Received: 16 September 2019 | Revised: 1 July 2020 | Accepted: 26 August 2020

DOI: 10.1002/ett.4130

SPECIAL ISSUE ARTICLE

WILEY

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

Razaullah Khan[1] | Xiaofeng Tao[1] | Adeel Anjum[2,3] | Saifur Rehman Malik[4] | Shui Yu[5] | Abid Khan[6] | Waheedur Rehman[7] | Hassan Malik[8]

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

[2]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China

[3]Department of Computer Science, COMSATS University Islamabad, Islamabad, Pakistan

[4]Cybernetica AS Estonia, Tallinn, Estonia

[5]Faculty of Engineering and Information Technology, School of Software, University of Technology Sydney, Ultimo, Australia

[6]Department of Computer Science, Aberystwyth University, Aberystwyth, UK

[7]Department of Computer Science, University of Peshawar, Peshawar, Pakistan

[8]Department of Computer Science, Edge Hill University, Ormskirk, UK

Correspondence
Xiaofeng Tao, National Engineering Laboratory for Mobile Network Technologies, Beijing University of Posts and Telecommunications, Beijing, China.
Email: taoxf@bupt.edu.cn

Funding information
Beijing University of Posts and Telecommunications

Abstract

In a real-world scenario for privacy-preserving data publishing, the original data are anonymized and released periodically. Each release may vary in number of records due to insert, update, and delete operations. An intruder can combine, that is, 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 Furthermore, we formally model and analyze the proposed model to show that the *collusion attack* is no longer applicable.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.

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), and so on.[1-3] Such data may contain private information, for example, name, address, account number, patient disease information in EHR, and so on. The leakage of

[Correction added on 13 November 2020, after first online publication: The author name has been corrected to Waheedur Rehman, in Table 8, rows 9 and 10 have been corrected, and the acknowledgement section has been added.]

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 aim 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 are 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 References 11,12 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 (ie, 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} (eg, social security number or an individual name), quasi identifier attributes - A^{QI} (eg, age, gender, zipcode), and sensitive attribute (SA) - A^{S} (eg, disease.) For data de-identification, removing the A^{ID} is not enough because an adversary (ie, an attacker) can use background knowledge (BK), that is, certain pattern of A^{QI} (partial identifiers) and A^{S},[16] and can link the published data to some external data (eg, voting system) to re-identify an individual, named as linking attack. Therefore, the original microdata T is k-anonymized[1] to T^* (eg, 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 1) 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 that is, in Table 1C, is used to perform anonymization through τ-safe (l,k)-diversity[12] and the proposed (τ, m)-slicedBucked privacy models.

The PPDDP can support records insertion, deletion or update operations during different releases (ie, $T_1^*, T_2^*, T_3^*, \ldots\ T_n^*$). 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 that is, . A^{QI} or A^{S} values, over different times. In this paper, we consider the internal updates for A^{S} values and are arbitrary that is, old values have not been correlated with the new ones. An intruder can identify an individual record by correlating different releases, that is, 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

TABLE 1 (A) Microdata table T (before Cach), (B) Cach table, (C) microdata T (after Cach)

Age	Gender	Zipcode	Disease
(A)			
22	M	47 906	Aids
22	F	47 305	Flu
33	F	47 905	SSM-pos
52	F	47 905	Asthma
54	M	47 906	Flu
60	M	47 302	Cardiac
60	M	47 304	Dyspepsia
64	F	47 304	Gastritis
21	M	47 901	Bronchitis
54	F	47 902	Dyspepsia
27	M	47 303	Aids
65	M	47 308	Cardiac
58	F	47 905	Asthma
64	F	47 308	Gastritis
21	M	*47 907*	Bronchitis
(B)			
22	M	47 906	Aids
22	F	47 305	Flu
33	F	47 905	SSM-pos
52	F	47 905	Asthma
60	M	47 302	Cardiac
60	M	47 304	Dyspepsia
64	F	47 304	Gastritis
21	M	47 901	Bronchitis
52	F	47 907	Malaria
32	M	47 955	Diarrhea
52	M	47 915	Pneumonia
(C)			
21	M	47 907	Bronchitis
27	M	47 303	Aids
33	F	47 905	SSM-pos
58	F	47 905	Asthma
54	M	47 906	Flu
54	F	47 902	Dyspepsia
64	F	47 308	Gastritis
65	M	47 308	Cardiac

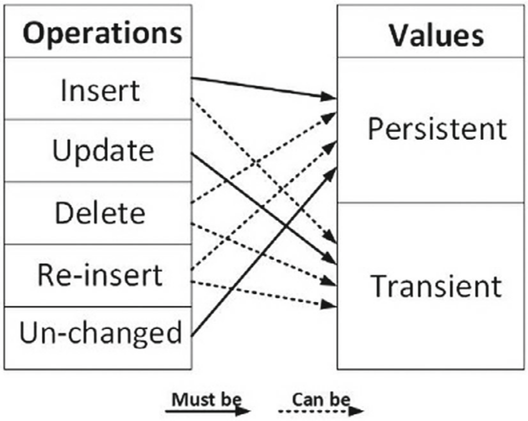


FIGURE 1 Relating values with the possible operations on them

TABLE 2 (A) τ-Safe (2,2)-diversity GT1 for T_1^*, (B) τ-safe (2,2)-diversity BT1 for T_1^*, (C) del table at time 1

(A)

GID (name)	Age	Zip code	BID
1(p1)	[21-27](21)	[47303-47 907](47907)	1
1(p2)	[21-27](27)	[47303-47 907](47303)	1
2(p3)	[33-58](33)	[47905-47 906](47905)	2
2(p4)	[33-58](58)	[47905-47 906](47905)	2

(B)

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

(C)

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

(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.

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

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, that is, . $T_n^* \neq T_{n-2}^*$, $T_{n-1}^* \neq T_{n-3}^*$ 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 for example, p3 = *SSM-pos* in Table 2. At time 2 in T_2^*, for example p3 has suffered from *Diarrhea*, so a new signature {*Glaucoma, Diarrhe*a} has been created. At time 3, if p3 suffers again with same disease value *SSM-pos*, its intersection with {*Glaucoma, Diarrhea*} is zero, that is, $S(t) \notin Sig(BT_{p-1}(t))$ 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 that is, 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 3 (A) τ-Safe (2,2)-diversity GT2 for T_3^*, (B) τ-safe (2,2)-diversity BT2 for T_3^*, (C) del table at time 3

(A)

GID (name)	Age	Zip code	BID
1(p1)	[21, 22](21)	[47906-47 907](47906)	1
1(C1)	[21, 22](22)	[47906-47 907](47907)	1
2(p3)	[33-58](33)	[47905-47 921](47905)	2
2(p5)	[33-58](58)	[47905-47 921](47921)	2

(B)

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

(C)

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

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^{QI} 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 High Level Petri-Net (HLPN). Section 5 presents 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 categorizes 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 for example, 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 are 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 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 are 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 that is, 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. Shmuely 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 $T = \{T_1, T_2, T_3, \ldots T_r\}$ 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. 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 for example, the publicly available voting data. The A^S, which contains sensitive information for example, disease in our case, needs to be protected the most. Table 4 depicts the notations used in this paper.

3.1 | Adversarial BK

The 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,

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	CβK	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.
$T^s = T_r \cap T_{r-1}$.
$T^n = (T_r - T_{r-1}) + T^u$, $t \notin$ Del, $t \notin$ Cach.
$T^r : \forall t \in T_r, t \notin T_{r-1,} t \in$ Del, $t \notin$ Upd.
$T^u : \forall t \in T_r, t \notin T_{r-1,} t \in$ Upd, $t \notin$ Del.
$T^d : \forall t \notin T_r, t \in T_{r-1,} t \in$ 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.

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.

$$PK_i^{sv} = BK_i^{sv} \bowtie UBK_{i+1}^{sv} \tag{1}$$

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

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^{sv} and UBK^{sv} in Equations (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: $C\beta K = (UBK^{sv}, PK_i^{sv}, PD)$

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^{sv} and UBK^{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 that is, 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 $T_1^*, T_2^*, T_3^*, \ldots T_r^*$ 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 $Sig(t_j) = Sig(t_{j+1}) = \ldots = Sig(t_{j+k})$, 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 | ($\mathrm{PK}_{\mathrm{i}}^{\mathrm{sv}} \wedge \mathrm{UBK}_{\mathrm{i+1}}^{\mathrm{sv}}$) where p(t) is the probability of tuple t and $\mathrm{PK}_{\mathrm{i}}^{\mathrm{sv}}$ and $\mathrm{UBK}_{\mathrm{i+1}}^{\mathrm{sv}}$ are obtained from Equations (1) and (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, that is, collusion attack. To perform the formal modeling 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 5. 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 are transferred to a trusted data-sanitizer. The data sanitizer performs an anonymization process over the raw data, seeking to minimize A^{S} values disclosure. Data are 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, that is, 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 Reference 12. The transition rule is given for adversarial collusion attack only, for the identification of inconsistence 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^{r}) 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-BktS, else transition Crt-OthrBkt for a new signature bucket. The reinserted tuples from Del table are processed via transition Crt-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 $\mathrm{T}^{\mathrm{CtS}}$ (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 Reference 11. This phase divides the records in the T^{n} into correct buckets. Transition addCt adds counterfeit to $\mathrm{T}^{\mathrm{new}}$ to make it *l*-eligible. The transition Chk asg-equ check the T^{n} via asg-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 ($\mathrm{T}^{\mathrm{CtQ}}$) in transition Initialize-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-BUC represent tables GT and BT respectively.

The inconsistent signatures created in classification phase allow the privacy breach. Section 1.1 motivation, illustrates 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-BUC. Inside the *Sign_Disc()* function the adversary correlate the sensitive knowledge of BA-BUC and UBK with QI values from GenData (Rule 1 depicts the Figure 2 collusion attack transition). The adversary correlates same record signatures in different releases by continuously using Equation (2) iteratively to get UBK, until it results in $\mathrm{Sign}_{\mathrm{dis}}$ and QI identification, against an individual.

$$\boldsymbol{R}\ (\ \textit{\textbf{Collusion Attacks}}\ ) = \forall\, i45 \in x45,\ i46 \in x46, \forall\, i48 \in x48,\ i49 \in x49 \mid$$
$$\mathit{Sign\_Disc}(\{i46[2], i48[2]\} \cup i45[2]) \rightarrow i49[2] = i1[2] \wedge i49[1] = i29[2]) \qquad \text{Rule 1}$$

TABLE 5 (A) Description of places and types used in HLPN for **τ**-safe (l, k)-diversity model, (B) mapping of data types on places

(A)	
Types	**Description**
T_{p-1}	Place holding sets of same records, signatures, and re-inserted records.
GID	An integer type describing group identifier.
PID	An integer type describing 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^{CtS}, 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
$T_{sort'}$	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

(B)	
Places	**Description**
$\varphi\ (T_{p-1})$	$\mathbb{P}\ (T^{s} \times \mathrm{Sign}_{same} \times T^{re})$
$\varphi\ (BT_{p-1})$	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{Sign}_{p-1} \times \mathrm{BID})$
φ (B)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{Sign}_{s} \times \mathrm{BID})$
φ (B′)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{Sign}_{ru} \times \mathrm{BID})$
φ (DEL)	$\mathbb{P}\ (\mathrm{PID} \times \mathrm{Sign}_{d} \times \mathrm{BID})$
φ (BUC′)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{Sign}_{all} \times \mathrm{BID})$
φ (BUC)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{Sign}_{m} \times \mathrm{BID})$
$\varphi\ (T^{n})$	$\mathbb{P}(\mathrm{PID} \times \mathrm{Sign}_{new} \times \mathrm{BID} \times \ell)$
$\varphi\ (T^{CtS})$	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{Sign}_{Ct} \times \mathrm{BID})$
φ (BA-BUC)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{Sign}_{BA} \times \mathrm{BID})$
$\varphi\ (T^{n'})$	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{Sign}_{new'} \times \mathrm{BID} \times T_{ct})$
φ (A-BUC)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{Sign}_{a} \times \mathrm{BID})$
$\varphi\ (T_{p})$	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{QI} \times \mathrm{BID})$
$\varphi\ (T^{CtQ})$	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{QI} \times \mathrm{BID})$
φ (T-Q)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{QI} \times \mathrm{BID} \times \lambda \times \mathrm{k})$
φ (asg-var)	$\mathbb{P}(\alpha \times \beta \times \gamma)$
φ (array)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{QI}_{sort} \times \mathrm{BID} \times \gamma\ )$
φ (GenArrData)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{QI}_{array} \times \mathrm{BID})$
$\varphi\ (T_{sort})$	$\mathbb{P}\ \mathrm{GID} \times \mathrm{QI}_{sort} \times \mathrm{BID} \times \gamma)$
$\varphi\ (T_{sort'})$	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{GQI} \times \mathrm{BID})$
φ (GenTData)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{QI}_{T} \times \mathrm{BID})$
φ (GenData)	$\mathbb{P}\ (\mathrm{GID} \times \mathrm{QI}_{F} \times \mathrm{BID})$
φ (UBK)	$\mathbb{P}\ (\mathrm{QI}_{adv} \times \mathrm{Sign}_{adv})$
φ (Sign Dis)	$\mathbb{P}\ (\mathrm{QI} \times \mathrm{Sign}_{dis})$

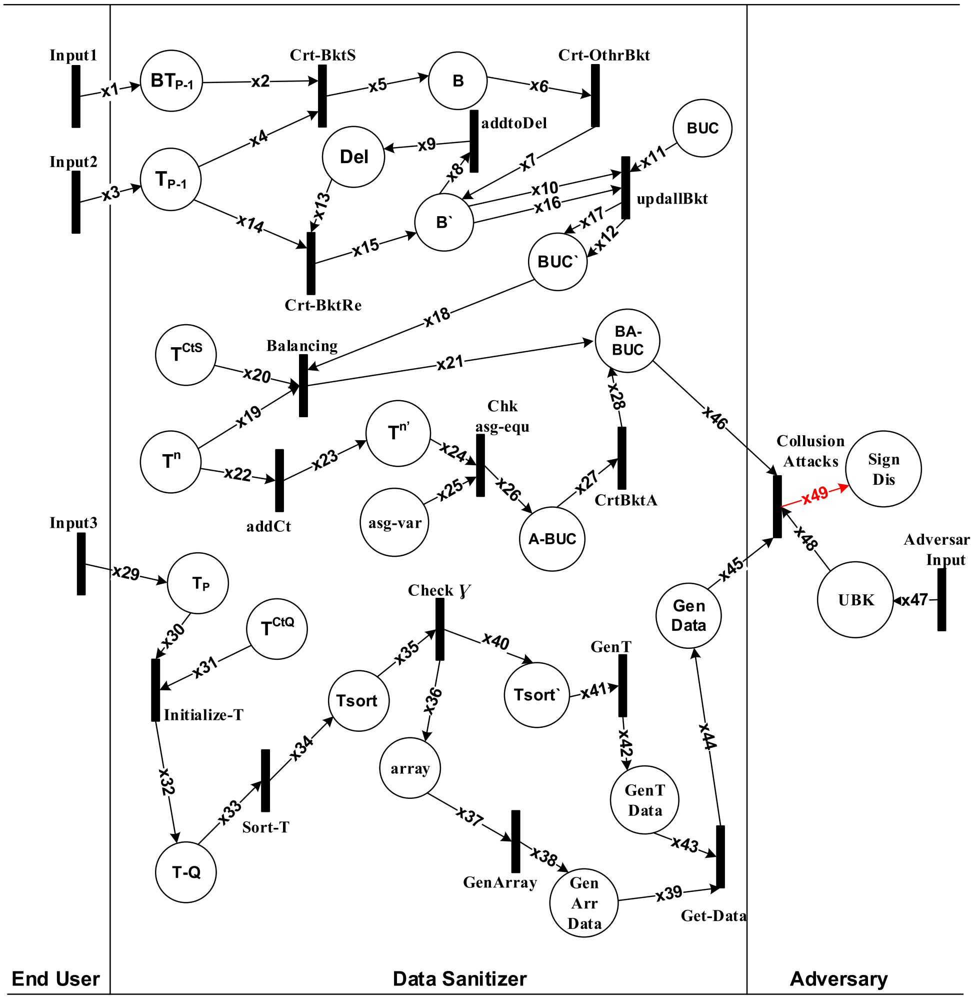


FIGURE 2 HLPN of τ-safe (l, k)-diversity model with adversarial attacks identification

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 $\mathrm{Sig}(T_r^*(t)) = \mathrm{Sig}(T_{r-1}^*(t)) = \mathrm{Sig}(T_{r-2}^*(t)) = \dots = \mathrm{Sig}(T_{r-n}^*(t))$, where $0 \leq i \leq n$ and if $\exists$ a partial intersection probability $\{0 < p < 1\}$ during any release then $\mathrm{Sig}(T_r^*(t)) \cap \mathrm{Sig}(T_{r-1}^*(t)) \cap \mathrm{Sig}(T_{r-2}^*(t)) \cap \dots \cap \mathrm{Sig}(T_{r-n}^*(t)) = \emptyset, \forall$ internal updates.
3. Shuffle AC^{QI} and A^S values inside the sliced buckets randomly in each anonymized release, that is, . $T_1^*, T_2^*, T_3^*, \dots T_r^*$ before publishing.

The three conditions for the proposed (τ, m)-slicedBucket privacy model guarantee privacy for the anonymized release. Condition (1) guarantee m-uniqueness for each k-anonymous SBUC over time. Condition (2) maintains consistent signature for the same record during its lifespan. This ensures protection from collusion attack. Condition (3) guarantees 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 detailed explanation for the proposed (τ, m)-slicedBucket privacy model is 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 improves 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 Equation (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.

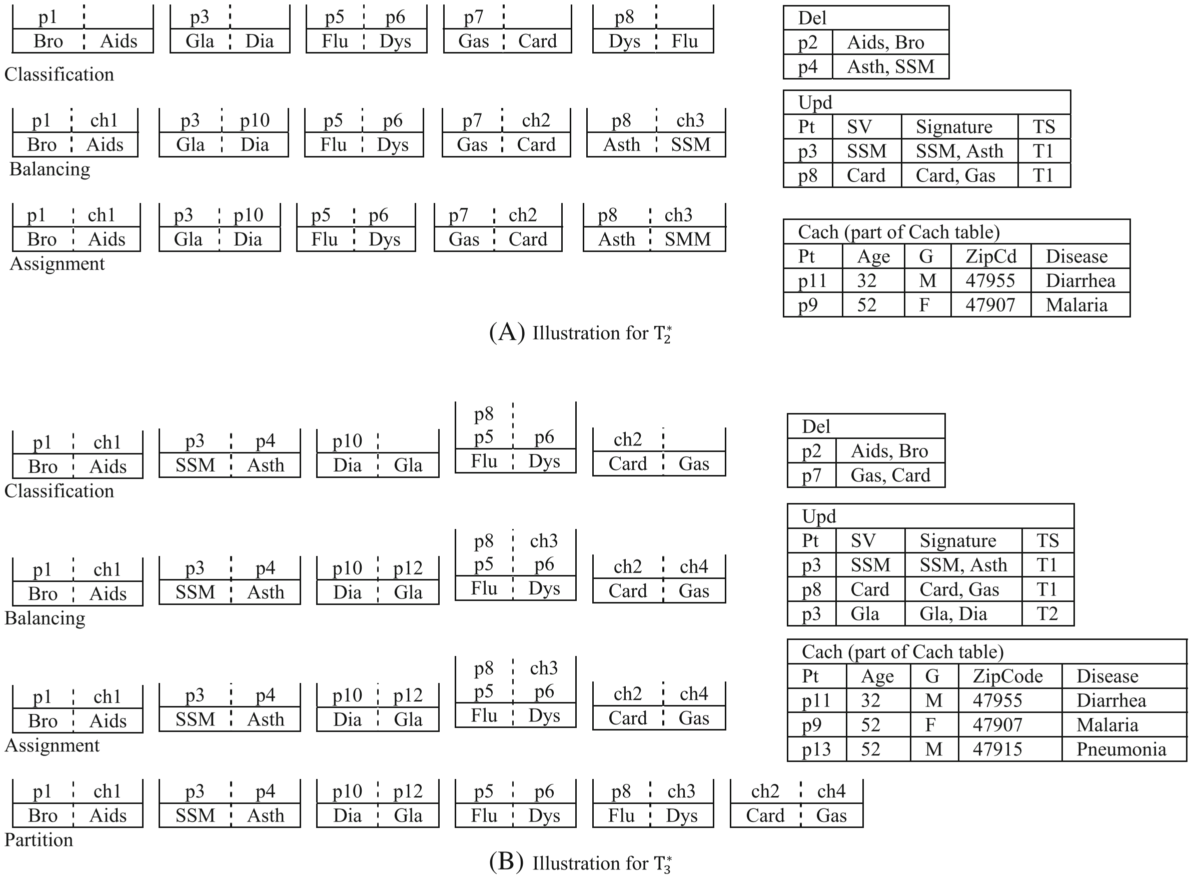
5.2 | (τ, m) Persistent invariance

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, $\mathrm{Sig}(T_r^*(t)) = \mathrm{Sig}(T_{r-1}^*(t)) = \mathrm{Sig}(T_{r-2}^*(t)) = \ldots = \mathrm{Sig}(T_{r-n}^*(t))$ and if $\exists$ a partial intersection probability $\{0 < p < 1\}$ during any release then $\mathrm{Sig}(T_r^*(t)) \cap \mathrm{Sig}(T_{r-1}^*(t)) \cap \mathrm{Sig}(T_{r-2}^*(t)) \cap \ldots \cap \mathrm{Sig}(T_{r-n}^*(t)) = \emptyset$. 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 3A,B) otherwise new signature will be created according to Definition 11.

Tables 6-8 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 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*

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)	47 905	SSM-pos
p4	(33-58, F)	47 905	Asthma
p5	(54, *) [M]	4790* [47906]	Flu
p6	(54, *) [F]	4790* [47902]	Dyspepsia
p7	(64-65, *) [64, F]	47 308	Gastritis
p8	(64–65, *) [65,M]	47 308	Cardiac



Del	
p2	Aids, Bro
p4	Asth, SSM

Upd			
Pt	SV	Signature	TS
p3	SSM	SSM, Asth	T1
p8	Card	Card, Gas	T1

Cach (part of Cach table)				
Pt	Age	G	ZipCd	Disease
p11	32	M	47955	Diarrhea
p9	52	F	47907	Malaria

(A) Illustration for T_2^*

Del	
p2	Aids, Bro
p7	Gas, Card

Upd			
Pt	SV	Signature	TS
p3	SSM	SSM, Asth	T1
p8	Card	Card, Gas	T1
p3	Gla	Gla, Dia	T2

Cach (part of Cach table)				
Pt	Age	G	ZipCode	Disease
p11	32	M	47955	Diarrhea
p9	52	F	47907	Malaria
p13	52	M	47915	Pneumonia

(B) Illustration for T_3^*

FIGURE 3 Processing of bucket data

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 that is, Del. Figure 3 illustrates the data processing inside the buckets during T_2^* and T_3^* and the data inside the three supporting tables that is, 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, that is, 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 counerfiets while the differential priacy[19,47] adds laplacian noise. We bridge the gap similar to Reference 48 of syntatic and semantic data publishing via Cach table (actual records) but in a dynamic data sequential

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]	47 905	SSM-pos
p4	(33-58, F) [58]	47 905	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

scenario where no counterfiet 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, that is, . 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 : $T_r, T^*_{r-1}$, Cach, Del, Upd
1.Calculate Cach table : $T_{rep}$ and $T_{so}$
2. Calculate correlating attributes : chi-square method
3.Calculate initially and after each release :
$T^n = T_r - T_{r-1}, T^s = T_r \cap T_{r-1}, T^u = Sig(t_r) \neq Sig(t_{r-i})$
4.Reinsertion : $T^r = Sig(Del(t))$
5.SBUC = Classify($T^s, T^*_{r-i}, T^u, T^r$, 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 60 000 records that have only 49 SA 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 (e, δ)-differential privacy[48] and provides strong privacy guarantee. Table 1B is an example of Cach table obtained from original Table 1 before its anonymization. Figure 3A,B also illustrates the Cach table during T^*_2 and T^*_3 creation.

Algorithm 2. Cach creation

```
for all records in Dataset do
    $T_{rep}$ ← Comp rep(SA)
    Cach ← single_rec_frm($T_{rep}$)
    $T_{so}$ ← Comp single_occured(SA)
    Cach ← copy_of($T_{so}$)
endfor
return Cach
```

(ii) Grouping Correlated Attributes: Generalization, that is, *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 that is, 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 Equation (3).

Algorithm 3. Calculate co-related attributes

```
for all attributes in microdata T do
    Calc – affinity ($A_i \varsigma A_j$) 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 $\{v_{11}, v_{12}, \ldots v_{1d_1}\}$ and $\{v_{21}, v_{22}, \ldots v_{2d_2}\}$ 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).

$$\emptyset^2(A_1, A_2) = \frac{1}{\min(d_1, d_2) - 1} \sum_{i=1}^{d_1} \sum_{j=1}^{d2} \frac{(f_{i\,j} - f_{i.} f_{.j})^2}{f_{i.} f_{.j}} \tag{3}$$

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 $0 \le \emptyset^2(A_1, A_2) \le 1$.

In this work we adopt the same attributes correlation approach as in Reference 9 The attributes *{age, gender}* have high affinity and form the first column, that is, . AC^{QI}. The zipcode and disease are considered as separated attributes. For attributes, age and gender, affinity example can be seen in Tables 6-8.

(iii) Classification: This phase is the main source of signatures inconsistency, which is not modeled properly in Reference 11 and in Reference 12 In this phase the SBUCs are created and classify the records, that is, . $t_1, t_2, t_3, \ldots, t_n$ in T^s, T^u and T^r into their correct buckets using T_{r-1} $(1 \le j \le r)$, 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 exist in Upd or Del tables then the tuples are stored in temporary arrays as T^{un}_i (updated new records) or T^{rn}_i (re-inserted new records), respectively and are partially treated as new tuples. The new signatures for new buckets for records in T^{un}_i or T^{rn}_i are created based on zero integration condition (see Definition 11) and Algorithm 6 assignment condition. Because the work in Reference 11 or 12 are not sure about the bucket size for the records in T^{un} or T^{rn} which can be β or β+1, where β=*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, that is, . $T_{r-2}, T_{r-3}, \ldots, T_{r-n} = \bigcap_{r-2}^{r-n} Sig(t) = 0$ and checked via assignment condition in Algorithm 6. The classification algorithm in Reference 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, that is, *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 ($S(t) \in Sig(ST_{r-1}(t))$) then
        SB = Crt_SBkt($Sig(ST_{r-1}(t))$)
    end if
    put(SB, t)
    if ($SB \notin SBUC$)  then
        $SBUC = SBUC \cup \{SB\}$
    end if
end for
```

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

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

```
for all records in $T^n$ do
    $T_{rep} \leftarrow$ Comp rep(SA)
    Cach $\leftarrow$ single_rec_frm($T_{rep}$)
    $T_{so} \leftarrow$ Comp single_occured(SA)
    Cach $\leftarrow$ 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
```

do not simply add counterfeit to balance the SBUCs as in References 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 (e, 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 Section 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 (eg, 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 Section 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 References 11,12 to assign the remaining T^n into their correct SBUCs. After completion of Algorithm 6, all the remaining T^n records if exist 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 Reference 10 the variables α and β are computed, to assign the correct number of records and to handle the SBUC signatures, respectively. Let C = ($sv_i, sv_2, sv_3, \ldots, sv_\lambda$) 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, $sv_i (1 \le i \le \lambda)$ 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 *(*$sv_i, sv_2, sv_3, \ldots, sv_\beta$*)*. 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_i, sv_2, $sv_3, \ldots, sv_\beta$ 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 $\alpha \le sv_\beta$ and $sv_1 - \alpha \le \frac{(\gamma-\alpha.\beta)}{m}$ and $sv_{\beta+1} \le \frac{(\gamma-\alpha.\beta)}{m}$ 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 3A,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.

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

```
initialize $\lambda$ = Total distinct sensitive attribute values in $T^n$
if $\left( \frac{T^n}{m} \ngeq m \right)$ 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 $C = (sv_i, sv_2, sv_3, \ldots, sv_\lambda)$, i.e.$sv_i (1 \leq i \leq \lambda)$
    $\beta = m$
    $\alpha$ = largest positive integer that satisfy the ineqalities below
    if!($\alpha \leq sv_\beta$ and $sv_1 - \alpha \leq \frac{(\gamma - \alpha.\beta)}{m}$ and $sv_{\beta+1} \leq \frac{(\gamma - \alpha.\beta)}{m}$ then
        $\beta = \beta + 1$
        go to line $\alpha$ calculation above
end if
Create – Bucket SB having $(Sig(SB) = (sv_i, sv_2, sv_3, \ldots, sv_\beta)$
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, that is, $s \geq m$ sensitive values. The extra s tuples are removed from SBUCs by splitting or partitioning the SBUCs into balanced new SBUCs that is, . $SBUC_{new}$ 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.

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

```
$S_{sbuc} = \{SBUC\}$
while $(SBUC > s)$, where $s \geq m$
    $\frac{SBUC}{s}$ and
    Crt_new_SBkt($Sig(SBUC_{new}) = Sig(SBUC)$,
    with min. Interval w.r.t. $A_i^{QI}$
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, $(1 \le i \le c)$, a slicedBucket is B_b, $(1 \le j \le b)$, 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.

a. Age will be generalized to any interval in the range [a, b] where $a, b \in A_1$ and $a \le b$.
b. Gender can be M or F, therefore it is generalized by suppression only, that is, the original value is retained or it is suppressed.
c. 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.
d. Disease is a SA and no generalization is allowed.

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

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 $\mathbf{T_r^*}$
```

After generalization, attribute values on both sides, that is, AC^{QI} and A^s values are randomly permuted in each SBUC to break the linking between the uncorrelated attributes, that is, . 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 modeling 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 broadly consists of three entities: end user, trusted data sanitizer, and adversary. There are 27 Places (P) and 22 Transitions (T) involved in the modeling process. The P description with variable type are described in Table 9 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.

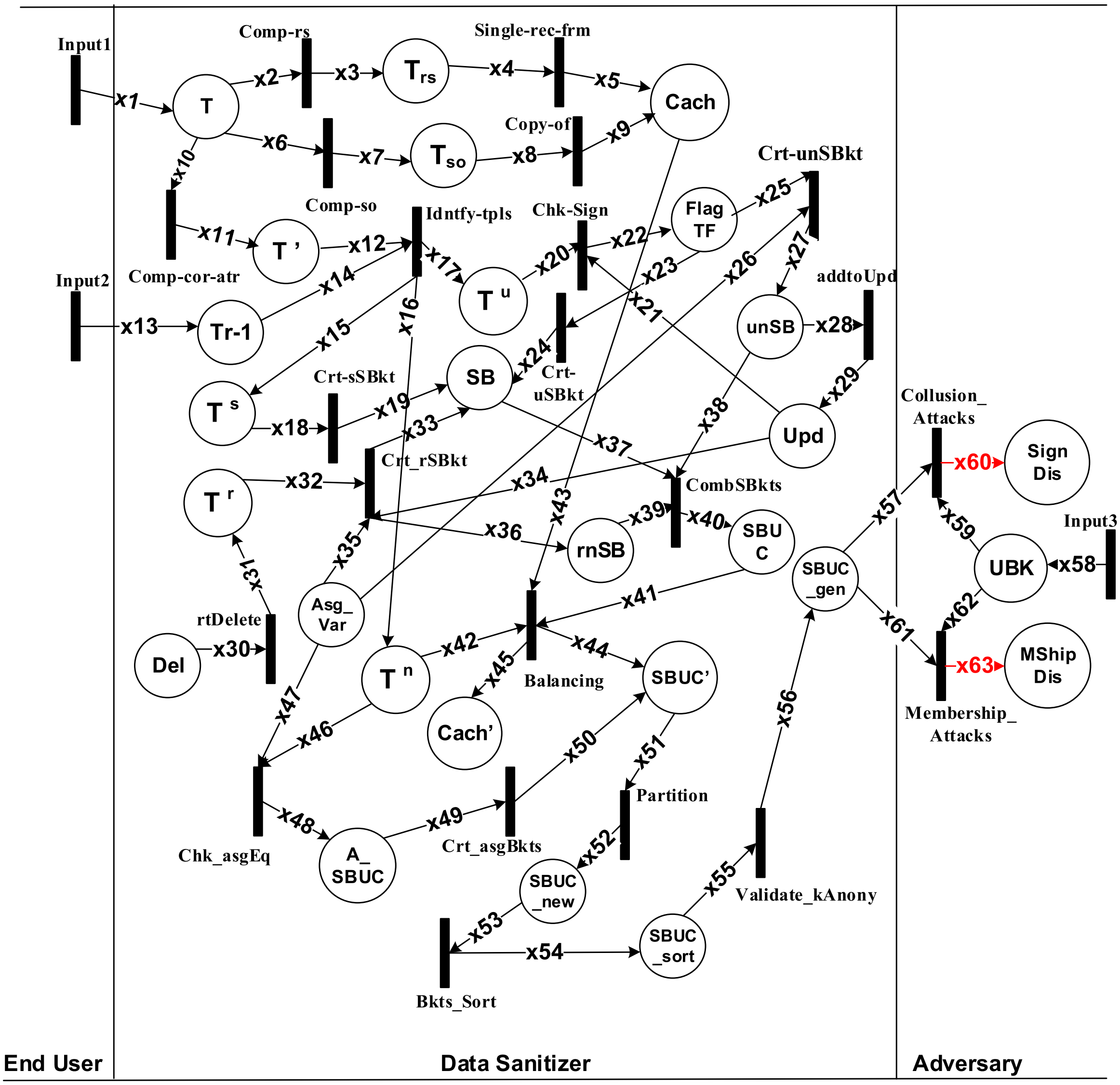


FIGURE 4 HLPN of (τ, m)-slicedBucket Privacy model with adversarial attacks invalidation

$$\boldsymbol{R}(\text{Comp} - \boldsymbol{rs}) = \forall i2 \in x2, i3 \in x3|$$
$$i3[1] := i2[1] \wedge i3[2] := i2[2] \wedge (i3[3] := Comprs(i2[3])) \wedge x3' := x3 \cup \{(i3[1], i3[2], i3[3])\} \qquad \text{Rule 2}$$

$$\boldsymbol{R}(\text{Single} - \boldsymbol{rec} - \boldsymbol{frm}) = \forall i4 \in x4, i5 \in x5|$$
$$i5[1] := i4[1] \wedge i5[2] := i4[2] \wedge (i5[3] := srfrm(i4[3]))$$
$$x5' := x5 \cup \{i5[1], i5[2], i5[3] \qquad \text{Rule 3}$$

$$\boldsymbol{R}(\text{Comp} - \boldsymbol{so}) = \forall i6 \in x6, i7 \in x7|$$
$$i7[1] := i6[1] \wedge i7[2] := i6[2] \wedge (i7[3] := Compso(i6[3])) \wedge x7' := x7 \cup \{(i7[1], i7[2], i7[3])\} \qquad \text{Rule 4}$$

$$\boldsymbol{R}(\text{Copy} - \text{of}) = \forall i8 \in x8, i9 \in x9|$$
$$i9[1] := i8[1] \wedge i9[2] := i8[2] \wedge (i9[3] := Copyof(i8[3])) \wedge x9' := x9 \cup \{(i9[1], i9[2], i9[3])\} \qquad \text{Rule 5}$$

TABLE 9 (A) Description of places and types used in HLPN for (**τ**,m)-slicedBucket model, (B) mapping of data types on places

(A)

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
$Sign_s$, $Sign_u$, $Sign_n$	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^r during classification phase
unSB	Holding record signatures for T^u 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

(B)

Places	Description
φ (T)	$\mathbb{P}$ (PID × QID × SAs)
φ (T_{rs})	$\mathbb{P}$ (PID × QID × SA_{ms})
φ (T_{so})	$\mathbb{P}$ (PID × QID × SA_{so})
φ (Cach)	$\mathbb{P}$ (PID × QID × SAs)
φ (T′)	$\mathbb{P}$ (PID × QID_{cor} × SAs)
φ (T_{r-1})	$\mathbb{P}$ (QI × $Sign_{r-1}$)
φ (T^s)	$\mathbb{P}$ (PID × QID × SA_{sign})
φ (T^r)	$\mathbb{P}$ (PID × QID × SA_{dSign})
φ (T^u)	$\mathbb{P}$ (PID × QID × $SA_{updSign}$)
φ (T^n)	$\mathbb{P}$ (PID × QID × SAs)
φ (Del)	$\mathbb{P}$ (PID × QID × $Sign_{r-1}$)
φ (Upd)	$\mathbb{P}$ (PID × QID × $Sign_u$)
φ (Flag)	$\mathbb{P}$ (ConditionTF)
φ (SB)	$\mathbb{P}$(PID × QI × $Sign_s$ × $Sign_u$ × $Sign_r$ × BID)
φ (rnSB)	$\mathbb{P}$ (PID × QI × $Sign_{rnc}$ × BID)
φ (unSB)	$\mathbb{P}$ (PID × QI × $Sign_{unc}$ × BID)
φ (SBUC)	$\mathbb{P}$(PID × QI × $Sign_{all}$ × BID)
φ (SBUC′)	$\mathbb{P}$(PID × QID × $Sign_{bal\,\&\,n}$ × BID)
φ (Asg_Var)	$\mathbb{P}$ ($\alpha \times \beta \times \gamma$)
φ (A_SBUC)	$\mathbb{P}$ (PID × QI × $Sign_n$ × BID)
φ (SBUC_new)	$\mathbb{P}$ (PID × QI × $Sign_{PRT}$ × BID × s × m)
φ (SBUC_sort)	$\mathbb{P}$ (PID × QI_{sort} × $Sign_{PRT}$ × BID)
φ (SBUC_gen)	$\mathbb{P}$(QI_{kAnony} × $Sign_{PRT}$)
φ (UBK)	$\mathbb{P}$ (QI_{adv} × $Sign_{adv}$)
φ (Sign Dis)	$\mathbb{P}$ (QI × $Sign_{disclosure}$)
φ (MShip Dis)	$\mathbb{P}$ ($QI_{disclosure}$)

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

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

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

$$\begin{aligned}
&\boldsymbol{R}(\text{Idntfy} - \text{tpls}) = \forall i12 \in x12, i14 \in x14, i15 x15, i16 \in x16, i17 \in x17 \mid\\
&i15[1] := i12[1] \wedge (i12[2] = i14[1]) \rightarrow (i15[2] := i12[2]) \wedge (i12[3] = i14[2]) \rightarrow (i15[3] := i12[3])\\
&x15' := x15 \cup \{(i15[1], i15[2], i15[3])\}\\
&i16[1] := i12[1] \wedge (i16[2] \neq (i12[2], \exists i12[2] \notin i14[1])) \wedge (i16[3] \neq (i12[3], \exists(\ i12[3] \notin i14[2])))\\
&x16' := x16 \cup \{(i16[1], i16[2], i16[3])\}\\
&i17[1] := i12[1] \wedge (i17[2] := (i12[2], \exists i12[2] \in i14[1])) \wedge (i17[3] \neq (i12[3], \exists(\ i12[3] \notin i14[2])))\\
&x17' := x17 \cup \{(i17[1], i17[2], i17[3])\} \qquad \text{Rule 7}
\end{aligned}$$

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

$$\begin{aligned}
&\boldsymbol{R}(\boldsymbol{Crt} - \text{sSBkt}) = \forall i18 \in x18, i19 \in x19 \mid\\
&i19[1] := i18[1] \wedge i19[2] := i18[2] \wedge (i18[3] \in i19[3]) \rightarrow (i19[3] := \mathit{ssbkt}(i18[3])) \wedge\\
&x19' := x19 \cup \{(i19[1], i19[2], i19[3])\} \qquad \text{Rule 8}
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{R}(\boldsymbol{Chk} - \text{Sign}) = \forall i20 \in x20, i21 \in x21, i22 \in x22 \mid\\
&\mathit{Check}((i20[3] := i21[3]) \rightarrow i22[1] := \mathit{TRUE} \wedge x22' := x22 \bigcup \{(i22)\} \vee\\
&\mathit{Check}((i20[3] \neq i21[3]) \rightarrow i22[1] := \mathit{FALSE} \wedge x22' := x22 \bigcup \{(i22)\} \qquad \text{Rule 9}
\end{aligned}$$

For the TRUE case in rule 9, transition Crt-uSBkt uses signatures (ie, $i24[4] := \mathit{usbkt}(i24[4])$) for given T^{u} tuples that are already available in place Upd, otherwise new signatures through transition Crt – unSBkt 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 (ie, $i24[4] := \mathit{usbkt}(i24[4])$) into Upd place for future references, as shown in rule 12.

$$\begin{aligned}
&\boldsymbol{R}(\boldsymbol{Crt} - \text{uSBkt}) = \forall i23 \in x23, i24 \in x24 \mid\\
&i23[1] = \mathit{TRUE} \rightarrow i24[4] := \mathit{usbkt}(i24[4]) \wedge i24[1] := i20[1] \wedge\\
&x24' := x24 \cup \{(i24[1], i24[4])\} \qquad \text{Rule 10}
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{R}(\boldsymbol{Crt} - \text{unSBkt}) = \forall i25 \in x25, i26 \in x26, i27 \in x27 \mid\\
&i25[1] = \mathit{FALSE} \rightarrow i27[3] := \boldsymbol{\mathit{unSBkt}}(\boldsymbol{\mathit{asg}} - \boldsymbol{\mathit{var}}(i27[3], (i26[1], i26[2], i26[3])))_{\exists(i27[3] \cap (\forall i21[3]) = 0)} \wedge\\
&x27' := x27 \cup \{(i27[3])\} \qquad \text{Rule 11}
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{R}(\text{addtoUpd}) = \forall i28 \in x28, i29 \in x29 \mid\\
&i29[1] := i28[1] \wedge i29[2] := i28[2] \wedge i29[3] := \mathit{utUpd}(i28[3]) \wedge x29' := x29 \cup \{(i29[1], i29[2], i29[3])\} \qquad \text{Rule 12}
\end{aligned}$$

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

$$\begin{aligned}
&\boldsymbol{R}(\mathrm{rtDelete}) = \forall i30 \in x30, i31 \in x31 \mid \\
&i31[1] := \mathit{delrTpl}(i30[1]) \wedge i31[2] := i30[2] \wedge i31[3] := i30[3] \\
&x31' := x31 \cup \{(i31[1], i31[2], i31[3])\} \qquad \text{Rule 13}
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{R}(\boldsymbol{Crt} - \mathrm{rSBkt}) = \forall i32 \in x32, i33 \in x33, i34 \in x34, i35 \in x35, i36 \in x36 \mid \\
&i33[1] := i32[1] \wedge i33[2] := i32[2] \wedge (i33[5] := \mathit{rSBkt}(i32[3]) \wedge x33' := x33 \cup \{(i33[1], i32[2], i33[5])\} \vee \\
&i36[1] := i32[1] \wedge i36[2] := i32[2] \wedge (i36[3] := \mathit{rnSBkt}(\mathit{asg} - \mathit{var}(i36[3], i35[1], i35[2], i35[3])))_{\exists (i32[3] \cap (\forall i34[3]) = 0)} \wedge \\
&x36' := x36 \cup \{(i36[1], i36[2], i36[3])\} \qquad \text{Rule 14}
\end{aligned}$$

All buckets created in rules 8, 10, 11, are14 are updated through transition CombSBkts by taking union of all those buckets (where $\mathrm{Bkt_i} \cap \mathrm{Bkt_j} = 0$, and $\mathrm{i} \neq \mathrm{j}$) in order to have signatures for complete dataset at one place. Transition Balancing adds the records from Cach table or from $\mathrm{T^n}$ 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.

$$\begin{aligned}
&\boldsymbol{R}(\mathrm{CombSBkts}) = \forall i37 \in x37, i38 \in x38, i39 \in x39, i40 \in x40 \mid \\
&i40[1] := \mathit{union}(i37[1], i38[1], i39[1]) \wedge i40[2] := \mathit{union}(i37[2], i38[2], i39[2]) \wedge \\
&i40[3] := \mathit{union}(i37[3], i37[4], i37[5], i38[3], i39[3]) \wedge i40[4] := \mathit{union}(i37[6], i38[4], i39[4]) \wedge \\
&x40' := x40 \cup \{(i40[1], i40[2], i40[3], i40[4])\} \qquad \text{Rule 15}
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{R}(\mathrm{Balancing}) = \forall i41 \in x41, i42 \in x42, i43 \in x43, i44 \in x44 \mid \\
&i44[1] := (i41[1] + (i42[1] \vee i43[1])) \wedge (i44[2] := i41[2] + (i42[2] \vee i43[2])) \wedge \\
&(i44[3] := \mathit{Balance}(\ i41[3]i_{\forall i41[3] \in i} + (i42[3] \vee i43[3]))) \wedge x44' := x44 \cup \{(i44[1], i44[2], i44[3])\} \qquad \text{Rule 16}
\end{aligned}$$

Algorithm 6 is formally depicted in transitions Chk – asgEq and transition Crt – asgBkts which checks the assignment equation to create buckets for the remaining $\mathrm{T^n}$ tuples (ie, $\forall \mathrm{T^n} \neq 0$). Transition Partition separates the multiple QI-groups (ie, $\mathrm{i52[3] := part(i51[3])_{\exists s \geq m})}$) 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.

$$\begin{aligned}
&\boldsymbol{R}(\boldsymbol{Chk} - \mathrm{asgEq}) = \forall i46 \in x46, i47 \in x47, i48 \in x48 \mid \\
&i48[1] := i46[1] \wedge i48[2] := i46[2] \wedge (|i46[3]| \neq 0) \rightarrow \{i48[3] := \mathit{asg} - \mathit{var}(i46[3]i_{\forall i46[3] \in i}, i47[1], i47[2], i47[3])\} \\
&x48' := x48 \cup \{(i48[1], i48[2], i48[3])\} \qquad \text{Rule 17}
\end{aligned}$$

$$\begin{aligned}
&\boldsymbol{R}(\boldsymbol{Crt} - \mathrm{asgBkts}) = \forall i49 \in x49, i50 \in x50 \mid \\
&i50[1] := i49[1] \wedge i50[2] := i49[2] \wedge i50[3] := \mathit{asgBkts}(50[3] \cup i49[3])_{\exists i49[3]_x \in 49, i50[3]_y \in 50 | x \neq y} \wedge i50[4] := i49[4] \\
&x50' := x50 \cup \{(i50[1], i50[2], i50[3], i50[4])\} \qquad \text{Rule 18}
\end{aligned}$$

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

$$\begin{aligned}
&\boldsymbol{R}(\mathrm{Bkts} - \mathrm{Sort}) = \forall i53 \in x53, i54 \in x54 \mid \\
&i54[1] := i53[1] \wedge i54[2] := \mathit{sort} - \mathit{asc}(i53[2]) \wedge i54[3] := i53[3] \wedge i54[4] := i53[4] \\
&x54' := x54 \cup \{(i54[1], i54[2], i54[3], i54[4])\} \qquad \text{Rule 20}
\end{aligned}$$

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.

$$\begin{aligned}&\boldsymbol{R}(\text{Validate\_kAnony}) = \forall i55 \in x55, i56 \in x56 \mid \\ &i56[1] := \mathit{cellGen}(i55[1], i55[2]) \wedge i56[2] := \mathit{randPermute}(i55[3], i55[4]) \\ &x56' := x56 \cup \{(i56[1], i56[2])\}\end{aligned} \qquad \text{Rule 21}$$

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 that is, $\forall(\text{Sign}_{\text{PRT}} \cup \text{Sign}_{\text{adv}}) \neq \text{Sign}_{\text{disclosure}}$ and hence the intruder gets the $\emptyset$ result. The function randPermute() in rule 21 prevents the membership attack (ie, $\text{QI}_{\text{kAnony}} \cup \text{QI}_{\text{adv}} \neq \text{QI}_{\text{disclosure}}$) 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: $\text{Sig}(\text{ST}_{\text{i}}(\text{t})) \cap \text{Sig}(\text{ST}_{\text{j}}(\text{t})) = 0, \wedge i \neq j$ and fulfillment 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.

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

$$\begin{aligned}&\boldsymbol{R}(\text{Membership} - \text{Attack}) = \forall i61 \in x61, i62 \in x62, i63 \in x63 \mid \\ &\mathit{MShipDis}(i61[1]) \rightarrow (i61[1] \cup i62[1]) \neq i63[1] \vee \mathit{MshipDisc}(i61[1] \cup i62[1]) = \emptyset\end{aligned} \qquad \text{Rule 23}$$

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 3500 tuples have randomly been deleted and 4000 tuples have been inserted from the remaining tuples. For internal updates, 1000 tuples have been updated from previous release with respect to A^{S} and 1000 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 subsection 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 References 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.

$$\mathrm{NCP}_{\mathrm{QI_i}}(\mathrm{t}) = \frac{\mathrm{z_i} - \mathrm{y_i}}{|\mathrm{QI_i}|} \tag{4}$$

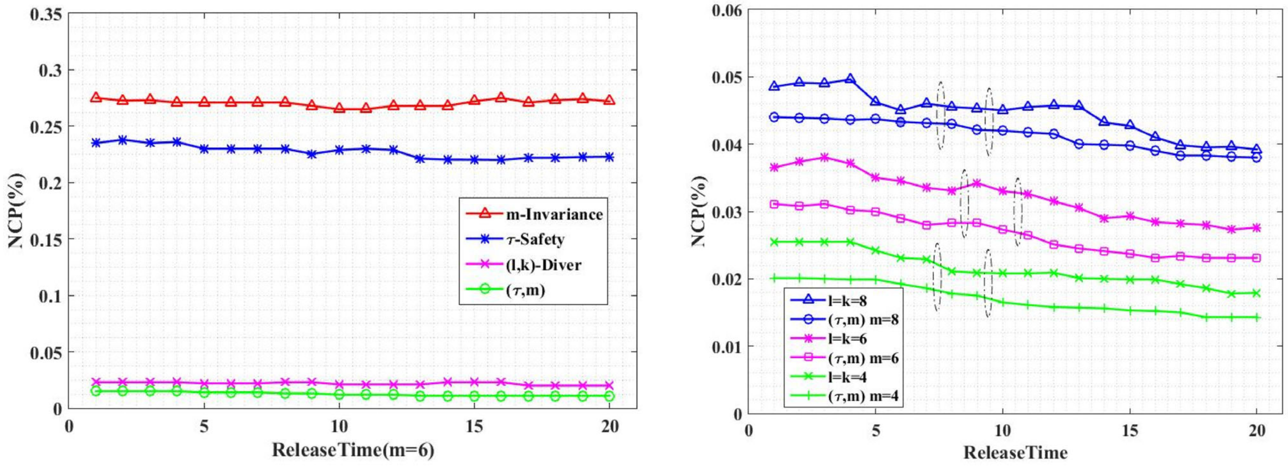
$$\mathrm{NCP}(\mathrm{t}) = \sum_{\mathrm{i}=1}^{\mathrm{q}} \mathrm{w_i}.\mathrm{NCP}_{\mathrm{Q_i}}(\mathrm{t}) \tag{5}$$

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

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

Figure 5 depicts utility measurement for the anonymized releases using NCP. The algorithm was tested under different parameters that is, release time and *m* value. Figure 5A 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 5A the y-axis is the percentage value from Equation (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.

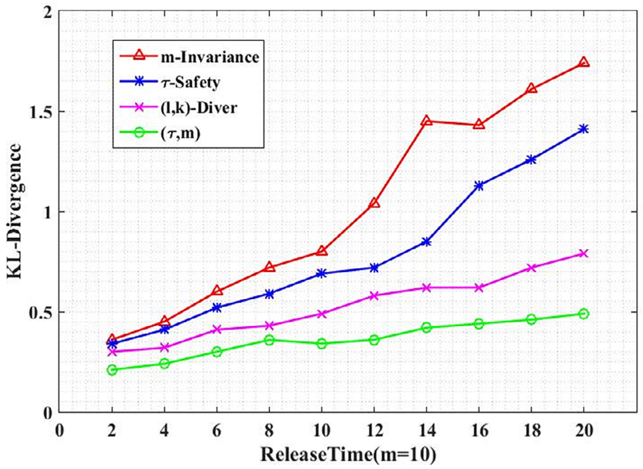
Figure 5B compares τ-safe (*l, k*)-diversity and (τ, *m*)-slicedBucket algorithms to calculate NCP% for varying values of *m*, that is, $m = 4$, $m = 6$, and $m = 8$ during the 20 anonymized releases. The comparative graphs produced by both approaches have been encircled. As time evolves, closer records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



(A) (B)

FIGURE 5 Utility measurements, (A) Inform. loss for fixed *m*, (B) Inform. Loss of (*l,k*) and (τ, m) for varying *m*

FIGURE 6 Utility measurements through KL-divergence



k)-diversity for each parameter during all the releases because of the negligible cell generalization in each sliced bucket and reduced dimensionality during slicing.

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.

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

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 that is, 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

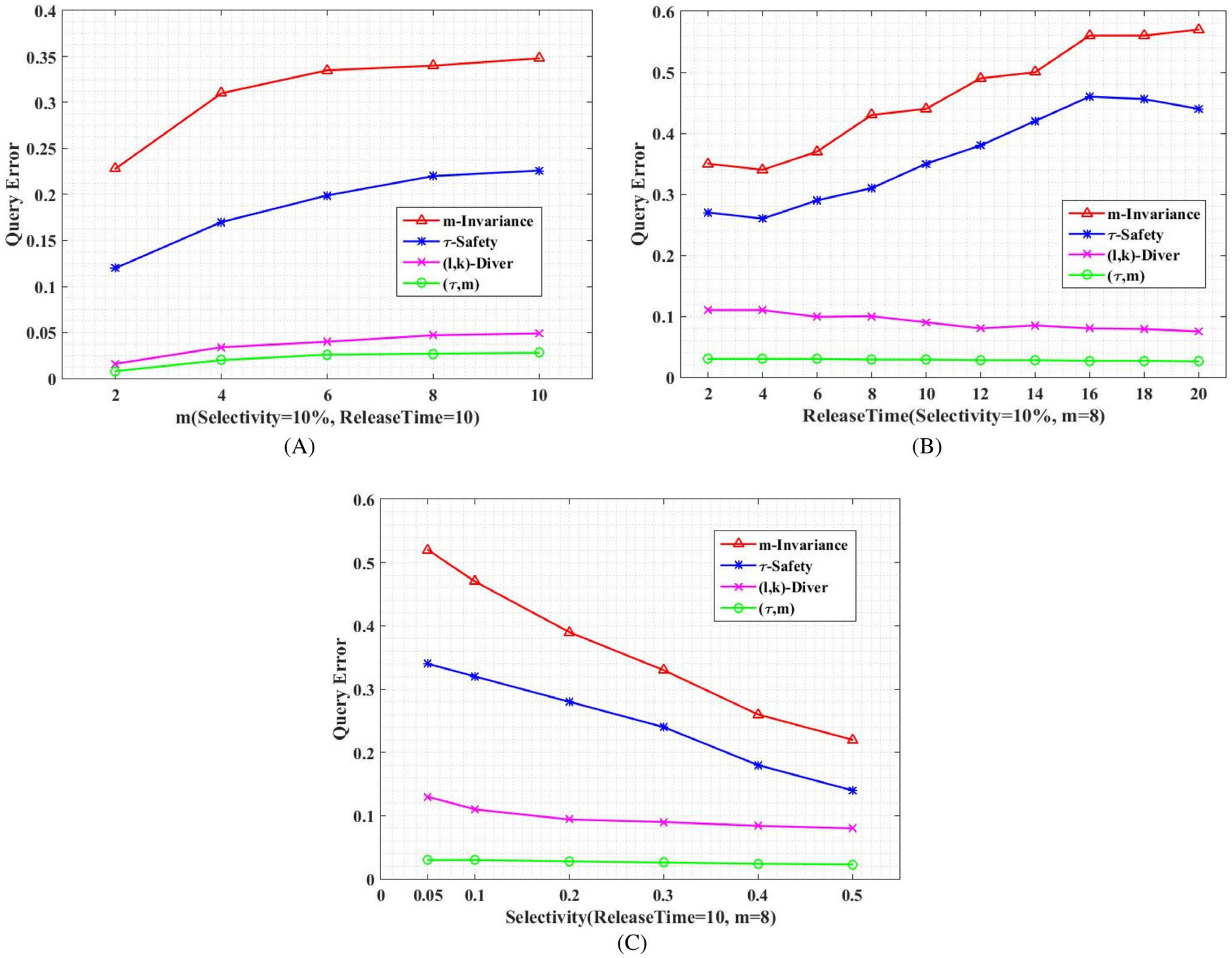
Query accuracy[11,12] evaluates utility of the anonymized release by triggering aggregate queries. The anonymized release T^* from original microdata T having q as maximum A^{QI}'s, that is, . $A_1^{QI}, A_2^{QI} A_3^{QI}, \dots, A_q^{QI}$ where $D(A_i^{QI})$ is the domain of the ith A^{QI}. Then the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

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

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 $|T_{\text{Query}}|$ on T where |T| are the total number of records in the dataset, the query selectivity is $\theta = \frac{|T_{\text{Query}}|}{|T|}$. The query error in Equation (8) is then a difference between the anonymized and original dataset.

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

where count(anonymized) is the output from T^* and count(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, that is, varying m, varying time and varying selectivity. We executed 1000 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 for example, 10th release, can be seen in Figure 7A at 10% selectivity. High value of m means high generalization of A^{QI}'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 7B, 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^{QI}'s over different releases. In the proposed (τ, m)-slicedBucket model because of the reduced dimensionality in A^{QI}s the query error is further reduced with the same minimum distance A^{QI} placement in an EC. Figure 7C 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.



(A)

(B)

(C)

FIGURE 7 Query error (A) Variable m size (B) Updating release time (C) Change in selectivity

FIGURE 8 Counterfeit used

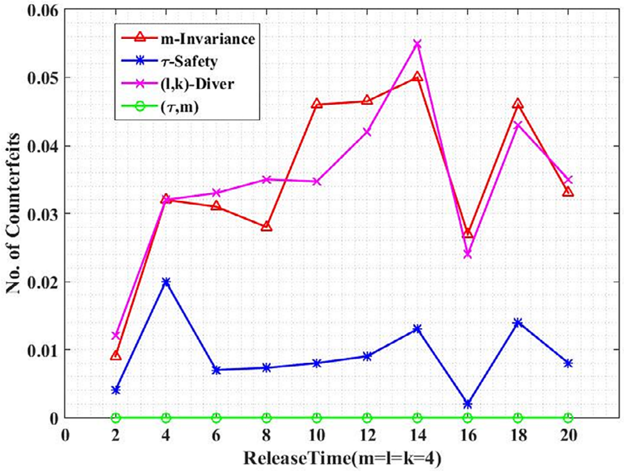
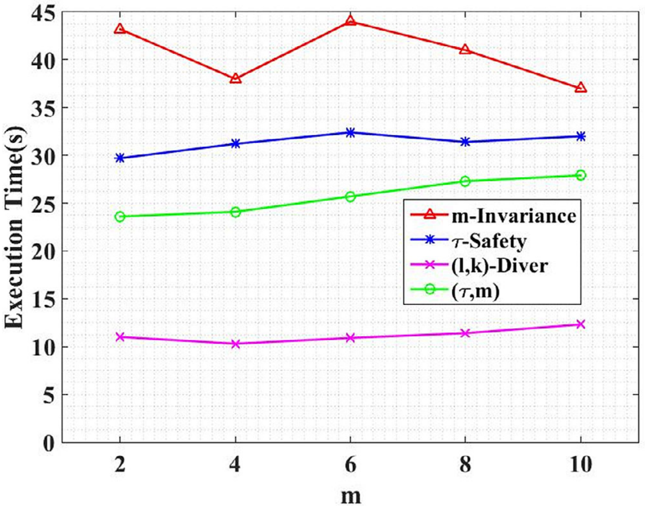


FIGURE 9 Algorithm execution time



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 Sections 1.1(iii) and 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 20 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 $(\tau$,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.

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 that is, 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, that is, 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 SAs (MSA),[49] (ii) sequential dynamic data publication with individual having more than one records that is, 1:M microdata,[29] and the more challenging work is to propose a solution for (iii) 1:M-MSA with sequential dynamic data publishing.

ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China under Grant 61932005.

CONFLICT OF INTERESTS

The authors declare no potential conflict of interest.

ORCID

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

REFERENCES

1. Asghari P, Rahmani AM, Javadi HHS. A medical monitoring scheme and health-medical service composition model in cloud-based IoT platform. *Trans Emerg Telecommun Technol*. 2019;6:30. https://doi.org/10.1002/ett.3637.
2. Chen C-L, Yang T-T, Deng Y-Y, Chen C-H. A secure Internet of Things medical information sharing and emergency notification system based on nonrepudiation mechanism. *Trans Emerg Telecommun Technol*. 2020;e3946;1–21. https://doi.org/10.1002/ett.3946.
3. Azad MA, Arshad J, Mahmoud S, Salah K, Imran M. A privacy-preserving framework for smart context-aware healthcare applications. *Trans Emerg Telecommun Technol*. 2019;e3634;1–20. https://doi.org/10.1002/ett.3634.
4. Largest healthcare data breaches of 2019. https://www.hipaajournal.com/2019-healthcare-data-breach-report/. Accessed April 27, 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 April 27, 2020.
6. Sweeney L. k-Anonymity: A model for protecting privacy. *Int J Uncertainty Fuzziness Knowl Based Syst*. 2002;10(5):1-14.
7. Sweeney L. Achieving k-anonymity privacy protection using generalization and suppression. *Int J Uncertainty Fuzziness Knowl Based Syst*. 2002;10(5):1-18.
8. Machanavajjhala A, Kifer D, Gehrke J. l-Diversity: Privacy Beyond k-Anonymity. *ACM Trans Knowl Discov Data*. 2007;1(1):1-52.
9. Li N, Li T, Venkatasubramanian S. t-Closeness: privacy beyond k-anonymity and l-diversity. Paper presented at: IEEE 23rd Int. Conf. Data Eng.; 2007; Istanbul:106-115.
10. Xiao X Tao Y. m-invariance: towards privacy preserving re-publication of dynamic datasets. Paper presented at: Proceedings of the ACM SIGMOD International Conference on Management of data; 2007:689–700.
11. Anjum A et al. τ-safety: A privacy model for sequential publication with arbitrary updates. *Comput Secur*. 2017;66:20-39.

12. Zhu H et al. τ-Safe (l,k)-diversity privacy model for sequential publication with high utility. *IEEE Access*. 2019;7:687-701.
13. Li T, Li N, Zhang J, Molloy L. Slicing: A new approach for privacy preserving data publishing. *IEEE Trans Knowl Data Eng*. 2012;24(3):561-574.
14. Touhidul Hasan ASM et al. A new approach to privacy-preserving multiple independent data publishing. *Appl Sci*. 2018;783:8.
15. Shmueli E, Tassa T, Wasserstein R. Limiting disclosure of sensitive data in sequential releases of databases. *Inform Sci*. 2012;191:98-127.
16. Riboni D, Pareschi L, Bettini C. JS-reduce: defending your data from sequential background knowledge attacks. *IEEE Trans Dep Sec Comp*. 2012;9(3):387-400.
17. Liang H, Wu J, Mumtaz S, Li J, Lin X, Wen M. MBID: micro-blockchain-based geographical dynamic intrusion detection for V2X. *IEEE Commun Mag*. 2019;57(10):77-83.
18. Chen J et al. Collaborative trust blockchain based unbiased control transfer mechanism for industrial automation. *IEEE Trans Indus Appl*. 2019;56(4):4478–4488. https://doi.org/10.1109/TIA.2019.2959550.
19. Domingo-Ferrer J, Soria-Comas J. From t-closeness to differential privacy and vice versa in data anonymization. *Knowl Based Syst*. 2015;74:151-158.
20. Cao J, Karras P. Publishing microdata with a robust privacy guarantee. Paper presented at: Proceedings of the 38th International Conference on Very Large Data Bases(VLDB), VLDB Endowment; 2012, vol. 5, no. 11:1388-1399.
21. Khan R, Tao X, Anjum A, Kanwal T, Khan A, Maple C. θ-Sensitive k-anonymity: an anonymization model for iot based electronic health records. *Electronics*. 2020;9(5):1–24. https://doi.org/10.3390/electronics9050716.
22. Cormode G, Srivastava D, Shen E. Aggregate query answering on possibilistic data with cardinality constraints. Paper presented at: IEEE 28th International Conference on Data Engineering; 2012; Washington, DC:258-269.
23. Shi Y, Zhang Z, Chao HC. Data privacy protection based on micro aggregation with dynamic sensitive attribute updating. *Sensors (Basel)*. 2018;7:18.
24. Zhang Q, Koudas N, Srivastava D Yu T. Aggregate query answering on anonymized tables. Paper presented at: IEEE 23rd International Conference on Data Engineering; 2007; Istanbul:116-125.
25. Domingo-Ferrer J Soria-Comas J. Steered microaggregation: a unified primitive for anonymization of data sets and data streams. Paper presented at: IEEE International Conference on Data Mining Workshops; 2017; New Orleans, LA:995-1002.
26. Soria-Comas J, Domingo-Ferrer J. Enhancing data utility in differential privacy via micro-aggregation based k-anonymity. *VLDB J*. 2014;23(5):771-794.
27. Xiao X, Tao Y. Anatomy: simple and effective privacy preservation. Paper presented at: Proceedings of the 32nd International Conference on Very Large Data Bases; 2006:139-150.
28. Tao Y, Chen H, Xiao X, Zhou S, Zhang D. Angel: enhancing the utility of generalization for privacy preserving publication. *IEEE Trans Knowl Data Eng*. 2009;21(7):1073-1087.
29. Gonga Q, Luoa J, Yang M. Anonymizing 1:m microdata with high utility. *Knowl Based Syst*. 2016;115:15-26.
30. Shannon CE. Prediction and entropy of printed english. *Bell Syst Tech J*. 1951;30(1):50-64.
31. Saba Yaseen SM, Abbas A, Anjum A. Improved generalization for secure data publishing. *IEEE Access*. 2018;6:27156-27165.
32. Amiri F et al. Hierarchical anonymization algorithms against background knowledge attack in data releasing. *Knowl Based Syst*. 2016;101:71-89.
33. Song F, Ma T, Tian Y. A new method of privacy protection: random k-anonymous. *IEEE Access*. 2019;7:75434-75445.
34. Fung BCM, Wang K, Yu PS. Top-down specialization for information and privacy preservation. Paper presented at: Proceedings of the 21st IEEE International Conference on Data Engineering (ICDE); 2005:205-216.
35. Wang K, Yu PS, Chakraborty S. Bottom-up generalization: a data mining solution to privacy protection. Paper presented at: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM); 2004:249–256.
36. Barak B, Chaudhuri K, Dwork KC, Kale S, Mcsherry F Talwar K. Privacy, accuracy, and consistency too: A holistic solution to contingency table release. Paper presented at: Proc. the 26th ACM Symposium on Principles of Database Systems (PODS); 2007:273-282.
37. Shmueli E, Tassa T. Privacy by diversity in sequential releases of databases. *Inform Sci*. 2015;298:344-372.
38. Wang K, Fung B. Anonymizing sequential release. Paper presented at: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2006:414–423.
39. Bu Y, Fu AWC, Wong RCW, Chen L Li J. Privacy preserving serial data publishing by role composition. Paper presented at: Proc the VLDB Endowment; 2008, vol. 1, no. 1:845-856.
40. Bouna BA, Clifton C, Malluhi Q. Efficient sanitization of unsafe data correlations. Paper presented at: Proc. the Workshops of the EDBT/ICDT Joint Conf; 2015:278-285.
41. Li T, Li N, Zhang J. Modeling and integrating background knowledge in data anonymization. Paper presented at:Proc the 25st IEEE Int Conf on Data Engineering (ICDE); 2009:6-17.
42. Wang H, Liu R. Privacy-preserving publishing microdata with full functional dependencies. *Data Knowl Eng*. 2011;70:249-268.
43. Byun JW, Sohn Y, Bertino E. Secure anonymization for incremental datasets. *Secure Data Management*. Lecture Notes in Computer Science. Vol 4165. Berlin, Heidelberg: Springer; 2006. https://link.springer.com/chapter/10.1007/118446624#citeas.
44. Anjum A et al. An efficient privacy mechanism for electronic health records. *Comput Secur*. 2018;72(C):196–211.
45. Malik SU, Khan SU. Modeling and analysis of state-of-the-art VM-based cloud management platforms. *IEEE Trans Cloud Comput*. 2013;1(1):1-1.
46. Kanwal T et al. Privacy-preserving model and generalization correlation attacks for 1:M data with multiple sensitive attributes. *Inform Sci*. 2019;488:238-256.

47. Dwork C, Roth A. The algorithmic foundations of differential privacy. *Found Trends Theoret Comput Sci*. 2014;9(3–4):211-407.
48. Li N Qardaji W, Su D. Provably private data anonymization: Or, k-anonymity meets differential privacy. https://www.cerias.purdue.edu/assets/pdf/bibtex_archive/2010-24.pdf. Accessed February 11, 2020.
49. Razaullah Khan, Xiaofeng Tao, Adeel Anjum, "Privacy preserving for multiple sensitive attributes against fingerprint correlation attack satisfying *c*-diversity," *Wirel Commun Mobile Comput*. 2020;2020:1–18. https://www.hindawi.com/journals/wcmc/2020/8416823/. Accessed February 12, 2020.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

How to cite this article: Khan R, Tao X, Anjum A, et al. (τ, m)-slicedBucket privacy model for sequential anonymization for improving privacy and utility. *Trans Emerging Tel Tech*. 2022;33:e4130. https://doi.org/10.1002/ett.4130