

(τ, m) -slicedBucket privacy model for sequential anonymization for improving privacy and utility

Razaullah Khan¹, Xiaofeng Tao^{1*}, Adeel Anjum², Saif ur Rehman Malik³, Shui Yu⁴, Abid Khan⁵, Waheed ur Rahman⁶, Hassan Malik⁷

¹National Engineering Laboratory for Mobile Network Technologies, Beijing University of Posts and Telecommunications, Beijing, China,

²Department of Computer Science, COMSATS University Islamabad, 45550, Islamabad, Pakistan,

³Cybernetica AS Estonia, Tallinn, Estonia,

⁴Faculty of Engineering and Information Technology, School of Software, University of Technology Sydney, Australia,

⁵Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, UK,

⁶Department of Computer Science, University of Peshawar, Peshawar, Pakistan,

⁷Department of Computer Science, Edge Hill University, Ormskirk, UK.

*Corresponding author: taoxf@bupt.edu.cn

Abstract: In a real-world scenario for privacy-preserving data publishing, the original data is anonymized and released periodically. Each release may vary in number of records due to insert, update, and delete operations. An intruder can combine i.e. correlate different releases to compromise the privacy of the individual records. Most of the literature, such as τ -safety, τ -safe (l, k) -diversity, have an inconsistency in record signatures and adds counterfeit tuples with high generalization that causes privacy breach and information loss. In this paper, we propose an improved privacy model (τ, m) -slicedBucket, having a novel idea of “Cache” table to address these limitations. We indicate that a *collusion attack* can be performed for breaching the privacy of τ -safe (l, k) -diversity privacy model, and demonstrate it through formal modeling. The objective of the proposed (τ, m) -slicedBucket privacy model is to set a tradeoff between strong privacy and enhanced utility. Furthermore, we formally model and analyze the proposed model to show that the *collusion attack* is no longer applicable. Extensive experiments reveal that the proposed approach outperforms the existing models.

Keywords Big Data, Electronic Health Records, k -anonymity, Privacy Preserving Data Publishing, Privacy, Security.

1. Introduction

With the advent of recent technologies such as Internet of Things (IoT) and Big Data technologies, huge amount of data is collected on daily basis [1-2]. The accumulated data can be of various type such as credit card transactions, phone calls, web browsing, social media activities, or Electronic Health Records (EHR), etc. [1-3]. Such data may contain private information e.g. name, address, account number, patient disease information in EHR, etc. The leakage of private information is a major concern in today’s modern society. Recently, it has been reported that 41 million healthcare record breached in 2019 [4]. Moreover, another report indicates that U.S. healthcare department loss \$6.2 billion annually due to the private information leakage in EHR [5]. On the other hand, data publishing and sharing is also very crucial for research innovation and improving business processes. In this regard, Privacy-Preserving Data Publishing (PPDP) methods aims to keep the privacy of an individual before publishing the data using anonymization techniques [6-8].

The PPDP methods can be either static or dynamic. In case of Privacy-Preserving Static Data Publishing (PPSDP), the data is published one time. The most common PPSDP techniques include k -anonymity [6-7], l -diversity [8], t -closeness [9]. Contrary, Privacy-Preserving Dynamic Data Publishing (PPDDP) is based on republication in which data can be modified over time through insertion, update or deletion. The first contribution introduced in PPDDP is known as m -invariance [10], where the signature (a set of sensitive information in a group of records), of each record must remain the same in every republication. The authors in [11] and [12] improved m -invariance with τ -safety and τ -safe (l, k) -diversity privacy models, respectively. In all the three models dummy records (noise tuples) have been used as counterfeit tuples to create a consistent signature for a record. However, it has been identified that τ -safe (l, k) -diversity [12] prevents signature inconsistency during internal updates up to two releases and cannot prevent during further alternate releases. Moreover, all the mentioned techniques suffer from counterfeit usage limitations.

The focus of this paper is to address the above-mentioned limitation in existing PPDDP techniques by proposing a new privacy model named as (τ, m) -slicedBucket, in comparison to τ -safe (l, k) -diversity model. The proposed model identifies that the τ -safe (l, k) -diversity model contains inconsistent record signatures in an alternate releases. The inconsistent record signatures prone the patient's data to a new type of attack, that is termed as *collusion attack* which breaches the privacy of the intended target individual. The proposed algorithm creates a consistent record signature for each record over time to prevent the collusion attack. The proposed approach presents a novel idea of cache table (i.e. Cach), which are few repeated records extracted from an original microdata T to avoid use of counterfeit records. The proposed Algorithm 1 (see Section 5.4) begins working after Cach table creation. To the best of our knowledge the idea of Cach table is a novel work in PPDDP. The (τ, m) -slicedBucket also separate the delete table (Del) work in τ -safe (l, k) -diversity model into Del and an update (Upd) tables for better record identification. During the anonymization process, we apply slicing [13] and cell generalization [14-15] approaches for an improved utility and privacy. The proposed (τ, m) -slicedBucket privacy model groups the tuples into sliced buckets. Each slice bucket must fulfill the following privacy requirements; (i) be of size k and m -invariant, (ii) be in sliced form, (iii) fulfill the m -invariance requirement and (iv) maintain a consistent signature for an individual record in each release during internal or external updates (discussed in Section 1.1 Motivation).

1.1 Motivation

The microdata data T contains EHRs of different individuals. The attributes of T are classified into; identifier attribute - A^{ID} (e.g. social security number or an individual name), quasi identifier attributes - A^{QI} (e.g. age, gender, zipcode), and sensitive attribute - A^S (e.g. disease.) For data de-identification, removing the A^{ID} is not enough because an adversary (i.e. an attacker) can use background knowledge (BK) i.e. certain pattern of A^{QI} (partial identifiers) and A^S [16], and can link the published data to some external data (e.g. voting system) to re-identify an individual, named as linking attack. Therefore, the original microdata T is k -anonymized [1] to T^* (e.g. Table 2a) for preventing the linking attack. Before anonymizing the actual data using the proposed (τ, m) -slicedBucket privacy model, the repeated A^S records in microdata T (Table 1a) are extracted to produce Cach table (Table 1b). The remaining records (Table 1c) are considered as original microdata T for anonymization. Therefore, to identify the privacy breach in [12] the same data i.e. in Table 1c, is used to perform anonymization through τ -safe (l, k) -diversity [12] and the proposed (τ, m) -slicedBucked privacy models.

TABLE 1a Microdata Table T (before Cach)

Age	Gender	Zipcode	Disease
22	M	47906	Aids
22	F	47305	Flu
33	F	47905	SSM-pos
52	F	47905	Asthma
54	M	47906	Flu
60	M	47302	Cardiac
60	M	47304	Dyspepsia
64	F	47304	Gastritis
21	M	47901	Bronchitis
54	F	47902	Dyspepsia
27	M	47303	Aids
65	M	47308	Cardiac
58	F	47905	Asthma
64	F	47308	Gastritis
21	M	47907	Bronchitis

TABLE 1b Cach table

Age	Gender	Zipcode	Disease
22	M	47906	Aids
22	F	47305	Flu
33	F	47905	SSM-pos
52	F	47905	Asthma
60	M	47302	Cardiac
60	M	47304	Dyspepsia
64	F	47304	Gastritis
21	M	47901	Bronchitis
52	F	47907	Malaria
32	M	47955	Diarrhea
52	M	47915	Pneumonia

TABLE 1c Microdata T (after Cach)

Age	Gender	Zipcode	Disease
21	M	47907	Bronchitis
27	M	47303	Aids
33	F	47905	SSM-pos
58	F	47905	Asthma
54	M	47906	Flu
54	F	47902	Dyspepsia
64	F	47308	Gastritis
65	M	47308	Cardiac

The PPDDP can support records insertion, deletion or update operations during different releases (i.e. $T_1^*, T_2^*, T_3^*, \dots, T_n^*$). Modification in data is known as external and internal updates. First time insertion or deletion that effects the total number of records; are known as external updates. Internal updates are the re-insertion of deleted records or change in the record attribute values i.e. A^{QI} or A^S values, over different times. In this paper we consider the internal updates for A^S values and are arbitrary i.e. old values have not been correlated with the new ones. An intruder can identify an individual record by correlating different releases i.e. intersection or subtraction, published over time. Like [16], we consider that some of the A^S values are persistent (values that never change) while others are transient (may freely change with time) but are arbitrary. Figure 1 shows the relationship between persistent and transient values with the possible operations on them. For example, if insertion is performed initially, the value must be persistent, but it can be transient which will be known in T_j ($2 \leq j \leq n$). An update operation shows that the value must be transient, and so on.

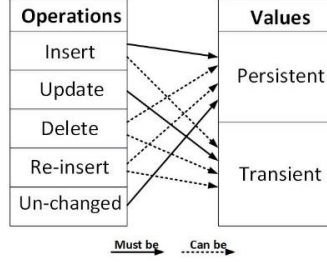


FIGURE 1 Relating values with the possible operations on them

Consider a portion from original microdata T (after Cach reduction) in Table 1c having A^S values that are common between male and female.

TABLE 2a τ -safe (2,2)-diversity GT1 for T_1^*

GID (name)	Age	Zip code	BID
1(p1)	[21-27](21)	[47303-47907](47907)	1
1(p2)	[21-27](27)	[47303-47907](47303)	1
2(p3)	[33-58](33)	[47905-47906](47905)	2
2(p4)	[33-58](58)	[47905-47906](47905)	2

TABLE 3a τ -safe (2,2)-diversity GT2 for T_3^*

GID (name)	Age	Zip code	BID
1(p1)	[21-22](21)	[47906-47907](47906)	1
1(C1)	[21-22](22)	[47906-47907](47907)	1
2(p3)	[33-58](33)	[47905-47921](47905)	2
2(p5)	[33-58](58)	[47905-47921](47921)	2

TABLE 2b τ -safe (2,2)-diversity BT1 for T_1^*

BID	Signature	Count
1	Bron, Aids	1
2	SSM-pos, Asthma	1

TABLE 2c Del table at time 1

p2	Bron, Aids
p4	SSM-pos, Asthma
p3	SSM-pos, Asthma

TABLE 3b τ -safe (2,2)-diversity BT2 for T_3^*

BID	Signature	Count
1	Bron, Aids	1
2	SSM-pos, Dysp	1

TABLE 3c Del table at time 3

p2	Bron, Aids
p4	SSM-pos, Asthma
p3	SSM-pos, Asthma
p3	SSM-pos, Dysp

This work has been motivated by the following limitations in τ -safe (l, k)-diversity privacy model [12].

(i) **Fails to prevent Collusion Attack.** The τ -safe (l, k)-diversity [12] prevents signature inconsistency during internal updates only for two releases and cannot prevent during further alternate releases i.e. $T_n^* \neq T_{n-2}^*$, $T_{n-1}^* \neq T_{n-3}^*$ and so on. For example, after releasing T_1^* the deleted records = {p2, p4}, updated = {p3}. T_2^* creates consistent signatures with T_1^* but the problem arises in T_3^* . Consider an internally update scenario for any of the record e.g. p3=SSM-pos in Table 2a. At time 2 in T_2^* , for example p3 has suffered from *Diarrhea*, so a new signature {*Glaucoma, Diarrhea*} has been created. At time 3, if p3 suffers again with same disease value *SSM-pos*, its intersection with {*Glaucoma, Diarrhea*} is zero, i.e. $S(t) \notin \text{Sig}(\text{BT}_{p-1}(t))$ in τ -safe (l, k)-diversity algorithm, and $T_3^* \cap T_2^* = 0$. The reason behind is; the algorithm checks the signature inconsistency with its previous release only i.e. p-1, and no further checking is performed. The τ -safe (l, k)-diversity algorithm allows to create a new signature for p3 that may causes signature inconsistency with the same p3 record at time 1. The signatures stored in Del Table 3c are useless at all because the algorithm only stores record signatures in Del table. While creating new signatures, there is no verification of the same signature existence in Del table. It also causes the increase in the size of Del table drastically. For example, at time 3, the newly created signature for p3 is (*SSM-pos, Dyspepsia*), where p5 = *Dyspepsia* is a newly inserted sensitive value at time 3. The new signature for p3 has an intersection value *SSM-pos* with his own signature at time 1. The intersection value obtained is a collusion of the three releases. Such internal updates enable the intruder in identifying the p3 latest sensitive value using the A^Q values in previously published releases. Therefore, the collusion attack can occur.

(ii) **Improper use of Del table.** Two limitations were observed in Del table usage of τ -safe (l, k)-diversity [12], (i) both the deleted and updated record-signatures are stored at one place, (ii) no-deletion of re-inserted record signature. These limitations cause two problems: (i) Mixing of updated and deleted tuples; that eventually leads to a scenario where the legitimate records are unpublishable. For a specific record, each time the alternate internal updates create new signatures that will lead to signature limitations, ii) Continuously increasing size of Del table. Therefore, searching for an internally updated record signature among the deleted records, will take longer time. The proposed (τ, m)-slicedBucket privacy model separates the Del work into Del and Upd tables.

(iii) **Bound to use Counterfeit tuples.** The use of counterfeit; a noise or dummy record, is common in m -invariance [10], τ -safety [11] and τ -safe (l, k)-diversity [12]. These models directly use the counterfeit tuples when there is no other record to create the required signature. Using counterfeits, increases the privacy but reduces the truthfulness in records and utility because of the noise addition. In a special case, there may be insertion of such records that will only need the counterfeits for signature consistency, so a drastic utility decrease with complete fake publishing may occur. The proposed privacy model in this work, does not use any counterfeit tuple with the help of novel idea of Cach table.

1.2 Contributions

The main contributions of the proposed (τ, m) -slicedBucket privacy model are as follows.

1. We propose an improvement of τ -safe (l, k) -diversity, named as (τ, m) -slicedBucket privacy model for sequential dynamic data publishing. The proposed approach prevents against a new type of attack, named collusion attack. Unlike the τ -safe (l, k) -diversity, the (τ, m) -slicedBucket uses Del, Upd and Cach tables during the anonymization process to help in creating consistent record signatures.
2. We formally model and investigate the invalidation of τ -safe (l, k) -diversity for the collusion attack and correctness of the proposed (τ, m) -slicedBucket privacy model.
3. Based on the above points, the simulation results prove that the proposed privacy model outperform its counterpart in terms of privacy and utility.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 discusses the preliminaries. Section 4 shows the considered attacks and limitations in τ -safe (l, k) -diversity [12] with respect to HLPN. Section 5 present the proposed (τ, m) -slicedBucket privacy model and its formal analysis using HLPN. In Section 6, experiments and evaluations have been discussed. Section 7 concludes the paper.

2. Related Work

This section investigates and categorize the available relevant research work in order to define narrow scope of the proposed work.

For implementing privacy and security in IoT and Big Data, numerous techniques exist in literature [17-26]. For PPDP, a bunch of anonymization algorithms have also been proposed so far. These PPDP approaches can be broadly categorized to semantic and syntactic privacy models. The ϵ -differential privacy (random noise addition) [19] is an example of semantic data privacy. While syntactic privacy is a clustering framework which creates quasi identifier (QI) groups called equivalence classes (ECs). The k -anonymity [6-7] by Sweeney et. al., and all its refinements e.g. l -diversity [8], t -closeness [9], β -likeness [20], θ -Sensitive k -anonymity [21], are the syntactic approaches which prevent the linking attack. In all these privacy approaches the microdata is generalized into k -anonymized groups where every record is un-differentiable from at least $k-1$ other records. The syntactic privacy algorithms can be further categorization into microaggregation [22-26], anatomization [27-28] and generalization [29-35]. Microaggregation replaces the QI values in an EC to the centroid of the EC. Anatomization performs vertical partition by separating the microdata T into QI attributes table and sensitive attribute (SA) table. Generalization includes top-down [34] or bottom-up [35] approaches where more specific QI values are replaced with less specific values. As compared to generalization, anatomy [27] may have high degree of privacy disclosure because of publishing actual values. The centroid values in microaggregation are not the real record QI values. Hence the published data is meaningless. Due to the advantages of generalization; as compared to microaggregation and anatomization, we have used bottom-up cell generalization in the proposed algorithm.

A more practical and challenging scenario is the PPDDP which can be categorized to (i) Multiple and, (ii) Sequential data publishing. In multiple data publishing, at the same time and on the same data, different set of attributes are published [14], [36]. In sequential data publishing, many releases of the same table are published over a period of time [10-12], [15-16], [37-39]. The focus of this paper is sequential data publishing, where data are published over time having the same schema. In each release the number of records may vary. The variation during different releases is due to adding records (insertion), removing few existing records (deletion), or modifying the existing records or re-insertion of old records (updating). Most privacy models fail to preserve privacy due to adversary BK. Several works consider BK for single publishing [40-42], however an adversary cannot adopt the static BK for re-publishing scenario. For privacy breach in re-publishing the adversary correlate attributes or records in different sequential releases.

Byun et. al. [43] was first to propose the idea of data re-publishing where only insertion is considered (incremental update). The m -invariance [10] being the first to address dynamic publications; all the operations i.e. insertion, updating and deletion, is considered in a sequential scenario. A top-down global generalizing algorithm; k -likability [38] confirmed that a record could not be linked with less than k distinct A^S values in a sequential release. The applicability of the algorithm was only for two releases. Shmueli et al. [37] proposed a sequential privacy model in a multipartite graph with more releases. In the above model, only few records could be inserted over time. τ -safety [11] is a state-of-the-art privacy model for sequential data publication based on m -invariance [10] concept. τ -safe (l, k) -diversity [12] further claimed to improve the τ -safety model however, the algorithm failed to reflect the claim and applicability is limited to only two releases. Also, in m -invariance [10], τ -safety [11] and τ -safe (l, k) -diversity [12], adding counterfeit tuples is compulsory which fails to claim the truthfulness of published data. The proposed privacy

model (τ, m) -slicedBucket is a syntactic sequential data anonymization privacy model without the counterfeit tuples, using cell generalization approach.

3. Preliminaries

Let the microdata $T = \{T_1, T_2, T_3, \dots, T_r\}$ be the EHRs tables generated at times 1,2,3,...r, respectively, where each table T_j before anonymization is in the form: $T_j = \{A^{ID}, A^{QI}, A^S\}$. The total number of tuples in any T_j are t_d ($1 \leq i \leq d$) tuples where each tuple belong to an individual i , and is known as the record respondent. Let T_j^* be the anonymized release, that have been published at time j ($1 \leq j \leq r$). The T_j^* consists of QI attributes column (AC^{QI}) and A^S column (AC^S), where A^{ID} (i.e. work as a patient identifier - PID) is dropped before publication. Del, Upd and Cach are the supporting tables, which store the deleted records with their signatures, internally updated records with their signatures and the repeated A^S values tuples from original microdata, respectively. The intruder identifies an individual after linking the A^{QI} s with some external dataset e.g. the publicly available voting data. The A^S , which contains sensitive information e.g. disease in our case, needs to be protected the most. Table 4 depicts the notations used in this paper.

TABLE 4 Notation used in the paper

Symbol	Description	Symbol	Description
T_j	Table of m tuples of v individuals	Sign	Sensitive Attributes Signatures
T^s	Number of tuples having same attribute values in each release	Del	Table of deleted records.
T^n	Newly inserted records	GID	Group Identifier
T^r	The re-inserted tuples from Del table	BK^{sv}	Sensitive value background knowledge
T_{r-1}	The original microdata table released at time $r - 1$	UBK^{sv}	Sensitive value updated BK^{sv}
T_r	The current original micro data table to be released at time r	PK^{sv}	Sensitive value posterior knowledge
T^u	The tuples whose sensitive value updates between the releases	CBK	Composite Background knowledge
A^{ID}	Explicit identifiers in T	T_j^*	Anonymized data at time j
AC^{QI}	Quasi identifier column in T	BT	Bucket Table
AC^S	Sensitive attributes column in T	GT	Generalized Table

Tuples t_i in table T_j are classified into T^s , T^n , T^r , T^u and T^d as follows.

$$T^s = T_r \cap T_{r-1}$$

$$T^n = (T_r - T_{r-1}) + T^u, t \notin \text{Del}, t \notin \text{Cach}$$

$$T^r: \forall t \in T_r, t \notin T_{r-1}, t \in \text{Del}, t \notin \text{Upd}$$

$$T^u: \forall t \in T_r, t \notin T_{r-1}, t \in \text{Upd}, t \notin \text{Del}$$

$$T^d: \forall t \notin T_r, t \in T_{r-1}, t \in \text{Del}$$

where T_r and T_{r-1} are the original microdata tables at time r and $r-1$, t is a record of an individual.

3.1. Adversarial Background Knowledge

The background knowledge (BK); logical or probabilistic, is the information an adversary collects from different sources and personal observations that may cause a privacy breach in static [7-9], [21-22] or in re-publication of data [11], [16], [32], [44]. In the proposed privacy model, it is assumed that an adversary recursively updates the BK. Adversarial BK comprises of already published releases, history of each individual tuple, QI values and an updated knowledge obtained from joining different releases.

Sensitive value background knowledge (BK^{sv}) is the adversary's initial belief about a record respondent and about the corresponding possible sensitive value. It can be referred as prior belief about the sensitive values. Sensitive value updated background knowledge (UBK^{sv}) is the revised observation over the released tuples BK^{sv} during different releases. After a specific release, sensitive value posterior knowledge (PK_i^{sv}) is the adversary confidence about a possible sensitive value for a record respondent. After release at time 1, only BK_i^{sv} derives PK_i^{sv} , because during first release, UBK^{sv} is not available. In static data publishing, PK_i^{sv} is compared with prior BK_i^{sv} for privacy disclosure. In dynamic re-publication, continuous background knowledge (BK^{con}) is the correlation between two sensitive values at time 1 and 2. As, we are considering only arbitrary updates, this BK^{con} may not help the adversary. However PK_i^{sv} is joined with BK^{con} , which is further joined with BK_i^{sv} to derive UBK_{i+1}^{sv} after release, at time 2. The UBK_{i+1}^{sv} is then joined with BK_i^{sv} to derive PK_{i+1}^{sv} for release at time 2. This recursive joining process increases the adversarial knowledge continuously, which helps in breaching the privacy of an individual. Equation (1) and (2) depict PK_i^{sv} and UBK_{i+1}^{sv} calculation for some anonymized releases at time i .

$$PK_i^{sv} = BK_i^{sv} \bowtie UBK_{i+1}^{sv} \quad (1)$$

$$UBK_{i+1}^{sv} = PK_i^{sv} \bowtie BK^{con} \bowtie BK_i^{sv} \quad (2)$$

In this work, we consider collusion attack and membership attack based on intruder's BK. The m -invariance [10], τ -safety [11] and τ -safe (l, k) -diversity [12], instead of their claims are lacking with the inconsistencies of individual record signatures that causes the collusion attacks possible. For example in m -invariance [10] definition condition 2 "for any tuple t with lifespan $[x, y]$ have the same signature", τ -safety [11] definition condition 2, "signature of $[x]$ must remain the same" and τ -safe (l, k) -diversity [12] condition 2, "all signatures of the record t must be consistence and have no intersection in the lifetime" are the same but have different explanations. What they claim in their definitions, are not achieved in their algorithms. The adversary continuously uses PK_i^{sv} and UBK_i^{sv} in Equation 1 and 2 respectively for privacy breaches that leads to collusion attacks and membership disclosure attacks.

3.2 Adversarial Model

We assume the following adversarial model:

- The generalized table $GT = \{GID, A^{QI}, BID\}$.
- The bucket table $BT = \{BID, Sign, Count\}$.
- A published dataset $PD = \{GT, BT\}$ that is publicly available.
- The adversarial composite background knowledge is of the form: $CBK = (UBK_i^{sv}, PK_i^{sv}, PD)$

Definition 1: Collusion Attack. The adversary performs collusion attack, if $Sig(T_r^*(t_j)) \cap Sig(T_{r-i}^*(t_j)) \neq \emptyset$, where $2 < i < n$, which can be a tuple from T^u or from T^r that can eventually identify the individual.

Definition 2: Membership Attack. The adversary performs membership attack if he can map the known QI attribute of an individual i to an EC, to identify the A^S value with the help of available PK_i^{sv} and UBK_i^{sv} .

Definition 3: External Update [11-12]. $\forall t$, the operation is said to be external update of t , if $t \in T^n$ or $t \in T^d$.

Definition 4: Internal Updates [11-12]. $\forall t$, the operation is said to be internal update of t , if $t \in T^u$ or $t \in T^r$.

Definition 5: slicedBucket (SBUC). Partitioning correlated attributes into columns and, tuples into buckets makes SBUCs. The values inside the SBUC can be randomly permuted to break the correlation between A^{QI} and A^S .

Definition 6: Signature [11-12]. Let SBUC be a sliced set of distinct sensitive values in an EC. Signature of a tuple i.e. $Sig(t)$ is known by the distinct sensitive values in $SBUC_i$ from which tuple t belongs.

Definition 7: m -unique [10]. A $SBUC_i (1 \leq i \leq n)$ is m -unique if it contains at least m tuples and all the tuples have distinctive A^S values. An anonymized table T_r^* is m -unique if all the SBUCs are m -unique.

Definition 8: m -invariance [10]. Let $T_1^*, T_2^*, T_3^*, \dots, T_r^*$ be the sequential anonymized published relations, are said to be m -invariant if:

1. $\forall T_i^*$ where $(1 \leq i \leq r)$ is m -unique.
2. For any tuple $t \in T$ with lifespan $[j, j+k] (1 \leq j \leq r), k \geq 0$, we have $Sig(t_j) = Sig(t_{j+1}) = \dots = Sig(t_{j+k})$, where t_j denotes tuple t at publication time j and $Sig(t_j)$ shows signature of tuple t at publication time j .

Definition 9: Privacy Risk [12]. $risk(t) = p(t | (PK_i^{sv} \wedge UBK_{i+1}^{sv}))$ where $p(t)$ is the probability of tuple t and PK_i^{sv} and UBK_{i+1}^{sv} are obtained from Eq. 1 and Eq. 2.

Definition 10. High Level Petri-Nets (HLPN) [45-46]. A HLPN represents the system in a graphical and mathematical way to examine the control of information. It consists of 7-tuple, $N = (P, T, F, \varphi, R_n, L, M_0)$. P represents set of all places where place is a single portion in the system represented by circle, T is the set of transitions such that $P \cap T = \emptyset, P \cup T \neq \emptyset$. F shows the flow such that $F \subseteq (P \times T) \cup (T \cup P)$, φ maps P to the data types. R_n defines the rules for transitions, L is a label on F and M_0 represents the initial marking [37]. In short, L, φ and R_n shows the static semantic whereas F, P and T represents the dynamic structure.

4. Formal modeling and analysis of τ -safe (l, k) -diversity privacy model with adversarial attack identification

In this section, we formally model and analyze the τ -safe (l, k) -diversity [12] algorithm to identify the adversarial attack, i.e. collusion attack. To perform the formal modelling and analysis, HLPN has been used to represent the τ -safe (l, k) -diversity model in terms of its mathematical properties. Descriptions of places and variable types are shown in Table 5a. Mapping of data types on places are in Table 5b used in HLPN Figure 2.

Figure 2 comprises of three entities, namely: end user, trusted data sanitizer and adversary. The transitions have been referred as Input. The first Input transition receives data from end user that serves for some health organization, EHRs; patients' records - raw data. After this stage, the data is transferred to a trusted data-sanitizer. The data sanitizer

Types	Description
T_{p-1}	Place holding sets of same records, signatures, and re-inserted records.
GID	An integer type describing group identifier
PID	Patient Identifier
BID	An integer type describing bucket identifier.
BT_{p-1}	Place holding bucket table and signatures
B	Place holding records signatures during classification phase
B'	Place holding SAs new signatures during classification phase
BUC, BUC'	Holds records signatures before and after updating all signatures
$T^n, T^{n'}$	Holding sets of new records and new records with added counterfeits
T^{CIS}, T^{CtQ}	Place holding counterfeit records SV only and QI values only.
BA-BUC	Holding combined records signature after balancing and assignment
asg-var	Place holding assignment variables: α is the number of buckets in T^n , β is the number of distinct sensitive values in bucket and γ is total number of distinct sensitive values in T^n .
A-BUC	Place holding updated bucket after assignment
T_p	Place holding QI values with BID from BA-BUC at release p
T-Q	Holds BID, quasi identifiers and generalization variables
T_{sort}	Holds sorted QIs records in the data set
T_{sortv}	Holds the same T_{sort} data for generalization after condition
array	Holds top k individuals QIs in the data set after condition
GenArrData	Holds array data after generalization
GenTData	Holds T data after generalization
GenData	Holds final generalized QI data, obtained
Sign Dis	Holds adversarial disclosed signatures

TABLE 5b Mapping of data types on places

Places	Description
$\varphi(T_{p-1})$	$\mathbb{P}(T^s \times \text{Sign}_{\text{same}} \times T^{re})$
$\varphi(BT_{p-1})$	$\mathbb{P}(GID \times \text{Sign}_{p-1} \times BID)$
$\varphi(B)$	$\mathbb{P}(GID \times \text{Sign}_c \times BID)$
$\varphi(B')$	$\mathbb{P}(GID \times \text{Sign}_{ru} \times BID)$
$\varphi(DEL)$	$\mathbb{P}(PID \times \text{Sign}_i \times BID)$
$\varphi(BUC')$	$\mathbb{P}(GID \times \text{Sign}_{all} \times BID)$
$\varphi(BUC)$	$\mathbb{P}(GID \times \text{Sign}_m \times BID)$
$\varphi(T^n)$	$\mathbb{P}(PID \times \text{Sign}_{new} \times BID \times \ell)$
$\varphi(T^{CIS})$	$\mathbb{P}(GID \times \text{Sign}_{ct} \times BID)$
$\varphi(BA-BUC)$	$\mathbb{P}(GID \times \text{Sign}_{BA} \times BID)$
$\varphi(T^{n'})$	$\mathbb{P}(GID \times \text{Sign}_{new'} \times BID \times T_{ct})$
$\varphi(A-BUC)$	$\mathbb{P}(GID \times \text{Sign}_a \times BID)$
$\varphi(T_p)$	$\mathbb{P}(GID \times QI \times BID)$
$\varphi(T^{CtQ})$	$\mathbb{P}(GID \times QI \times BID)$
$\varphi(T-Q)$	$\mathbb{P}(GID \times QI \times BID \times \lambda \times k)$
$\varphi(\text{asg-var})$	$\mathbb{P}(\alpha \times \beta \times \gamma)$
$\varphi(\text{array})$	$\mathbb{P}(GID \times QI_{\text{sort}} \times BID \times \gamma)$
$\varphi(\text{GenArrData})$	$\mathbb{P}(GID \times QI_{\text{array}} \times BID)$
$\varphi(T_{sort})$	$\mathbb{P}(GID \times QI_{\text{sort}} \times BID \times \gamma)$
$\varphi(T_{sortv})$	$\mathbb{P}(GID \times GQI \times BID)$
$\varphi(\text{GenTData})$	$\mathbb{P}(GID \times QI_T \times BID)$
$\varphi(\text{GenData})$	$\mathbb{P}(GID \times QI_T \times BID)$
$\varphi(\text{UBK})$	$\mathbb{P}(QI_{\text{adv}} \times \text{Sign}_{\text{adv}})$
$\varphi(\text{Sign Dis})$	$\mathbb{P}(QI \times \text{Sign}_{\text{dis}})$

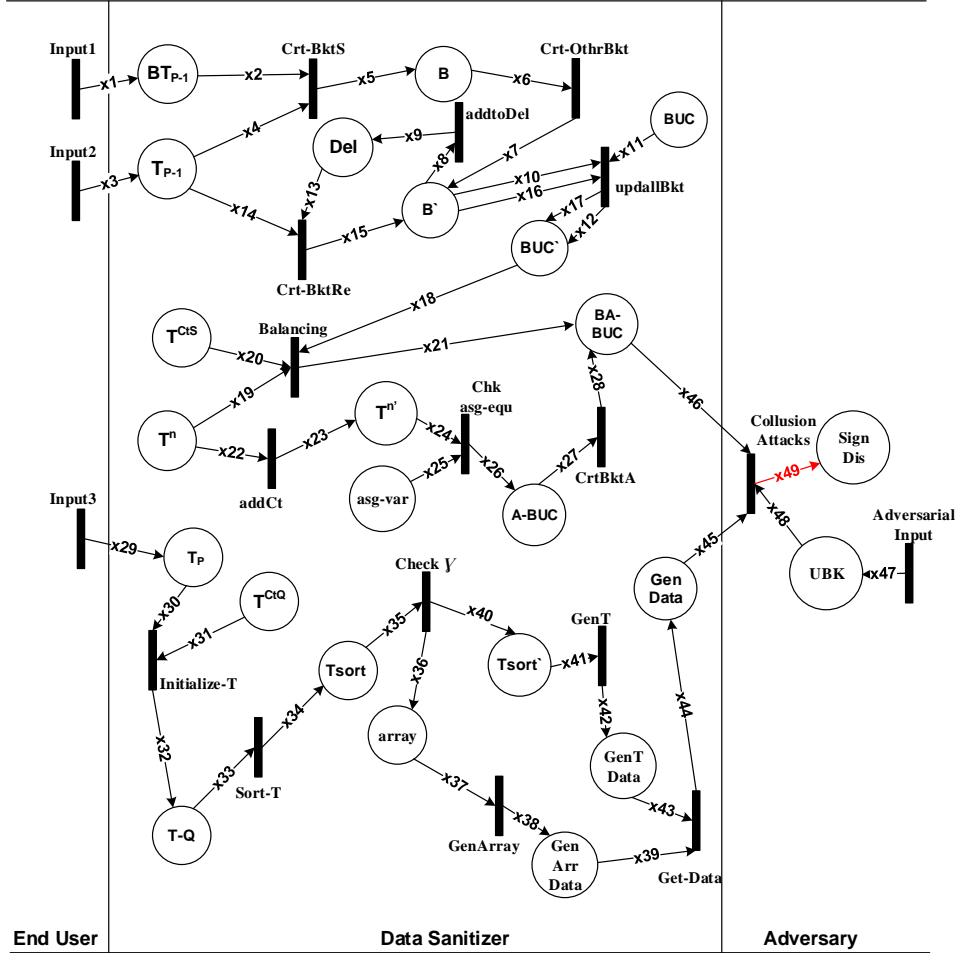


FIGURE 2 HLPN of τ -safe (l, k)-diversity model with adversarial attacks identification

performs an anonymization process over the raw data, seeking to minimize A^S values disclosure. Data is then published and can be subject to exploitation by an adversary.

Like τ -safety [11], the τ -safe (l, k) -diversity algorithm [12] consists of four phases i.e. classification, balancing, assignment, and generalization. The first three phases focus on A^S only while generalization phase focuses on A^{QI} . Bucket table is specifically used to store A^S signatures. The detailed algorithm is given in [12]. The transition rule is given for adversarial collusion attack only, for the identification of inconsistency signatures. A general critical review for τ -safe (l, k) -diversity algorithm with respect to HLPN transitions is given below.

The problem exists in classification phase that divides the records (T^S or T^F) into their correct buckets (l -diverse) along with their signatures but results in inconsistent signatures. Figure 2 shows that for a record to be added to a bucket if same signature exists then transition $Crt\text{-}BktS$, else transition $Crt\text{-}OthrBkt$ for a new signature bucket. The reinserted tuples from Del table are processed via transition $Crt\text{-}BktRe$ and finally all the different signatures are stored in BUC' via transition $updallBkt$. The balancing phase using transition $Balancing$, fills the unfilled created buckets with the same number of sensitive values. T^n or T^{CtS} (counterfeit tuples) are used to balance the buckets. While our proposed approach uses the $Cach$ table instead to these noise records. The assignment phase in (l, k) -diversity [12] is exactly the same as in [11]. This phase divides the records in the T^n into correct buckets. Transition $addCt$ adds counterfeit to T^{new} to make it l -eligible. The transition $Chk\ asg\text{-}equ$ check the T^n via $asg\text{-}var$. Transition $CrtBktA$ creates the l -eligible buckets and store it along with balanced buckets, all together at one place.

Generalization initializes the QI values with counterfeit QI part (T^{CtQ}) in transition $Initialize\text{-}T$, satisfying the k -anonymity and sorting it. The *Check* γ condition creates the generalize data at place $GenData$ which is received either via transition $GenArray$ or $GenT$. The places $GenData$ and $BA\text{-}BUC$ represents tables GT and BT respectively.

The inconsistent signatures created in classification phase allows the privacy breach. The subsection: 1.1 motivation, illustrate the collusion attack with example. The transition *Collusion Attack* is the adversary action that uses record signatures from previous alternate releases which leads to the privacy breach, given in rule 1. The *Collusion Attack* transition receives two data flows from places: $GenData$ and $BA\text{-}BUC$. Inside the $Sign_Disc()$ function the adversary correlate the sensitive knowledge of $BA\text{-}BUC$ and UBK with QI values from $GenData$ (Rule 1 depicts the Fig. 2 collusion attack transition). The adversary correlates same record signatures in different releases by continuously using Eq. 2 iteratively to get UBK , until it results in $Sign_{dis}$ and QI identification, against an individual.

$$R(\text{Collusion Attacks}) = \forall i45 \in x45, i46 \in x46, \forall i48 \in x48, i49 \in x49 | \\ Sign_Disc(\{i46[2], i48[2]\} \cup i45[2]) \rightarrow i49[2] = i1[2] \wedge i49[1] = i29[2]$$

Rule 1

5. Proposed (τ, m) -slicedBucket privacy model

Preserving privacy with enhanced utility is NP-hard problem specially in sequential dynamic publication. An intruder combines different releases to re-identify an individual sensitive information performing intersection or subtraction between the ECs in each release. The proposed (τ, m) -slicedBucket privacy model is an extension of τ -safe (l, k) -diversity model [12] with a balance in utility and privacy using the slicing [13] and cell generalization [14-15] approaches. The proposed (τ, m) -slicedBucket privacy model is defined as following.

Definition 11: (τ, m) -slicedBucket. A sequential published anonymized data $T^* = \{T_1^*, T_2^*, T_3^*, \dots, T_r^*\}$ where $(1 \leq j \leq r)$, is said to be (τ, m) -slicedBucket anonymous if the following conditions are satisfied:

1. Any release T_j^* at time j ($1 \leq j \leq r$) is m -unique and satisfies (τ, m) -slicedBucket $\forall j \in [0, 1]$.
2. Record signature $\forall t \in T_j^*$ during the lifetime $[r, r-1, r-2, r-3, \dots, r-n]$ must remain consistent such that $Sig(T_r^*(t)) = Sig(T_{r-1}^*(t)) = Sig(T_{r-2}^*(t)) = \dots = Sig(T_{r-n}^*(t))$, where $0 \leq i \leq n$ and if \exists a partial intersection probability $\{0 < p < 1\}$ during any release then $Sig(T_r^*(t)) \cap Sig(T_{r-1}^*(t)) \cap Sig(T_{r-2}^*(t)) \cap \dots \cap Sig(T_{r-n}^*(t)) = \emptyset, \forall$ internal updates.
3. Shuffle AC^{QI} and A^S values inside the sliced buckets randomly in each anonymized release i.e. $T_1^*, T_2^*, T_3^*, \dots, T_r^*$ before publishing.

The three conditions for the proposed (τ, m) -slicedBucket privacy model guarantees privacy for the anonymized release. Condition 1 guarantee m -uniqueness for each k -anonymous SBUC over time. Condition 2 maintain consistent signature for the same record during its lifespan. This ensures protection from collusion attack. Condition 3 guarantee protection against presence or membership attack. The random shuffling of sensitive values inside SBUCs in any T_j^* create fake tuples which misguide the adversary. Collectively the proposed system provides protection from collusion attack and membership disclosures attack.

The detail explanation for the proposed (τ, m) -slicedBucket privacy model are given below.

5.1 slicedBucket (SBUC)

To create slicedBucket (SBUC) the data are partitioned both vertically and horizontally using slicing technique. Signature of each slice must be m -invariant during each release. Vertical partitioning merges co-related A^{QI} s into AC^{QI} s where each column consists of subset of correlated attributes. This reduces data dimensionality and improve generalization for more utility. The actual SBUCs are formed by performing horizontal partitioning of the tuples. Horizontal partitioning groups the same or closer distance QI records into an almost homogeneous cluster or SBUC. Each SBUC is processed separately by applying the m -unique constraint. Thus, an anonymized release T_j^* , published at time j ($1 \leq j \leq r$) using the proposed (τ, m) -slicedBucket privacy model has AC^{QI} s and A^S . For preventing membership disclosure attack or presence attack the values inside each SBUC are permuted randomly to break the association among different columns and attributes but the association within each bucket will be preserved. This creates fake tuples which misguides the intruder. The fake tuples produced are not considered as invalid because we assume the common diseases for male and female.

Table 6 shows the anonymous release T_1^* at time 1 of the original microdata Table 1c. The three A^{QI} are $\{age, gender, zipcode\}$ and A^S is $\{Disease\}$. The co-related attributes age and $gender$ (via Eq. 3) are merged to form a column. The $zipcode$ and $disease$ are drawn as separate attributes. Values in each slicedBucket of the anonymized release are randomly permuted to break the association between the uncorrelated attributes.

5.2 (τ, m) persistent invariance

The (τ, m) keeps persist during all the releases to prevent signature variance inside the slicedBuckets for each record. τ keep all the internal updates for a record while m maintains the m -invariance diversity without counterfeit in slicedBuckets for sequential releases. Signature of a record must fulfill, $Sig(T_r^*(t)) = Sig(T_{r-1}^*(t)) = Sig(T_{r-2}^*(t)) = \dots = Sig(T_{r-n}^*(t))$ and if \exists a partial intersection probability $\{0 < p < 1\}$ during any release then $Sig(T_r^*(t)) \cap Sig(T_{r-1}^*(t)) \cap Sig(T_{r-2}^*(t)) \cap \dots \cap Sig(T_{r-n}^*(t)) = \emptyset$. If for a specific record during all releases the $\cap \neq \emptyset$ then (τ, m) adds the tuples from T^n or Cach table (Table 1b), otherwise will store back the same record in Cach for publishing in next releases. During internal updates, the same record signature from previous releases is checked in Upd table history (see Figure 3 (a) and (b)) otherwise new signature will be created according to definition 11.

Table 6, Table 7 and Table 8 are the 2-diverse dynamically published anonymized releases T_1^* , T_2^* and T_3^* produced through (τ, m) -slicedBucket privacy model from the original microdata Table 1c. Initially the sensitive values create m -invariance signatures and the corresponding records are added in the buckets. For a specific record, their signature

TABLE 6 2-Anonymous T_1^*

Patient	(Age, Gender)	Zip Code	Disease
p1	(21-27, M) [21]	47*** [47907]	Bronchitis
p2	(21-27, M) [27]	47*** [47303]	Aids
p3	(33-58, F)	47905	SSM-pos
p4	(33-58, F)	47905	Asthma
p5	(54, *) [M]	4790* [47906]	Flu
p6	(54, *) [F]	4790* [47902]	Dyspepsia
p7	(64-65, *) [64, F]	47308	Gastritis
p8	(64-65, *) [65, M]	47308	Cardiac

TABLE 7 2-Anonymous T_2^*

Patient	(Age, Gender)	Zip Code	Disease
p1	(21-22, M) [21]	4790* [47907]	Bronchitis
ch1	(21-22, M) [22]	4790* [47906]	Aids
p3	(33-45, F) [33]	4790* [47905]	Glaucoma
p10	(33-45, F) [45]	4790* [47901]	Diarrhea
p5	(54, *) [M]	4790* [47906]	Flu
p6	(54, *) [F]	4790* [47902]	Dyspepsia
p7	(60-64, *) [64, F]	4730* [47308]	Gastritis
ch2	(60-64, *) [60, M]	4730* [47302]	Cardiac
p8	(22-65, *) [65, M]	4730* [47308]	Dyspepsia
ch3	(22-65, *) [22, F]	4730* [47305]	Flu

TABLE 8 2-Anonymous T_3^*

Patient	(Age, Gender)	Zip Code	Disease
p1	(21-22, M) [21]	4790* [47907]	Bronchitis
ch1	(21-22, M) [22]	4790* [47906]	Aids
p3	(33-58, F) [33]	47905	SSM-pos
p4	(33-58, F) [58]	47905	Asthma
p10	(40-45, *) [45, F]	47*** [47901]	Diarrhea
p12	(40-45, *) [40, M]	47*** [47407]	Glaucoma
p5	(54, *) [M]	4790* [47906]	Flu
p6	(54, *) [F]	4790* [47902]	Dyspepsia
ch2	(60-64, *) [60, M]	4730* [47302]	Cardiac
ch4	(60-64, *) [64, F]	4730* [47304]	Gastritis
p8	(22-65, *) [65, M]	4730* [47308]	Dyspepsia
ch3	(22-65, *) [22, F]	4730* [47305]	Flu

remains consistent during each release. At time 1, 2-anonymous 2-diverse table (Table 6) is published. At time 2, the records $p3=SSM-pos$ and $p8=Cardiac$ are updated (internal update) to $p3=Gloucoma$ and $p8=Dyspepsia$, and $p2=Aids$ and $p4=Asthma$ are deleted (external update), while records $p9=Malaria$, $p10=Diarrhea$ and $p11=Diarrhea$ are newly inserted (external update), as shown in Table 7. Because of the same sensitive values for $p10$ and $p11$, one of the sensitive values say $p11$ is stored in Cach. Since $p9$ is the last tuple to accommodate and can use a tuple from Cach to

create a new signature but the algorithm does not do so because this will empty the Cach. Instead p9 is cached in Cach table till next release. At time 3, for Table 8, record p7 is deleted (external update), p3 updates (internal update) to the same sensitive value *SSM-pos* (chances of collusion attack but prevented from re-insertion) as was in Table 6, records p12=*Glaucoma* and p13=*Pneumonia* are newly inserted (external update) and p4 is re-inserted (internal update) from delete table i.e. Del. Figure 3 illustrates the data processing inside the buckets during T_2^* and T_3^* and the data inside the three supporting tables i.e. Del, Upd and Cach for each. The anonymized Tables 6,7 and 8 obtained via proposed Algorithm 1 (τ, m)-slicedBucket privacy model shows that after all these internal and external updates the signature of all the SBUCs remain consistent for the same record among themselves.

5.3 Empowering Cach tuples over Counterfeit tuples

Cache Table 1b named as Cach (obtained via Algorithm 2), is a novel idea to avoid the use of dummy counterfeit records. In re-publication of dynamic data, among the three major operations i.e. insertion, update and deletion in PPDDP, deletion is more challenging because of the *critical absence* dilemma. Most of the work [15], [32] related to dynamic publication considers insertion and update and do not focus the deletion operation. The well-known previous work that talk about the deletion are [10-12]. They have adopted the direct and easy option of fake tuples named as *counterfeit*. So, dynamic data publication adds the counterfeit while the differential privacy [19], [47] adds laplacian noise. We bridge the gap similar to [48] of syntactic and semantic data publishing via Cach table (actual records) but in a dynamic data sequential scenario where no counterfeit or noise is used and have consistent signatures. In this paper, the counterfeit avoidable scenario is considered. The basic purpose of the Cach is not to use any counterfeit in any case specially for critical absence caused by deletion operation. To the best of our knowledge, providing Cach records instead of counterfeit fake records is the first solution proposed in this work. Whenever there is a need to create m -unique signatures, records from Cach table are ready to use while all those tuples that cannot fulfill m -unique criteria are stored back in Cach table for future release. In this way deletion from and insertion into the Cach is performed automatically during the proposed algorithm execution.

Table 7 is the anonymized release at time 2, i.e. T_2^* uses ch1, ch2 and ch3 tuples from Cach table to fulfill the m -unique and signature consistency constraints. Similarly, for T_3^* ch4 is inserted from Cach. This whole scenario explains that the counterfeit tuples have no use at all in the proposed approach. The proposed Cach idea drastically enhances the truthfulness of the anonymized releases.

5.4 Proposed Algorithm

The proposed Algorithm 1 consists of the following steps: (i) Cache table creation (ii) Grouping correlated attributes (iii) Classification (iv) Balancing (v) Assignment (vi) Partitioning (vii) Cell Generalization.

Algorithm 1. (τ, m)-slicedBucket privacy algorithm Big Picture

Require: $T_r, T_{r-1}^*, \text{Cach}, \text{Del}, \text{Upd}$

1. Calculate Cach table: T_{rep} and T_{so}
 2. Calculate correlating attributes: chi-square method
 3. Calculate initially and after each release:
 $T^n = T_r - T_{r-1}, T^s = T_r \cap T_{r-1}, T^u = \text{Sig}(t_r) \neq \text{Sig}(t_{r-1})$
 4. Reinsertion: $T^r = \text{Sig}(\text{Del}(t))$
 5. SBUC = Classify($T^s, T_{r-1}^*, T^u, T^r, \text{Del}$)
 6. T_r^* = Balancing-Assignment-Partition-CellGeneralization
 7. Publish T_r^*
-

(i) Cache table Creation: The original microdata consists of thousands of records. There are limited number of unique A^S values frequently repeating over the tuples. Before anonymization, a Cach table is created (Algorithm 2) from the original microdata whose records will be used to beat the counterfeit tuples. Cach consists of two different types of records, (i) single record from repeated same SA values (ii) copy of a single occurred SA record. The Cach is a small table as compared to the whole dataset. For example, the dataset used in our experiment consists of 60000 records that have only 49 sensitive attribute values, which is only .082% of the whole dataset. This small amount of records for delaying in publishing will not affect the anonymized release, instead it can effectively increase the utility. Therefore, Cach table in the proposed (τ, m)-slicedBucket privacy model, not only vanishes the counterfeit tuples but also implies (ϵ, δ)-differential privacy [48] and provides strong privacy guarantee. Table 1b is an example of Cach table obtained from original Table 1a before its anonymization. Figure 3 (a) and (b) also illustrates the Cach table during T_2^* and T_3^* creation.

Algorithm 2. Cach Creation

for all records in Dataset **do**
 $T_{rep} \leftarrow \text{Comp rep(SA)}$
 Cach $\leftarrow \text{single_rec_frm}(T_{rep})$
 $T_{so} \leftarrow \text{Comp single_occured(SA)}$
 Cach $\leftarrow \text{copy_of}(T_{so})$
end for
return Cach

(ii) Grouping Correlated Attributes: Generalization i.e. k -anonymity, losses utility in case of high-dimensional data. For effective generalization that have higher data utility, the data must be low-dimensional. But low dimensions do not mean to drop any attribute from the dataset. We adopt the slicing [13] approach to create single column from multiple co-related attributes by calculating affinity between the attributes. This improves the privacy i.e. break the association between uncorrelated attributes. Because uncorrelated attribute values are less associated with each other and thus without grouping the correlated attributes the identification risk is higher. Algorithm 3 calculates the affinity \emptyset^2 between any two attributes A_i and A_j using Eq. 3.

Algorithm 3. Calculate co-related attributes

for all attributes in microdata T **do**
 Calc – affinity (A_i & A_j) i.e. $\emptyset^2(A_i, A_j)$
end for

A commonly used method for measuring the correlation between two categorical attributes is *chi-square* method or *mean-square contingency coefficient* [13-14]. This method groups the attributes according to their pairwise affinity. Attributes in different columns have high affinity within columns and low affinity between columns.

Let there are two attributes A_1 and A_2 with domain values $\{v_{11}, v_{12}, \dots, v_{1d_1}\}$ and $\{v_{21}, v_{22}, \dots, v_{2d_2}\}$ respectively, where d_1 and d_2 are domain sizes. The *mean-square contingency coefficient* between A_1 and A_2 can be calculated as shown in Equation (3).

$$\emptyset^2(A_1, A_2) = \frac{1}{\min(d_1, d_2) - 1} \sum_{i=1}^{d_1} \sum_{j=1}^{d_2} \frac{(f_{ij} - f_i \cdot f_j)^2}{f_i \cdot f_j} \quad (3)$$

where f_i . And f_j are the fractional occurrences of v_{1i} and v_{2j} in the data, respectively. f_{ij} is the fractional occurrence of v_{1i} and v_{2j} . Formal definitions of f_i and f_j which are the marginal totals of f_{ij} are: $f_i = \sum_{j=1}^{d_2} f_{ij}$ and $f_j = \sum_{i=1}^{d_1} f_{ij}$. Here $0 \leq \emptyset^2(A_1, A_2) \leq 1$.

In this work we adopt the same attributes correlation approach as in [9]. The attributes *{age, gender}* have high affinity and form the first column, i.e. AC^{Q1} . The zipcode and disease are considered as separated attributes. For attributes, age and gender, affinity example can be seen in Table 6, Table 7 and Table 8.

(iii) Classification: This phase is the main source of signatures inconsistency, which is not modeled properly in [11] and in [12]. In this phase the SBUCs are created and classify the records i.e. $t_1, t_2, t_3, \dots, t_n$ in T^s , T^u and T^r into their correct buckets using T_{r-1} ($1 \leq j \leq r$), Del and Upd tables. Each SBUC has a unique signature created. Attribute sensitive values and signature of SBUC (Sig(SB)) with respect to an individual record are considered in this phase only. Records in T^s get the same signatures from T_{r-1}^* . If signature for records in T^u or T^r exists in Upd or Del tables, respectively then the same signature SBUCs are created for those records. The entry for re-inserted records are deleted from Del table. The crucial part is if signature does not exist for T^u or T^r , or any one of them. If signature for records in T^u or T^r does not exists in Upd or Del tables then the tuples are stored in temporary arrays as T_1^{un} (updated new records) or T_1^{rn} (re-inserted new records), respectively and are partially treated as new tuples. The new signatures for new buckets for records in T_1^{un} or T_1^{rn} are created based on zero integration condition (see definition 11) and Algorithm 6 assignment condition. Because the work in [11] or [12] are not sure about the bucket size for the records in T^{un} or T^{rn} which can be β or $\beta+1$, where $\beta=m$. which is possibly the main reason for signature inconsistency. The new signatures from T^{un} or T^{rn} are stored permanently in Upd table along with the corresponding records to avoid the collusion attack. The classification phase is shown in Algorithm 4.

Figure 3 shows the classification phase for sequential releases of T_2^* and T_3^* at time 2 and time 3 respectively. At time 2 $T^s=\{p1, p5, p6, p7\}$, and at time 3 $T^s=\{p1, p5, p6, p8, p10, ch1, ch2, ch3\}$. At time 2, $T^u=\{p3=Glaucoma, p8=Dyspepsia\}$. Both p3 and p8 are new sensitive values for each record at time 2. So new signature are created having intersection zero with all its previous releases, i.e. $T_{r-2}, T_{r-3}, \dots, T_{r-n} = \bigcap_{r-2}^{r-n} Sig(t) = 0$ and checked via assignment

condition in Algorithm 6. The classification algorithm in [12] checks integration with only T_{r-1} and create new signature if there is zero integration while we check integration with all its previous releases for the same record, and verify the assignment condition. At time 3, $T^r=\{p4\}$ which is the only re-inserted record from $T^d=\{p2, p4\}$ at time 1. The p4 SBUC got the same signature as was at time 1 from Del table. At time 3, the only $T^u=\{p3\}$. If p3 updates to the same disease as at time 1, i.e. *SSM-pos*, then it can cause collusion attack. Both the τ -safety and τ -safe (l, k) -diversity do not consider the same sensitive value reappearance at time 3. In our proposed technique the same signature already available in Upd table as was at time 1, is checked and reused. New signature will not be created for p3 internal re-update.

Algorithm 4. Classification ($T^s, T_{r-1}^*, T^r, T^u, Del$)

```

Initialize SB = 0
for all records t in  $T^s$  do
  if ( $S(t) \in \text{Sig}(ST_{r-1}(t))$ ) then
    SB = Crt_SBkt( $\text{Sig}(ST_{r-1}(t))$ )
  end if
  put(SB, t)
  if ( $SB \notin \text{SBUC}$ ) then
    SBUC = SBUC  $\cup$  {SB}
  end if
end for
for all records t in  $T^u$  do
  if ( $S(t) \in \text{Upd}(\text{Sig}(ST_{r-2}(t)) \text{ or } \text{Sig}(ST_{r-3}(t)) \text{ or } \dots \text{ or } \text{Sig}(ST_{r-n}(t)))$ )
    SB = Crt_SBkt( $\text{Sig}(ST_{r-2}(t)) \text{ or } ST_{r-3}(t) \text{ or } \dots \text{ or } ST_{r-n}(t)$ )
  else
     $T_i^{un} = t_i$  //temporary array
  end if
  put(SB, t)
  if ( $SB \notin \text{SBUC}$ ) then
    SBUC = SBUC  $\cup$  {SB}
  end if
end for
for all records t in  $T^{un}$  do //processing updates as new tuples with  $\cap$ 
  SB = Crt_un_SBkt( $\text{Sig}(S(t)) \cap$ 
  Upd( $\text{Sig}(ST_{r-2}(t)) \text{ or } \text{Sig}(ST_{r-3}(t)) \text{ or } \dots \text{ or } \text{Sig}(ST_{r-n}(t))$ ) = 0
   $\wedge$  Call Algoritm 6, line 5 to 13 where  $T^n = T^{un}$ 
  addToUpd(t, Sig(t))
  put(SB, t)
  if ( $SB \notin \text{SBUC}$ ) then
    SBUC = SBUC  $\cup$  {SB}
  end if
end for
for all records t in  $T^r$  do
  if ( $S(t) \in \text{Sig}(Del(t))$ ) then
    SB = Crt_SBkt( $\text{Sig}(Del(t))$ )
    Delete-entry Sig( $Del(t)$ )
  else
     $T_i^{rn} = t_i$  //temporary array
  end if
  put(SB, t)
  if ( $SB \notin \text{SBUC}$ ) then
    SBUC = SBUC  $\cup$  {SB}
  end if
end for
for all records t in  $T^{rn}$  do //processing reinst as new tuples with  $\cap$ 
  SB = Crt_New_SBkt( $\text{Sig}(S(t)) \cap$ 
  Upd( $\text{Sig}(ST_{r-2}(t)) \text{ or } \text{Sig}(ST_{r-3}(t)) \text{ or } \dots \text{ or } \text{Sig}(ST_{r-n}(t))$ ) = 0
   $\wedge$  Call Algoritm 6, line 5 to 13 where  $T^n = T^{rn}$ 
  addToUpd(t, Sig(t))
  put(SB, t)
  if ( $SB \notin \text{SBUC}$ ) then
    SBUC = SBUC  $\cup$  {SB}
  end if
end for
return SBUC

```

(iv) Balancing: This phase balances the SBUCs created in classification phase through T^n and Cach table records. A SBUC is said to be balanced if every sensitive value in its signature is owned by the same number of tuples. There must be atleast one record respondent in the SBUC otherwise the bucket will be deleted. In the proposed approach we

(iv) Algorithm 5. Balancing
(SBs from Classifications phase, T^n , Cach)

```

for all records in  $T^n$  do
   $T_{rep} \leftarrow \text{Comp rep}(SA)$ 
  Cach  $\leftarrow \text{single\_rec\_frm}(T_{rep})$ 
   $T_{so} \leftarrow \text{Comp single\_occured}(SA)$ 
  Cach  $\leftarrow \text{copy\_of}(T_{so})$ 
end for
return Cach
for all  $SB_i$  in SBUC do
  if SB is unbalanced
    if (Sig(SBUC)  $\rightarrow$  Exp[ $T^n$ ])
      put(Exp[ $(T^n(S(t)))$ ], SB)
    else if (Sig(SBUC)  $\rightarrow$  Exp[Cach])
      put(Exp[ $(Cach(S(t)))$ ], SB)
    else
      (Sig(SBUC)  $\notin$  Exp[ $T^n$ ])  $\wedge$  (Sig(SBUC)  $\notin$  Exp[Cach])
      put(SBUC(S(t)), Cach)
    end if
  end if
end if
end for

```

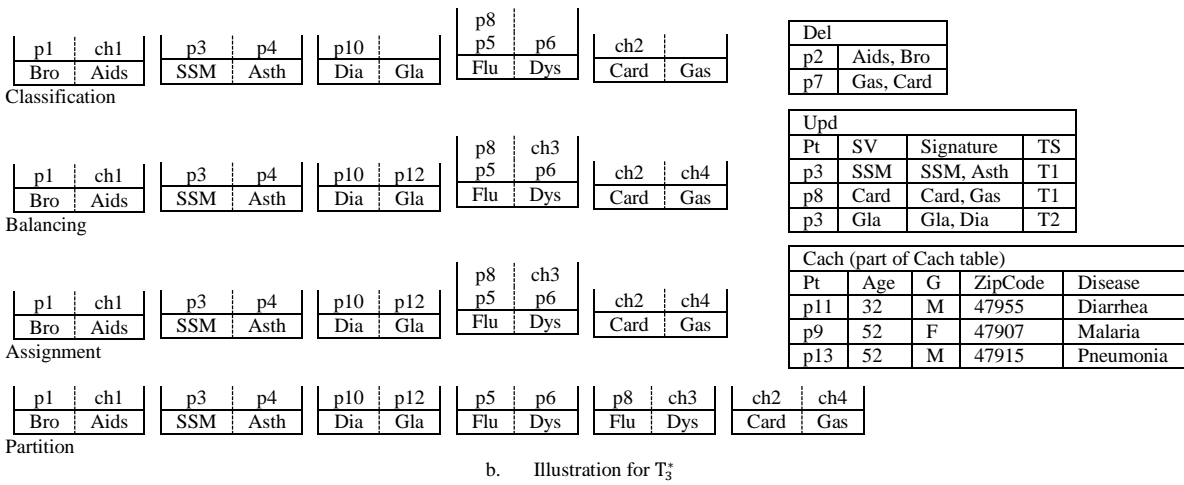
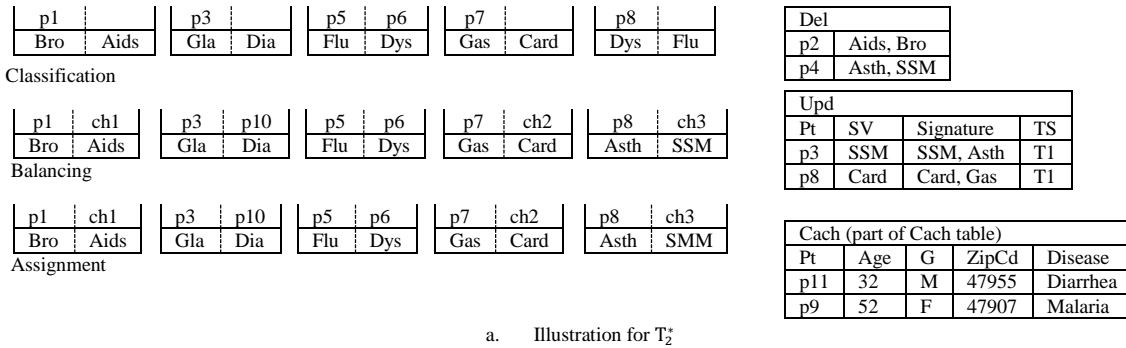


Figure 3 Processing of Bucket Data

do not simply add counterfeit to balance the SBUCs as in [10-12] instead the unbalanced SBUCs are filled with the expected T^n or records from Cach. Balancing Algorithm 5 begins by updating the Cach table (Section 5.4(i)). This not only populates the Cach but also implies the (ϵ, δ) -differential privacy [48] which improves the privacy in the published release.

Figure 3 illustrates the balancing phase at time 2 and time 3 during T_2^* and T_3^* release. As mentioned in subsection 5.2, at time 2, the three newly inserted tuples $T^n = \{p9=Malaria, p10=Diarrhea, p11=Diarrhea\}$. Because of the insertion of two similar sensitive values one (for example p11) is stored in Cach while the other (p10) is used to create a new signature. p10 balances the second SBUC. The p9 is stored in Cach (see subsection 5.2). The remaining SBUCs, first, fourth and fifth are balanced with Cach table ch1, ch2 and ch3 tuples. Similarly at time 3, $T^n = \{p12=Glaucoma, p13=Pneumonia\}$. p12 is used to balance signature for third SBUC. A tuple ch4=*Gastritis* from Cach is inserted to balance the ch2=*Cardiac* SBUC. Since there is no tuple for p13 to create signature, it is stored in Cach for publishing in next release. At the end of balancing phase all the SBUCs created during the classification phase are balanced.

(v) Assignment: This phase is the same as in [11] and [12] to assign the remaining T^n into their correct SBUCs. After completion of Algorithm 6, all the remaining T^n records if exists create new SBUCs and satisfy m -uniqueness property. For ensuring m -eligibility, records from Cach are added. The last records during EC creation in T^n that cannot create signatures with other records from T^n , are cached back that will help for balancing and signature eligibility during next release.

Similar to [10] the variables α and β are computed, to assign the correct number of records and to handle the SBUC signatures, respectively. Let $C = (sv_1, sv_2, sv_3, \dots, sv_\lambda)$ be the distinct A^S values list where λ is the total number of distinct A^S values in the T^n . Here, the number of tuples, $sv_i (1 \leq i \leq \lambda)$ are collected and sorted in descending order. In addition, $\gamma = |T^n|$ represents the total number of records. Signature of SBUC is created by choosing β sensitive values from each C such that SBUC has signature $(sv_1, sv_2, sv_3, \dots, sv_\beta)$. α picks the exact tuples with minimum distance in QI column from γ for a SBUC. The process of selecting α records repeats iteratively for each sensitive value in $sv_1, sv_2, sv_3, \dots, sv_\beta$ to form signature of SBUC in each cycle such that the remaining records must be m -eligible. Since the sensitive values are in descending order the most frequent sensitive value is sv_1 or $sv_{\beta+1}$. Therefore, α and β are formulated via inequalities $\alpha \leq sv_\beta$ and $sv_1 - \alpha \leq \frac{(\gamma - \alpha \cdot \beta)}{m}$ and $sv_{\beta+1} \leq \frac{(\gamma - \alpha \cdot \beta)}{m}$ if α exists, otherwise β is incremented to solve α again. This assignment condition is also used by the classification phase for T^{un} and T^{rn} records. At the end of the assignment phase all the T^n have been assigned to their correct balanced SBUCs.

Figure 3 (a) and (b) depicts the assignment phase in bucketized form for T_2^* and T_3^* . At time 2 and time 3, although $T^n = \{p9=Malaria\}$ and $T^n = \{p13=Pneumonia\}$ respectively but have been stored in Cach as explained in balancing. This means that the last tuples are stored back in Cach because of not having suitable record to create signature with. After completion of assignment phase all the SBUC are balanced and completed. No tuple in T^n is further lifted to accommodate.

(iv) Algorithm 6. Assignment (T^n , SBUC, Cach)

```

initialize  $\lambda =$  Total distinct sensitive attribute values in  $T^n$ 
if  $\left(\frac{T^n}{m} \geq m\right)$  then
    add tuples from Cach in  $T^n$  //we will not add counterfeit in  $T^n$ 
end if
while  $|T^n \neq \emptyset|$  do
     $\gamma = |T^n|$ 
    Calculate  $C = (sv_1, sv_2, sv_3, \dots, sv_\lambda)$ , i. e.  $sv_i (1 \leq i \leq \lambda)$ 
     $\beta = m$ 
     $\alpha =$  largest positive integer that satisfy the inequalities below
    if  $(\alpha \leq sv_\beta$  and  $sv_1 - \alpha \leq \frac{(\gamma - \alpha \cdot \beta)}{m}$  and  $sv_{\beta+1} \leq \frac{(\gamma - \alpha \cdot \beta)}{m})$  then
         $\beta = \beta + 1$ 
        go to line  $\alpha$  calculation above
    end if
    Create – Bucket SB having  $(Sig(SB) = (sv_1, sv_2, sv_3, \dots, sv_\beta))$ 
    if  $(SB \notin SBUC)$  then
         $SBUC = SBUC \cup \{SB\}$ 
    end if
    for  $i = 1$  to  $\beta$  do
         $SB \leftarrow \alpha$  nearest tuples with  $sv_i$  from  $T^n$ 
    end for
end while
return SBUC

```

(vi) Partition: This phase deals with the A^{QI} s. Using Algorithm 7 the microdata T has been transformed to SBUCs and needs to partition to their individual SBUCs with minimum distance between the A^{QI} values in a sorted form. SBUCs produced by assignment phase usually contains multiple of s tuples, i.e. $s \geq m$ sensitive values. The extra s tuples are removed from SBUCs by splitting or partitioning the SBUCs into balanced new SBUCs i.e. $SBUC_{new}$ with the same signature. While creating $SBUC_{new}$ all those s tuples are selected from a SBUC that have the minimum distance between the A^{QI} values for the purpose of minimum cell generalization which will increase the utility of the published data. The partitioning always produces m -eligible SBUCs because SBUCs are multiples of s . The resulted $SBUCs_{new}$ are sorted in ascending order with respect to their QI values.

(iv) Algorithm 7. Partition ($SBUC, SBUC_{new}$)

```

 $S_{sbuc} = \{SBUC\}$ 
while ( $SBUC > s$ ), where  $s \geq m$ 
   $\frac{SBUC}{s}$  and
   $Crt_{new\_SBkt}(\text{Sig}(SBUC_{new})) = \text{Sig}(SBUC)$ ,
  with min. interval w.r.t.  $A_1^{QI}$ 
end while
for each  $SBUC_{new}$  do
  sort each  $SBUC_{new}$ 
end for
return individual  $SBUCs_{new}$ 

```

(vii) Cell Generalization: In a relational approach a cell is an intersection of row and column. In our approach, a cell is a cross-section of column and a slicedBucket. We adopt a flexible generalization technique named as cell generalization (Algorithm 8) where each cell is generalized independently. In a microdata table T , if a column is C_c , ($1 \leq i \leq c$), a slicedBucket is B_b , ($1 \leq j \leq b$), then a cell is represented as $CB_{(i,j)}$. During the generalization, each attribute value of $CB_{(i,j)}$ is generalizes till the privacy requirements. Cell generalization has more utility than column or cut generalization [15], because it does not generalize the whole slicedBucket. For example, if the dataset T consists of attributes, age, gender, zipcode and disease. The following cell generalization rules in a function $gen()$ are followed independently on each attribute.

- Age will be generalized to any interval in the range $[a, b]$ where $a, b \in A_1$ and $a \leq b$.
- Gender can be M or F, therefore it is generalized by suppression only, i.e. the original value is retained or it is suppressed.
- Zipcode is generalized in the prefix format. For example zipcode 47901 will be generalized to any of the form, 4790 *, 479 **, 47 ***, 4 ****, ***** , but will be generalized as minimum as possible.
- Disease is a sensitive attribute and no generalization is allowed.

For example, Table 6, Table 7 and Table 8 shows the cell generalization for the three anonymized releases T_1^* , T_2^* , T_3^* , respectively.

(iv) Algorithm 8. Cell Generalization (T_r)

```

for each  $SBUC_i$  in  $T_r$  do
  check the tuple  $t_i$  validation
  if invalid record exists then do
     $genr\ SBUC_j = gen(CB_{(i,j)})$  to satisfy  $k$ -anonymity
  end if
end for
return  $T_r^*$ 

```

After generalization, attribute values on both sides i.e. AC^{QI} and A^S values are randomly permuted in each SBUC to break the linking between the uncorrelated attributes i.e. AC^{QI} and A^S . This limits the knowledge of an adversary and prevents the membership disclosure attack. The identifier attributes are removed and T_r^* is ready to publish. In the next section we formally model our proposed approach and show its verification against collusion attacks and membership disclosure attacks using the HLPN rules.

5.5 Formal modeling and analysis for (τ, m) -slicedBucket privacy model

The microdata is processed through the (τ, m) -slicedBucket privacy model to produce an anonymized form that can prevent from collusion attacks and membership disclosure attacks. In this section we provide formal modelling for the

proposed privacy model through HLPN and its invalidation with respect to identified collusion and membership attacks.

The HLPN for the proposed (τ, m) -slicedBucket privacy model with adversarial attacks invalidation is given in Figure 4, which is broadly consists of three entities: end user, trusted data sanitizer and adversary. There are 27 Places (P) and 22 Transitions (T) involved in the modelling process. The P description with variable type are described in Table 9a and the identification of P and mapping is in Table 9b. To begin with the model transitions, new token enters the model through the transition Input 1. The proposed algorithm complete process is depicted in rules 2-21, while rules 22 and 23 shows the prevention from collusion attacks and membership disclosure attacks.

The transitions Comp – rs and Single – rec – frm, compute the repeated A^S values records and select single records from each, respectively. Similarly, transition Comp – so computes single occurred A^S value records, while transition Copy – of stores single copy of that record in Cach table. So, the rules for the transitions to create the Cach table are 2,3,4 and 5 as follows.

$$\begin{aligned} R(\text{Comp} - \text{rs}) &= \forall i2 \in x2, i3 \in x3 \\ i3[1] &:= i2[1] \wedge i3[2] := i2[2] \wedge (i3[3] := \text{Comprs}(i2[3])) \wedge x3' := x3 \cup \{(i3[1], i3[2], i3[3])\} \end{aligned} \quad \text{Rule 2}$$

$$\begin{aligned} R(\text{Single} - \text{rec} - \text{frm}) &= \forall i4 \in x4, i5 \in x5 \\ i5[1] &:= i4[1] \wedge i5[2] := i4[2] \wedge (i5[3] := \text{srfm}(i4[3])) \\ x5' &:= x5 \cup \{(i5[1], i5[2], i5[3])\} \end{aligned} \quad \text{Rule 3}$$

$$\begin{aligned} R(\text{Comp} - \text{so}) &= \forall i6 \in x6, i7 \in x7 \\ i7[1] &:= i6[1] \wedge i7[2] := i6[2] \wedge (i7[3] := \text{Compso}(i6[3])) \wedge x7' := x7 \cup \{(i7[1], i7[2], i7[3])\} \end{aligned} \quad \text{Rule 4}$$

TABLE 9a Description of places and types used in HLPN for (τ, m) -slicedBucket model

Types	Description
PID	An integer type for Patient user identifier
BID	An integer for bucket identifier
T_{rs}	Records have repeated same sensitive values
T_{so}	Records have single occurrence of sensitive values.
Sign	Signature of SA at a specific level/phase.
Cach'	Place holding updated Cach table
T'	Holding T with correlated attributes after Cach deduction
T'_{r-1}	Place holding dataset from previous release.
QID_{cor}	Type for Correlated QI attributes
$Sign_s, Sign_n, Sign_{n'}$	Signature for same, updated and new tuples respectively
FlagTF	Boolean condition for either one or zero.
SB	Holds different record signatures during classification phase
rnSB	Holds new signatures for T' during classification phase
unSB	Holding record signatures for T^n during classification phase
SBUC	All updated record signatures during classification phase
SBUC'	Holds final record signatures after Balancing & Assignment
Asg_Var	Variables to check assignment equ. inequalities: α for no. of buckets in T^n , β is the no. of distinct SVs in bucket and γ is total number of distinct SVs in T^n .
A_SBUC	Holding set of records signatures after assignment phase.
SBUC_new	Holds top k individuals QIs in the data set.
SBUC_sort	Holds sorted QIs records in the data set.
SBUC_gen	Holds k -anonymous data in sliced bucket having QI and SA in each sliced bucket and is ready to publish.
UBK	Updated background knowledge
Sign Dis	Holds adversarial disclosed signatures.
MShip Dis	Holds adversarial disclosed

TABLE 9b Mapping of data types on places

Places	Description
$\varphi(T)$	$\mathbb{P}(PID \times QID \times SAs)$
$\varphi(T_{rs})$	$\mathbb{P}(PID \times QID \times SA_{ms})$
$\varphi(T_{so})$	$\mathbb{P}(PID \times QID \times SA_{so})$
$\varphi(\text{Cach})$	$\mathbb{P}(PID \times QID \times SAs)$
$\varphi(T')$	$\mathbb{P}(PID \times QID_{cor} \times SAs)$
$\varphi(T'_{r-1})$	$\mathbb{P}(QI \times Sign_{r-1})$
$\varphi(T^s)$	$\mathbb{P}(PID \times QID \times SA_{sign})$
$\varphi(T^r)$	$\mathbb{P}(PID \times QID \times SA_{dsign})$
$\varphi(T^u)$	$\mathbb{P}(PID \times QID \times SA_{updSign})$
$\varphi(T^n)$	$\mathbb{P}(PID \times QID \times SAs)$
$\varphi(\text{Del})$	$\mathbb{P}(PID \times QID \times Sign_{r-1})$
$\varphi(\text{Upd})$	$\mathbb{P}(PID \times QID \times Sign_u)$
$\varphi(\text{Flag})$	$\mathbb{P}(\text{ConditionTF})$
$\varphi(\text{SB})$	$\mathbb{P}(PID \times QI \times Sign_s \times Sign_n \times Sign_{n'} \times BID)$
$\varphi(\text{rnSB})$	$\mathbb{P}(PID \times QI \times Sign_{rnc} \times BID)$
$\varphi(\text{unSB})$	$\mathbb{P}(PID \times QI \times Sign_{unc} \times BID)$
$\varphi(\text{SBUC})$	$\mathbb{P}(PID \times QI \times Sign_{all} \times BID)$
$\varphi(\text{SBUC}')$	$\mathbb{P}(PID \times QID \times Sign_{bal\&n} \times BID)$
$\varphi(\text{Asg_Var})$	$\mathbb{P}(\alpha \times \beta \times \gamma)$
$\varphi(\text{A_SBUC})$	$\mathbb{P}(PID \times QI \times Sign_n \times BID)$
$\varphi(\text{SBUC_new})$	$\mathbb{P}(PID \times QI \times Sign_{prt} \times BID \times s \times m)$
$\varphi(\text{SBUC_sort})$	$\mathbb{P}(PID \times QI_{sort} \times Sign_{prt} \times BID)$
$\varphi(\text{SBUC_gen})$	$\mathbb{P}(QI_{kAnony} \times Sign_{prt})$
$\varphi(\text{UBK})$	$\mathbb{P}(QI_{adv} \times Sign_{adv})$
$\varphi(\text{Sign Dis})$	$\mathbb{P}(QI \times Sign_{disclosure})$
$\varphi(\text{MShip Dis})$	$\mathbb{P}(QI_{disclosure})$

$$\begin{aligned} R(\text{Copy} - \text{of}) &= \forall i8 \in x8, i9 \in x9 \\ i9[1] &:= i8[1] \wedge i9[2] := i8[2] \wedge (i9[3] := \text{Copyof}(i8[3])) \wedge x9' := x9 \cup \{(i9[1], i9[2], i9[3])\} \end{aligned} \quad \text{Rule 5}$$

Transition Comp – cor – atr calculates the correlated quasi attributes i.e. $(i10[2_{(j)}])_{\forall j \in T(qi)}$ to reduce the dimensionality of the data in the remaining microdata T, which is mapped to the following rule 6.

$$\begin{aligned} R(\text{Comp} - \text{cor} - \text{atr}) &= \forall i10 \in x10, i11 \in x11 \\ i11[1] &:= i10[1] \wedge (i11[2] := \text{affinity}(i10[2_{(j)}])_{\forall j \in T(qi)}) \wedge i11[3] := i10[3] \wedge \\ x11' &:= x11 \cup \{(i11[1], i11[2], i11[3])\} \end{aligned} \quad \text{Rule 6}$$

Transition Idntfy – tpls separates the T^s (i.e. $i15[1]$), T^u (i.e. $i17[1]$) and T^n (i.e. $i16[1]$) to create signatures for each accordingly.

$$\begin{aligned}
 R(\text{Idntfy} - \text{tpls}) &= \forall i12 \in x12, i14 \in x14, i15 \in x15, i16 \in x16, i17 \in x17 \\
 i15[1] &:= i12[1] \wedge (i12[2] = i14[1]) \rightarrow (i15[2] := i12[2]) \wedge (i12[3] = i14[2]) \rightarrow (i15[3] := i12[3]) \\
 x15' &:= x15 \cup \{(i15[1], i15[2], i15[3])\} \\
 i16[1] &:= i12[1] \wedge (i16[2] \neq (i12[2], \exists i12[2] \notin i14[1])) \wedge (i16[3] \neq (i12[3], \exists (i12[3] \notin i14[2]))) \\
 x16' &:= x16 \cup \{(i16[1], i16[2], i16[3])\} \\
 i17[1] &:= i12[1] \wedge (i17[2] := (i12[2], \exists i12[2] \in i14[1])) \wedge (i17[3] \neq (i12[3], \exists (i12[3] \notin i14[2]))) \\
 x17' &:= x17 \cup \{(i17[1], i17[2], i17[3])\}
 \end{aligned}$$

Rule 7

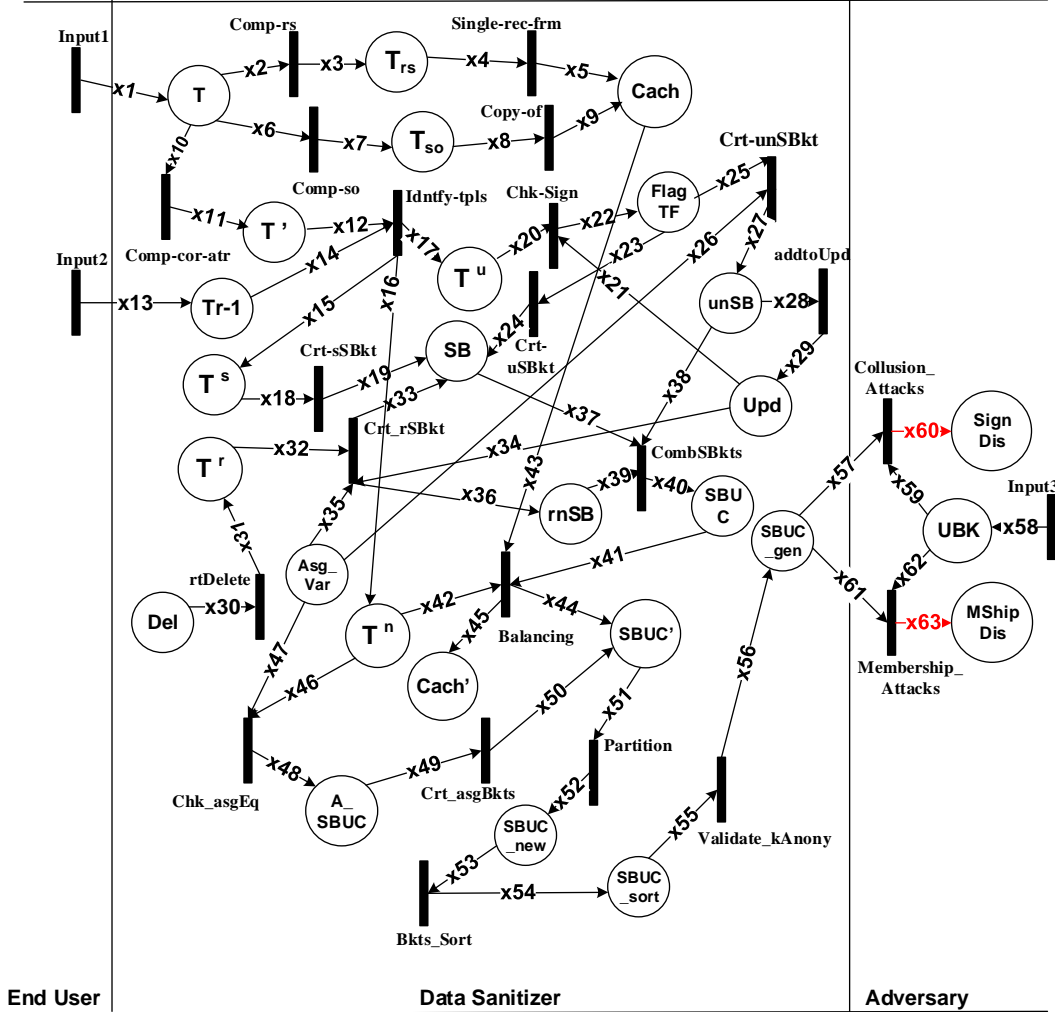


FIGURE 4 HLPN of (τ, m) -slicedBucket Privacy model with adversarial attacks invalidation

Transition $\text{Crt} - \text{sSBkt}$ mapped in rule 8 create same as previous buckets through $i19[3] := \text{ssbkt}(i18[3])$ for T^s , while transition $\text{Chk} - \text{sign}$ check the signatures for T^u tuples in rule 9. The signatures checking are modeled as either $i20[3] := i21[3]$ or $i20[3] \neq i21[3]$.

$$\begin{aligned}
 R(\text{Crt} - \text{sSBkt}) &= \forall i18 \in x18, i19 \in x19 \\
 i19[1] &:= i18[1] \wedge i19[2] := i18[2] \wedge (i18[3] \in i19[3]) \rightarrow (i19[3] := \text{ssbkt}(i18[3])) \wedge \\
 x19' &:= x19 \cup \{(i19[1], i19[2], i19[3])\}
 \end{aligned}$$

Rule 8

$$\begin{aligned}
 R(\text{Chk} - \text{Sign}) &= \forall i20 \in x20, i21 \in x21, i22 \in x22 \\
 \text{Check} &((i20[3] := i21[3]) \rightarrow i22[1] := \text{TRUE} \wedge x22' := x22 \cup \{(i22)\}) \vee
 \end{aligned}$$

$Check((i20[3] \neq i21[3]) \rightarrow i22[1] := FALSE \wedge x22' := x22 \cup \{(i22)\})$

Rule 9

For the TRUE case in rule 9, transition $Crt - uSBkt$ uses signatures (i.e. $i24[4] := usbkt(i24[4])$) for given T^u tuples that are already available in place Upd , otherwise new signatures through transition $Crt - uSBkt$ are created using the Asg_Var place. Both types of signatures creations for T^u tuples are mapped in rules 10 and 11. The $addtoUpd$ transition insert the new signatures (i.e. $i24[4] := usbkt(i24[4])$) into Upd place for future references, as shown in rule 12.

$R(Crt - uSBkt) = \forall i23 \in x23, i24 \in x24 |$
 $i23[1] = TRUE \rightarrow i24[4] := usbkt(i24[4]) \wedge i24[1] := i20[1] \wedge$
 $x24' := x24 \cup \{(i24[1], i24[4])\}$

Rule 10

$R(Crt - unSBkt) = \forall i25 \in x25, i26 \in x26, i27 \in x27 |$
 $i25[1] = FALSE \rightarrow i27[3] := unSBkt(asg - var(i27[3], (i26[1], i26[2], i26[3])))_{\exists(i27[3] \cap (v i21[3])=0)} \wedge$
 $x27' := x27 \cup \{(i27[3])\}$

Rule 11

$R(addtoUpd) = \forall i28 \in x28, i29 \in x29 |$
 $i29[1] := i28[1] \wedge i29[2] := i28[2] \wedge i29[3] := utUpd(i28[3]) \wedge x29' := x29 \cup \{(i29[1], i29[2], i29[3])\}$

Rule 12

To handle the T^r tuples, transition $Crt - rSBkt$ creates sliced buckets for re-inserted tuples have same signature (i.e. $i33[5] := rSBkt(i32[3])$) as stored in Del table, otherwise create new signatures such that $(Sign \in T^r) \cap (Sign \in Upd) = 0$, based on assignment phase technique. Transition $rtDelete$ deletes the reinserted tuples from Del table.

$R(rtDelete) = \forall i30 \in x30, i31 \in x31 |$
 $i31[1] := delrTpl(i30[1]) \wedge i31[2] := i30[2] \wedge i31[3] := i30[3]$
 $x31' := x31 \cup \{(i31[1], i31[2], i31[3])\}$

Rule 13

$R(Crt - rSBkt) = \forall i32 \in x32, i33 \in x33, i34 \in x34, i35 \in x35, i36 \in x36 |$
 $i33[1] := i32[1] \wedge i33[2] := i32[2] \wedge (i33[5] := rSBkt(i32[3]) \wedge x33' := x33 \cup \{(i33[1], i32[2], i33[5])\}) \vee$
 $i36[1] := i32[1] \wedge i36[2] := i32[2] \wedge (i36[3] := rnSBkt(asg - var(i36[3], i35[1], i35[2], i35[3])))_{\exists(i32[3] \cap (v i34[3])=0)} \wedge$
 $x36' := x36 \cup \{(i36[1], i36[2], i36[3])\}$

Rule 14

All buckets created in rules 8,10,11 are updated through transition $CombSBkts$ by taking union of all those buckets (where $Bkt_i \cap Bkt_j = 0$, and $i \neq j$) in order to have signatures for complete dataset at one place. Transition $Balancing$ adds the records from $Cach$ table or from T^n records to fill the required empty gap in the created sliced buckets. The transitions $CombSBkts$ and $Balancing$ are mapped to the following rules 15 and 16.

$R(CombSBkts) = \forall i37 \in x37, i38 \in x38, i39 \in x39, i40 \in x40 |$
 $i40[1] := union(i37[1], i38[1], i39[1]) \wedge i40[2] := union(i37[2], i38[2], i39[2]) \wedge$
 $i40[3] := union(i37[3], i37[4], i37[5], i38[3], i39[3]) \wedge i40[4] := union(i37[6], i38[4], i39[4]) \wedge$
 $x40' := x40 \cup \{(i40[1], i40[2], i40[3], i40[4])\}$

Rule 15

$R(Balancing) = \forall i41 \in x41, i42 \in x42, i43 \in x43, i44 \in x44 |$
 $i44[1] := (i41[1] + (i42[1] \vee i43[1])) \wedge (i44[2] := i41[2] + (i42[2] \vee i43[2])) \wedge$
 $(i44[3] := Balance(i41[3]_{i_{v i41[3]} \in i} + (i42[3] \vee i43[3]))) \wedge x44' := x44 \cup \{(i44[1], i44[2], i44[3])\}$

Rule 16

Algorithm 6 is formally depicted in transitions $Chk - asgEq$ and transition $Crt - asgBkts$ which checks the assignment equation to create buckets for the remaining T^n tuples (i.e. $\forall T^n \neq 0$). Transition $Partition$ separates the multiple QI -groups (i.e. $i52[3] := part(i51[3])_{\exists s \geq m}$) that exists in a single bucket and keep them in an individual sliced bucket that have minimum QI distance between them. The buckets with respect to QI columns are sorted in rule 20.

$R(Chk - asgEq) = \forall i46 \in x46, i47 \in x47, i48 \in x48 |$
 $i48[1] := i46[1] \wedge i48[2] := i46[2] \wedge (i46[3] \neq 0) \rightarrow \{i48[3] := asg - var(i46[3]_{i_{v i46[3]} \in i}, i47[1], i47[2], i47[3])\}$
 $x48' := x48 \cup \{(i48[1], i48[2], i48[3])\}$

Rule 17

$R(Crt - asgBkts) = \forall i49 \in x49, i50 \in x50 |$
 $i50[1] := i49[1] \wedge i50[2] := i49[2] \wedge i50[3] := asgBkts(50[3] \cup i49[3])_{\exists i49[3]_x \in 49, i50[3]_y \in 50 | x \neq y} \wedge i50[4] := i49[4]$

$x50' := x50 \cup \{(i50[1], i50[2], i50[3], i50[4])\}$ Rule 18

$R(\text{Partition}) = \forall i51 \in x51, i52 \in x52$
 $i52[1] := i51[1] \wedge i52[2] := i51[2] \wedge i52[3] := \text{part}(i51[3])_{\exists s \geq m} \wedge i52[4]_{\forall i52[4] \exists \text{SingleSign}} := i51[4]$
 $x52' := x52 \cup \{(i52[1], i52[2], i52[3], i52[4])\}$ Rule 19

$R(\text{Bkts} - \text{Sort}) = \forall i53 \in x53, i54 \in x54$
 $i54[1] := i53[1] \wedge i54[2] := \text{sort} - \text{asc}(i53[2]) \wedge i54[3] := i53[3] \wedge i54[4] := i53[4]$
 $x54' := x54 \cup \{(i54[1], i54[2], i54[3], i54[4])\}$ Rule 20

Transition `Validate_kAnony` is used to k -anonymize the QI columns for each cell. And the k -anonymous cells along with the sliced buckets are randomly permuted to prevent against membership attack. This final transition performs the last step to transform the raw data into the anonymized SBUCs form and is ready to publish. Rules 21 depicts the procedure as follows.

$R(\text{Validate_kAnony}) = \forall i55 \in x55, i56 \in x56$
 $i56[1] := \text{cellGen}(i55[1], i55[2]) \wedge i56[2] := \text{randPermute}(i55[3], i55[4])$
 $x56' := x56 \cup \{(i56[1], i56[2])\}$ Rule 21

The adversary combines the published releases, UBK and external available information to disclose the SA values and the corresponding user identifications in a specific release. The transitions `Collusion – Attack` prevent the intruder from disclosing an individual record signature i.e. $\forall (\text{Sign}_{\text{PRT}} \cup \text{Sign}_{\text{adv}}) \neq \text{Sign}_{\text{disclosure}}$ and hence the intruder gets the \emptyset result. The function `randPermute()` in rule 21 prevents the membership attack (i.e. $\text{QI}_{\text{kAnony}} \cup \text{QI}_{\text{adv}} \neq \text{QI}_{\text{disclosure}}$) as depicted in transition `Membership – Attack`.

Therefore, in each SBUC the proposed (τ, m) -slicedBucket keeps a consistent signature against each record or updated signature with condition: $\text{Sig}(\text{ST}_i(t)) \cap \text{Sig}(\text{ST}_j(t)) = 0, \wedge i \neq j$ and fulfilment of assignment equation to achieve m -unique signatures. Attraction for slicing to prevent membership disclosure is undoubtful. For example, if an adversary can get values of A^{QI} of an individual in AC^{QI} in a specific 3-anonymous SBUC then it is 33% chance that he can trace exact patient. In case if PID is correctly traced even then it is hard to identify its sensitive value because of the signature consistency in each release. The next section demonstrates the experimental proof of the proposed privacy model.

$R(\text{Collusion} - \text{Attack}) = \forall i57 \in x57, i59 \in x59, i60 \in x60$
 $\text{SignDis}(i57[2]) \rightarrow (i57[2] \cup i59[2]) \neq i60[2] \vee \text{SignDis}(i57[2] \cup i59[2]) = \emptyset$ Rule 22

$R(\text{Membership} - \text{Attack}) = \forall i61 \in x61, i62 \in x62, i63 \in x63$
 $\text{MShipDis}(i61[1]) \rightarrow (i61[1] \cup i62[1]) \neq i63[1] \vee \text{MShipDisc}(i61[1] \cup i62[1]) = \emptyset$ Rule 23

6 Experiment and analysis

This section validates the proposed (τ, m) -slicedBucket privacy model and evaluates the experimental results in comparison with m -invariance, τ -safety and τ -safe (l, k) -diversity algorithms. Various quality measures are available to test the effectiveness of the privacy models. In this paper, we use the normalized certainty penalty (NCP) [29], Kullback-Leibler (KL)-Divergence [11], query accuracy [11-12], number of counterfeits used and algorithm execution time analysis.

The proposed algorithm has been implemented in Python 3.7, on a machine having Intel Core i5 2.39 GHz processor with 4GB RAM, installed on Windows 10 operating system. Though, we did not have code for m -invariance, τ -safety and τ -safe (l, k) -diversity, we implemented these algorithms keeping their values as close to the original as possible. We have used the publicly available real dataset Adults; taken from U.C. Irvine Machine Learning Repository, <https://archive.ics.uci.edu/ml/datasets>. From the Adult dataset, we have randomly chosen 60,000 tuples and four attributes to perform experiments. Among the four attributes: age, gender, and zip code have been selected as A^{QI} s and occupation as A^{S} . After first release, for external updates in each subsequent release almost 3,500 tuples have randomly been deleted and 4,000 tuples have been inserted from the remaining tuples. For internal updates, 1,000 tuples have been updated from previous release with respect to A^{S} and 1,000 tuples have been re-inserted from the Del table. In this way, 20 times Adults dataset has been produced and has been anonymized by the implemented algorithm.

6.1 Anonymization Quality

This sub-section examines the quality of the anonymized releases. NCP [12], [29] and KL Divergence [11] are the statistical metrics that measure the utility loss caused by anonymization.

(i) Normalized Certainty Penalty (NCP)

The NCP used in [12], [29], is a well-known measure of utility loss for the anonymized releases using the QI attributes in the dataset. The utility loss for a single QI value, for a record t , and for the complete anonymized release T^* are given in Equations (4), (5) and (6) respectively.

$$NCP_{QI_i}(t) = \frac{z_i - y_i}{|QI_i|} \quad (4)$$

$$NCP(t) = \sum_{i=1}^q w_i \cdot NCP_{QI_i}(t) \quad (5)$$

$$NCP(T^*) = \sum_{t \in T^*} NCP(t) \quad (6)$$

where y_i and z_i are the generalized lower and upper attribute values in a domain and $|QI_i|$ is the domain of attribute QI_i , q are the total A^{QI} s and w_i are weights of attributes.

Figure 5 depicts utility measurement for the anonymized releases using NCP. The algorithm was tested under different parameters i.e. release time and m value. Figure 5(a) compares m -invariance, τ -safety, τ -safe (l, k)-diversity and the proposed (τ, m) -slicedBucket privacy algorithms. The proposed algorithm is tested to anonymize the Adults dataset for 20 release times having fixed $m=6$ during each release. In Figure 5(a) the y-axis is the percentage value from Eq. 6 during the 20 releases on x-axis. The higher NCP% value shows the highest utility loss, while (τ, m) -slicedBucket has the lowest information loss. m -invariance algorithm performs worst because it cannot control the generalization while τ -safety divides the records into their nearest ECs. In τ -safe (l, k)-diversity, although there is less information loss but still generalization exists in each EC and has certain amount of utility loss. The (τ, m) -slicedBucket privacy model performs comparatively good because of slicing that has no generalization or cell generalization approach. It also stores sorted records into their nearest slicedBuckets. The reason for almost straight graph for all the four approaches is because of the same generalization, bucketization and cell generalization during all the 20 releases in their respective algorithms.

Figure 5(b) compares τ -safe (l, k)-diversity and (τ, m) -slicedBucket algorithms to calculate NCP% for varying values of m , i.e. $m=4, m=6$ and $m=8$ during the 20 anonymized releases. The comparative graphs produced by both approaches have been encircled. As time evolves, closer records are achieved which results in small generalization. This reduces the information loss gradually. Though the loss is acceptable but for high parameter values i.e. l, k and m , more information loss and vice versa because of high generalization. The (τ, m) -slicedBucket outperforms the τ -safe (l, k)-diversity for each parameter during all the releases because of the negligible cell generalization in each sliced bucket and reduced dimensionality during slicing.

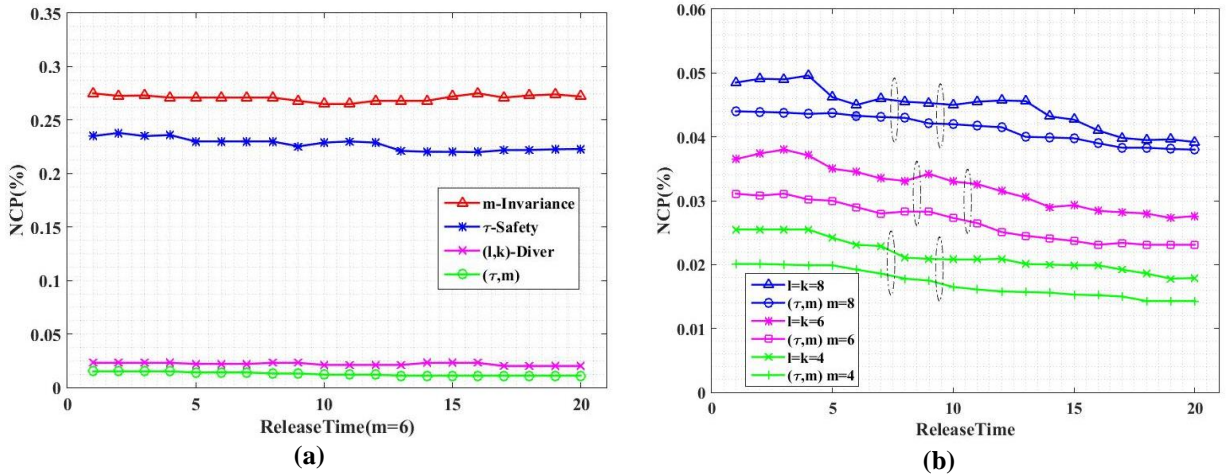


FIGURE 5 Utility measurements, (a) Inform. loss for fixed m , (b) Inform. loss of (l, k) and (τ, m) for varying m

(ii) KL-Divergence

Kullback-Leibler (KL)-Divergence [11] is also used to measure the utility loss when considering the overall distribution of QI attribute values in microdata. KL-Divergence is a logarithmic ratio of two probabilities. According to KL-Divergence, consider $P_1(t)$ and $P_2(t)$ as the probability distributions of a record in the original microdata and anonymized microdata, respectively. The KL-divergence for both tables is calculated in Equation (7) as follows.

$$KL(P_1, P_2) = \sum_t P_1(t) \log \frac{P_1(t)}{P_2(t)} \quad (7)$$

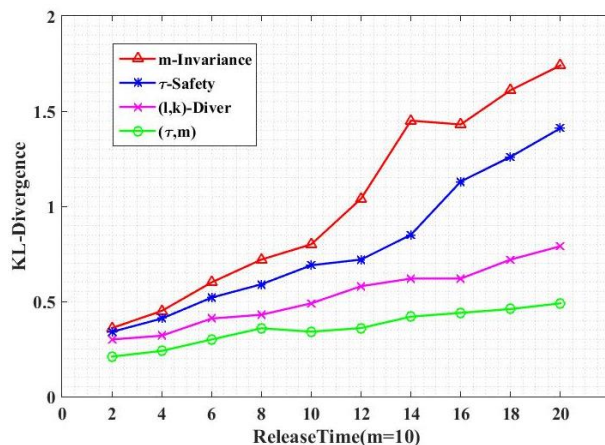


FIGURE 6 Utility measurements through KL-Divergence

Figure 6 depicts the utility through KL-Divergence. The algorithm executed for 20 releases having $m=10$ in size. During initial releases, the information loss is less while as time evolves the high graph shows more information loss. The three approaches i.e. m -invariance, τ -safety, (l, k) -diversity, adds the counterfeits to achieve the record signature in an EC. Adding the counterfeits repeatedly in each release reduces the truthfulness of data because of fake tuples. The delete list used by τ -safety improves its performance as compared to m -invariance and the same Del approach is used by τ -safe (l, k) -diversity also, which keeps it closer to τ -safety. The angelization [28] approach improves the τ -safe (l, k) -diversity utility from the τ -safety. For (τ, m) -slicedBucket privacy model no information loss with respect to counterfeit. Because of Cach table the (τ, m) -slicedBucket does not use any counterfeit tuples which enhances its utility a lot as discussed in the algorithm. For initial releases almost no loss while for higher releases the small increase in graph is because of the cell generalization used and high range values in generalization.

6.2 Query Accuracy

Query accuracy [11-12] evaluates utility of the anonymized release by triggering aggregate queries. The anonymized release T^* from original microdata T having q as maximum A^{QI} 's i.e. $A_1^{QI}, A_2^{QI}, A_3^{QI}, \dots, A_q^{QI}$ where $D(A_i^{QI})$ is the domain of the i^{th} A^{QI} . Then the following aggregate query is used to calculate query accuracy.

Query = select count(*) from R^* where $A_1^{QI} \in D(A_1^{QI})$ AND . . . AND $A_q^{QI} \in D(A_q^{QI})$

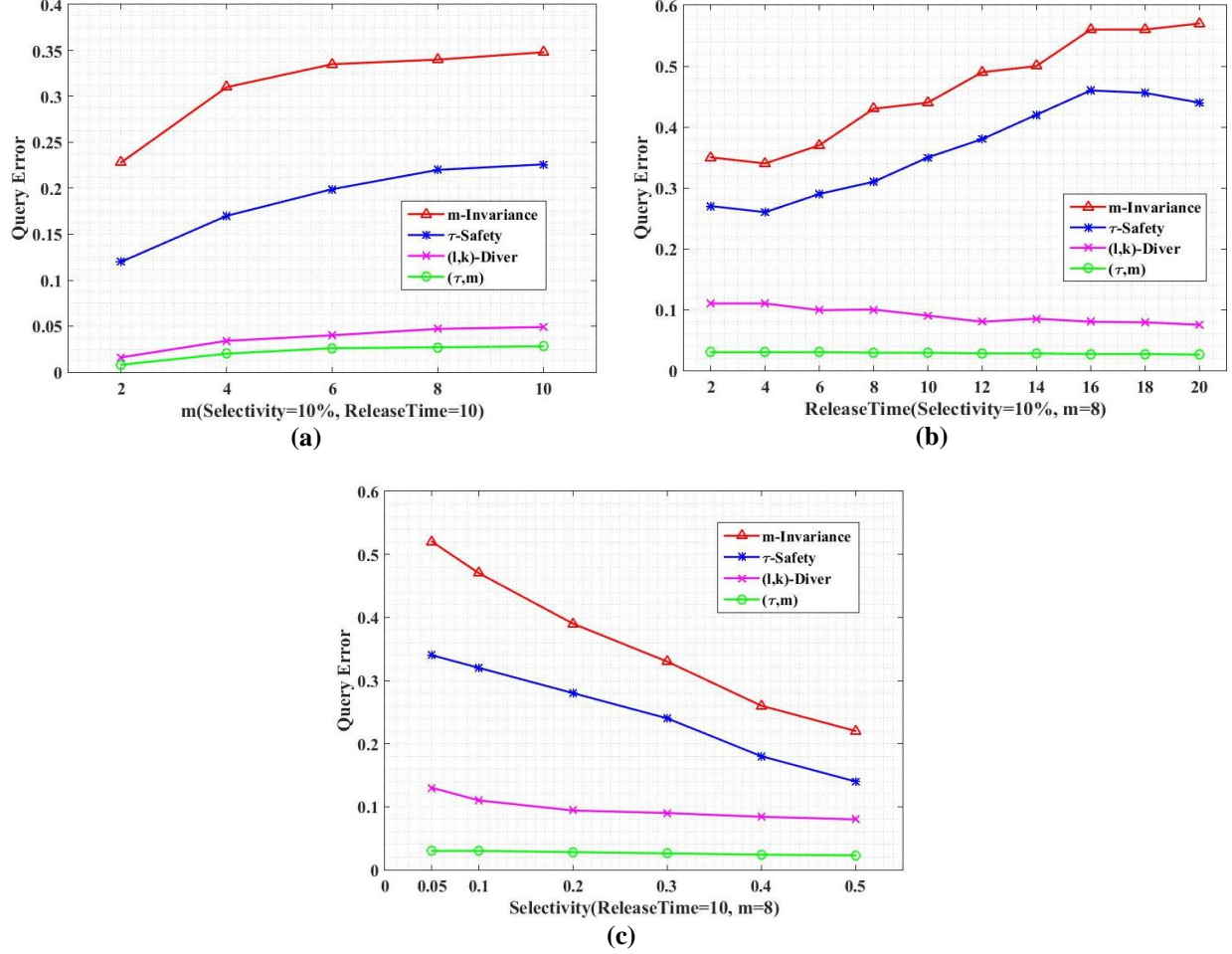


FIGURE 7 Query Error (a) Variable m size (b) Updating Release time (c) Change in Selectivity

Query dimensionality and query selectivity (number of records to be selected), are the two predicates in above query in which dimensionality is dependent on selectivity. If the number of tuples obtained after applying query are $|T_{Query}|$ on T where $|T|$ are the total number of records in the dataset, the query selectivity is $\theta = \frac{|T_{Query}|}{|T|}$. The query error in Equation (8) is then a difference between the anonymized and original dataset.

$$\text{QueryError} = \frac{|\text{Count}(\text{anonymized}) - \text{Count}(\text{original})|}{\text{Count}(\text{original})} \quad (8)$$

where $\text{count}(\text{anonymized})$ is the output from T^* and $\text{count}(\text{original})$ is the result from T using the aggregate operator COUNT. Queries with higher selectivity (more predicates) will have higher error rate. The query error was checked under different parameters i.e. varying m , varying time and varying selectivity. We executed 1,000 random queries for m -invariance, τ -safety, τ -safe (l, k) -diversity and for our proposed (τ, m) -slicedBucket privacy models. On average, the query errors for different values of m , and for a specific release e.g. 10th release, can be seen in Figure 7(a) at 10% selectivity. High value of m means high generalization of A^{Ql} 's. The proposed (τ, m) -slicedBucket uses cell generalization [14-15] while the remaining three approaches uses the same classical generalization that leads to their low utility comparatively. In Figure 7(b), the query error for a specific value of $m=8$ over different releases, τ -safety performs well as compared to m -invariance. The smooth increase in τ -safety is because of the new records insertions on minimum distance basis which reduces the interval of A^{Ql} 's over different releases. In the proposed (τ, m) -slicedBucket model because of the reduced dimensionality in A^{Ql} 's the query error is further reduced with the same minimum distance A^{Ql} placement in an EC. Figure 7(c) depicts query error with respect to varying selectivity. High selective queries have more predicates which results in a smaller number of records. So, query error increases for more selective queries and high selectivity results in decrease query error. For release 10th and $m=8$, (τ, m) -slicedBucket

has the lowest query error as compared to m -invariance, τ -safety, and τ -safe (l,k) -diversity, even for low selectivity value.

6.3 Counterfeit avoidance

Number of counterfeits added to the published anonymized release is a key factor in decreasing the truthfulness of QI values. As discussed in Section 1.1 (iii) and Section 5.3, m -invariance, τ -safety, and (l,k) -diversity are using counterfeits in all their releases while the proposed (τ, m) -slicedBucket Cach table helps to perform dynamic publishing without using any counterfeit tuple. Figure 8 depicts a zero-counterfeit tuple for the proposed (τ, m) -slicedBucket model while the remaining approaches randomly uses counterfeits. Counterfeit even creates more serious problem when the number of deletions is higher than the inserted tuples. This means that almost more than half of the data in a published release will be having no truthful QI values. Therefore, with high deletions in m -invariance, τ -safety, and τ -safe (l, k) -diversity will be having more utility loss and fake publication.

6.4 Execution time analysis

The computational efficiency of a model or an algorithm is expressed in terms of its total execution time. The algorithm executes for twenty releases with varying m in the range from 2 to 10. The average algorithm execution time obtained for different values of m is shown in Figure 9 that compares m -invariance, τ -safety, τ -safe (l, k) -diversity and (τ, m) -slicedBucket algorithms. Quicker buckets partitioning makes the τ -safety better than m -invariance in execution but high value of m responds in a slight increase in execution. The average execution time for τ -safe (l, k) -diversity has very small increase even if m increases. This is because τ -safe (l, k) -diversity only performs sorting and generalization. Although managing high size of Del table is time consuming for τ -safe (l, k) -diversity but still takes less time in execution as compared to (τ, m) -slicedBucket. The proposed (τ, m) -slicedBucket performs well as compared to m -invariance and τ -safety but has high execution time in comparison to τ -safe (l, k) -diversity.

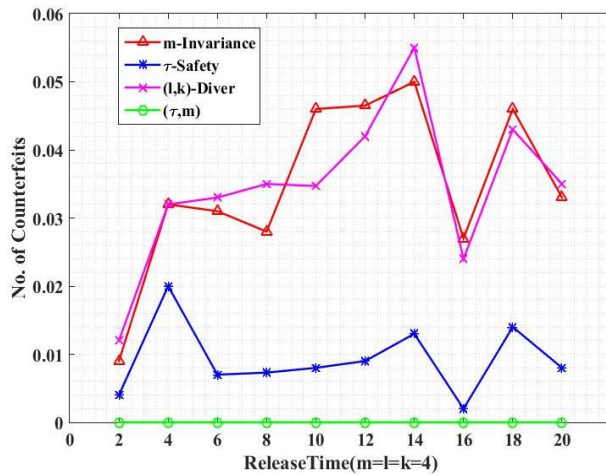


FIGURE 8 Counterfeit used

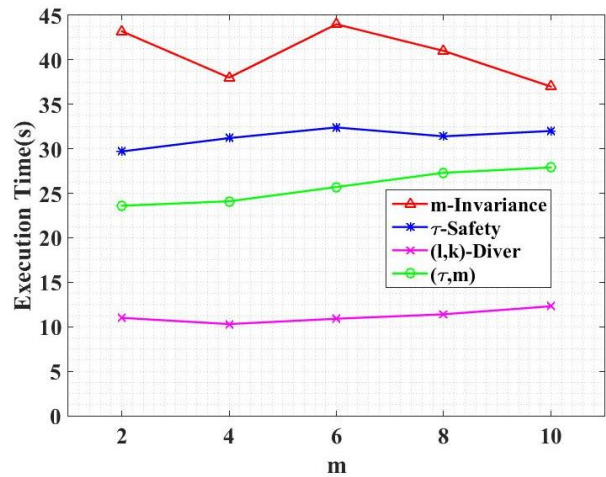


FIGURE 9 Algorithm Execution Time

Although cell generalization in (τ, m) -slicedBucket takes small amount of time however handling the Cach and Upd tables to avoid the counterfeits are the factors for extra time consumption.. The graph has a slight increase in execution time for the proposed work as m increases because more records needs to anonymize in an EC while at the same time contacting Cach and Upd tables needs to be done.

7 Conclusion

Privacy preserving data publishing has become a significant research area since the last decade. In this paper, a real-world challenging scenario for implementing privacy in sequential data publishing has been performed. The proposed (τ, m) -slicedBucket privacy model is a sequential anonymization model over time, which considers all possible operations i.e. insert, update, delete, on given dataset. The proposed algorithm follows certain steps to implement privacy and utility. For implementing privacy; attributes correlation, classification, balancing and assignments were

performed, while for utility enhancement; partition and cell generalizations are used. The Cach table creation helps to publish real anonymized records without using any counterfeit tuples. The counterpart privacy models i.e. m -invariance, τ -safety, τ -safe (l, k)-diversity were found vulnerable to our identified attack; collusion attack, and also have the counterfeit usage limitations. The formal modeling for τ -safe (l, k)-diversity identifies the collusion attack vulnerability which has been mitigated in the proposed privacy model formal modeling. The proposed (τ, m) -slicedBucket algorithm creates consistent signatures for a specific record respondent during all its releases. The experimental results proved the effectiveness for the proposed privacy model.

For future work, we consider the sequential dynamic data publication with progressive internal updates instead of arbitrary updates. Three other research directions can be (i) sequential dynamic data publication with multiple sensitive attributes (MSA) [49], (ii) sequential dynamic data publication with individual having more than one records i.e. 1:M microdata [29], and the more challenging work is to propose a solution for (iii) 1:M-MSA with sequential dynamic data publishing.

CONFLICT OF INTERESTS

The author declares that there is no conflict of interest that could be perceived as prejudicing the impartiality of the research reported.

DATA AVAILABILITY STATEMENT

The data used to support the findings of this study are available upon request from the corresponding author.

ORCID

Razaullah Khan: <https://orcid.org/0000-0002-4144-050X>

References

- [1] Parvaneh Asghari, Amir Masoud Rahmani, Hamid Haj Seyyed Javadi, "A medical monitoring scheme and health-medical service composition model in cloud-based IoT platform," *Transactions on Emerging Telecommunications Technologies*. vol. 30, no. 6, 2019. <https://doi.org/10.1002/ett.3637>
- [2] Chen, Chin-Ling & Yang, Tsai-Tung & Deng, Yong-Yuan & Chen, Chi-Hua. "A secure Internet of Things medical information sharing and emergency notification system based on nonrepudiation mechanism," *Transactions on Emerging Telecommunications Technologies*. Early access, 2020. <https://doi.org/10.1002/ett.3946>
- [3] Azad, Muhammad Ajmal, Arshad, Junaid, Mahmoud, Shazia, Salah, Khaled and Imran, Muhammad, "A privacy-preserving framework for smart context-aware healthcare applications," *Transactions on Emerging Telecommunications Technologies*, Early access, 2019. <https://doi.org/10.1002/ett.3634>
- [4] "Largest healthcare data breaches of 2019," <https://www.hipaajournal.com/2019-healthcare-data-breach-report/>, (accessed on 27th April 2020).
- [5] "Healthcare breaches cost \$6.2 billion annually," <https://www.beckershospitalreview.com/healthcare-information-technology/healthcare-breaches-cost-6-2b-annually.html>, (accessed on 27th April 2020).
- [6] L. Sweeney, " k -anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowledge-based Syst.*, vol. 10, no. 5, pp. 1–14, Oct. 2002.
- [7] L. Sweeney, "Achieving k -anonymity privacy protection using generalization and suppression," *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, vol. 10, no. 5, pp. 1–18, Oct. 2002.
- [8] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, " l -Diversity: Privacy Beyond k -Anonymity," *ACM Trans. Knowl. Discov. from Data*, vol. 1, no. 1, pp. 1-52, Mar. 2007.
- [9] Ninghui Li, Tiancheng Li, Venkatasubramanian, " t -Closeness: Privacy beyond k -anonymity and l -diversity," in *IEEE 23rd Int. Conf. Data Eng.*, Istanbul 2007, pp. 106-115.
- [10] X. Xiao and Y. Tao, " m -invariance: towards privacy preserving re-publication of dynamic datasets," in *Proceedings of the ACM SIGMOD International Conference on Management of data*, 2007, pp. 689–700.
- [11] A. Anjum et al., " τ -safety: A privacy model for sequential publication with arbitrary updates," *Computer and Security*, vol. 66, pp. 20-39, 2017.

- [12] Hui Zhu et al., " τ -Safe (l,k) -Diversity Privacy Model for Sequential Publication With High Utility," *IEEE Access*, vol. 7, pp. 687-701, 2019.
- [13] T. Li, N. Li, J. Zhang, and L. Molloy, "Slicing: A new approach for privacy preserving data publishing," *IEEE Transaction Knowledge data Eng.*, vol. 24, no. 3, pp. 561-574, Mar. 2012.
- [14] A S M Touhidul Hasan et al. "A New Approach to Privacy-Preserving Multiple Independent Data Publishing," *Applied Sciences*, vol. 8, no. 783, 2018.
- [15] Erez Shmueli, Tamir Tassa, Raz Wasserstein, "Limiting disclosure of sensitive data in sequential releases of databases," *Information Sciences*, vol. 191, pp. 98-127, 2012.
- [16] D. Riboni, L. Pareschi, C. Bettini, "JS-reduce: defending your data from sequential background knowledge attacks," *IEEE Trans. Dep. Sec. Comp.* vol. 9, no.3, pp. 387-400, 2012.
- [17] H. Liang, J. Wu, S. Mumtaz, J. Li, X. Lin and M. Wen, "MBID: Micro-Blockchain-Based Geographical Dynamic Intrusion Detection for V2X," in *IEEE Communications Magazine*, vol. 57, no. 10, pp. 77-83, Oct. 2019.
- [18] J. Chen et al., "Collaborative Trust Blockchain Based Unbiased Control Transfer Mechanism for Industrial Automation," in *IEEE Transactions on Industry Applications*, 2019, pp. 1-1, doi: 10.1109/TIA.2019.2959550.
- [19] J. Domingo-Ferrer and J. Soria-Comas, "From t -closeness to differential privacy and vice versa in data anonymization," *Knowledge-Based Systems*, vol. 74, pp. 151-158, 2015.
- [20] J. Cao, P. Karras, "Publishing microdata with a robust privacy guarantee," in *Proceedings of the 38th International Conference on Very Large Data Bases(VLDB)*, VLDB Endowment, vol. 5, no. 11, pp. 1388-1399, 2012.
- [21] Razaullah Khan, Xiaofeng Tao, Adeel Anjum et. al., " θ -Sensitive k -Anonymity: An Anonymization Model for IoT based Electronic Health Records," *Electronics*, Early access, vol. 9, no. 5, 2020. <https://doi.org/10.3390/electronics9050716> (registering DOI).
- [22] G. Cormode, D., Srivastava, E. Shen, "Aggregate Query Answering on Possibilistic Data with Cardinality Constraints," in *IEEE 28th International Conference on Data Engineering*, Washington, DC, 2012, pp. 258-269.
- [23] Shi Y, Zhang Z, Chao HC, "Data Privacy Protection Based on Micro Aggregation with Dynamic Sensitive Attribute Updating," *Sensors (Basel)*, vol. 18, no. 7, 2018.
- [24] Q. Zhang, N. Koudas, D. Srivastava and T. Yu, "Aggregate Query Answering on Anonymized Tables," in *IEEE 23rd International Conference on Data Engineering*, Istanbul, 2007, pp. 116-125.
- [25] J. Domingo-Ferrer and J. Soria-Comas, "Steered Microaggregation: A Unified Primitive for Anonymization of Data Sets and Data Streams," in *IEEE International Conference on Data Mining Workshops*, New Orleans, LA, 2017, pp. 995-1002.
- [26] J. Soria-Comas, J. Domingo-Ferrer, "Enhancing data utility in differential privacy via micro-aggregation based k -anonymity," *VLDB J.* vol. 23, no.5, pp. 771-794, 2014.
- [27] X. Xiao, Y. Tao, "Anatomy: simple and effective privacy preservation," in *Proceedings of the 32nd International Conference on Very Large Data Bases*, 2006, pp. 139-150.
- [28] Y. Tao, H. Chen, X. Xiao, S. Zhou, and D. Zhang, "Angel: enhancing the utility of generalization for privacy preserving publication," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 7, pp. 1073-1087, 2009.
- [29] Qiyuan Gong, Junzhou Luo, Ming Yang, "Anonymizing 1:m microdata with high utility," *Knowledge-Based Systems*, vol. 115, pp. 15-26, 2016.
- [30] C.E. Shannon, "Prediction and entropy of printed english," *Bell Syst. Tech. J.*, vol. 30, no. 1, pp. 50-64, 1951.
- [31] Saba Yaseen, Syed M. Ali Abbas, Adeel Anjum, "Improved Generalization for Secure Data Publishing," *IEEE Access*, vol. 6, pp. 27156-27165, 2018.
- [32] Fatemeh Amiri et al., "Hierarchical anonymization algorithms against background knowledge attack in data releasing," *Knowledge-Based Systems*, vol. 101, pp. 71-89, 2016.
- [33] F. Song, T. Ma, Y. Tian, "A New Method of Privacy Protection: Random k -Anonymous," *IEEE Access*, vol. 7, pp. 75434-75445, 2019.
- [34] B.C.M. Fung, K. Wang, P.S. Yu, "Top-down specialization for information and privacy preservation," in *Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE)*, 2005, pp. 205-216.
- [35] K. Wang, P.S. Yu, S. Chakraborty, "Bottom-up generalization: a data mining solution to privacy protection," in *Proceedings of the 4th IEEE International Conference on Data Mining (ICDM)*, 2004, pp. 249-256.
- [36] Barak, B., Chaudhuri, K., Dwork, K.C., Kale, S., Mcsherry, F. and Talwar, K., "Privacy, accuracy, and consistency too: A holistic solution to contingency table release," in *Proc. the 26th ACM Symposium on Principles of Database Systems (PODS)*, 2007, pp. 273-282.
- [37] E. Shmueli, T. Tassa, "Privacy by diversity in sequential releases of databases," *Information Sciences*, vol. 298, pp. 344-372, 2015.
- [38] K. Wang, B. Fung, "Anonymizing sequential release," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006, pp. 414-423.
- [39] Bu, Y., Fu, A.W.C., Wong, R.C.W., Chen, L. and Li, J., "Privacy preserving serial data publishing by role composition," in *Proc. the VLDB Endowment*, vol. 1, no. 1 pp.845-856, 2008.
- [40] Bouna, B.A.I., Clifton, C., Malluhi, Q., "Efficient sanitization of unsafe data correlations," in *Proc. the Workshops of the EDBT/ICDT Joint Conf*, 2015, pp. 278-285.
- [41] Li, T., Li, N., Zhang, J., "Modeling and integrating background knowledge in data anonymization," in *Proc. the 25th IEEE Int. Conf. on Data Engineering (ICDE)*, 2009, pp. 6-17.

- [42] Wang, H., Liu, R., "Privacy-preserving publishing microdata with full functional dependencies," *Data & knowl. Eng.*, vol. 70 pp. 249 – 268, 2011.
- [43] Byun JW., Sohn Y., Bertino E., "Secure Anonymization for Incremental Datasets, in Secure Data Management," *Lecture Notes in Computer Science*, vol. 4165, 2006.
- [44] Adeel Anjum et al., "An Efficient Privacy Mechanism for Electronic Health Records," *Computers & Security*, vol. 72, no. C, pp. 196-211, 2018.
- [45] S.U. Malik, S.U. Khan, "Modeling and analysis of state-of-the-art VM-based cloud management platforms," *IEEE Trans. Cloud Comput.* vol. 1, no. 1, pp. 1-1, 2013.
- [46] Tehsin Kanwal et. al., "Privacy-preserving model and generalization correlation attacks for 1:M data with multiple sensitive attributes," *Information Sciences*, vol. 488, pp. 238-256, 2019.
- [47] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy", *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211-407, 2014.
- [48] Li, N. Qardaji, W. Su. D., "Provably Private Data Anonymization: Or, k-Anonymity Meets Differential Privacy," available online: https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2010-24.pdf, (accessed on 11 February 2020).
- [49] Razaullah Khan, Xiaofeng Tao, Adeel Anjum, "Privacy Preserving for Multiple Sensitive Attributes against Fingerprint Correlation Attack Satisfying c -Diversity," *Wireless Comm. and Mob. Comp.*, available online: <https://www.hindawi.com/journals/wcmc/2020/8416823/>(accessed on 12 February 2020).