the B.S. degree in information technology education from the National University of Mongolia, in 2016. He is currently pursuing the M.S. degree with the Department of Computer and Information Engineering, Hoseo University, South Korea. His research interests include data privacy, big data processing, the Internet of Things, and mobile development.



JINHYUN AHN received the B.S. and M.S. degrees in computer science education from Korea University, in 2005 and 2007, respectively, and the Ph.D. degree in computer science and engineering from Seoul National University, in 2017. He is currently an Assistant Professor with the Department of Management Information Systems, Jeju National University, South Korea. His research interests include management information systems, graph processing, data access control, and distributed algorithms.

DONG-HYUK IM received the B.S. degree in computer science education from Korea University, in 2003, and the M.S. and Ph.D. degrees from the School of Computer Science and Engineering, Seoul National University, in 2005 and 2011, respectively. He is currently an Associate Professor with the Department of Computer and Information Engineering, Hoseo University, South Korea. His research interests include database systems, data privacy, and big data processing.