K-VARP

Otgonbayar, Ankhbayar; Pervez, Zeeshan; Dahal, Keshav; Eager, Steve

Published in:
Information Sciences

DOI:
10.1016/j.ins.2018.07.057

Published: 31/10/2018

Document Version
Peer reviewed version

Citation for published version (APA):
Otgonbayar, A., Pervez, Z., Dahal, K., & Eager, S. (2018). K-VARP: K-anonymity for varied data streams via partitioning. Information Sciences, 467, 238-255. https://doi.org/10.1016/j.ins.2018.07.057

K-VARP: K-anonymity for varied data streams via partitioning

Ankhbayar Otgonbayar, Zeeshan Pervez, Keshav Dahal, Steve Eager

University of the West of Scotland Paisley, Scotland, UK

Abstract

The Internet-of-Things (IoT) produces and transmits enormous amounts of data. Extracting valuable information from this enormous volume of data has become an important consideration for businesses and research. However, extracting information from this data without providing privacy protection puts individuals at risk. Data has to be sanitized before use, and anonymization provides solution to this problem. Since, IoT is a collection of numerous different devices, data streams from these devices tend to vary over time thus creating varied data streams. However, implementing traditional data stream anonymization approaches only provide privacy protection for data streams that have predefined and fixed attributes. Therefore, conventional methods cannot directly work on varied data streams. In this work, we propose K-VARP (K-anonymity for VARied data stream via Partitioning) to publish varied data streams. K-VARP reads the tuple and assigns them to partitions based on description, and all tuples must be anonymized before expiring. It tries to anonymize expiring tuple within a partition if its partition is eligible to produce a K-anonymous cluster. Otherwise, partition merging is applied. In K-VARP we propose a new merging criterion called R-likeness to measure similarity distance between tuple and partitions. Moreover, flexible re-using and imputation free-publication is implied in K-VARP to achieve better anonymization quality and performance. Our experiments on a real datasets show that K-VARP is efficient and effective compared to existing algorithms. K-VARP demonstrated approximately three to nine and ten to twenty percent less information loss on two real datasets, while forming a similar number of clusters within a comparable computation time.

Keywords: Internet of Things, Data privacy, Data streams, Anonymization, Missing values

1. Introduction

The technological revolution of the Internet-of-Things (IoT hereafter) has become an inseparable part of the modern world. We are living in an era in which enormous volumes of data are generated and transmitted in the form of streams[1]. Everything that we do in our lives leaves a trace, forming a digital data stream, such as, the browser history of internet users, bank transactions and energy consumption logs of houses. Extracting this valuable knowledge from the streaming data can provide a realistic and approximate insight into individuals' activities [2] and the behaviour of a society [3]. Many organizations publish and exchange data for business and research purposes; however, processing individuals' information without compromising privacy is a primary concern for IoT [4, 5].

The most popular technique to provide privacy protection for publishing data is anonymization [6, 7, 8, 9, 10]. Anonymization removes or replaces the information,

Email addresses: ankhbayar.otgonbayar@uws.ac.uk (Ankhbayar Otgonbayar), zeeshan.pervez@uws.ac.uk (Zeeshan Pervez), keshav.dahal@uws.ac.uk (Keshav Dahal), steve.eager@uws.ac.uk (Steve Eager)

which can be exploited by an attacker, to compromise the privacy of a user. Therefore, individuals remain hidden from potential threats when their data is published for analytical or business purposes. Confidential or identifier information of individuals, which must not be published to the public domain, is called sensitive information. Non-sensitive information which can be exploited by an attack is called quasi-identifiers (QID hereafter). Anonymization approaches are classified into two major classes; static data anonymization and data stream anonymization [11, 12, 13]. Static data anonymization works with pre-recorded datasets having pre-defined QIDs. The quality of the static data anonymization is measured by information loss which indicates the usability of anonymized data. Data stream anonymization processes the data on the fly (*i.e.* publishes the data as it arrives) [13, 14, 15, 16]. The quality of the data stream anonymization is defined by a tradeoff between data freshness and data usability. Some publishers may want fast anonymization - although it gives more disrupted data; whereas, some publishers may prioritize data usability rather than data freshness to get data which is more precise. For example, the data stream of a mission critical system requires a minimum delay to publish data that can be used to take immediate action against potential threats. On the other hand, sales transaction data can be processed with a longer delay when data usability is prioritized. Sliding window is the most widely used technique for data stream anonymization, it keeps an anonymization algorithm consistent and tolerant when dealing with fast and high dimensional data streams [17, 18, 19]. This technique is an accumulation based mechanism for anonymizing data streams, which prevents the overflow of memory and helps to publish data continuously.

IoT consists of multiple internet enabled sensing and actuating devices used by individuals for different purposes. For instance, smart car, smart heating, fire alarms and security cameras for smart homes and offices, wearable devices to measure the physical performance of a person, and data generated from smart cities to provision personalized services to the inhabitants. IoT data streams generate data streams with missing values due to its unstable and uncontrollable properties. There are three main factors that cause missingness on IoT data streams:

- **Individuals' preference:** each individual have varying types of devices depending on their preferences;
- **Different usage pattern:** each individual can choose to use different devices at any given time;
- **Uncertain environmental condition:** environmental conditions can cause devices to malfunction or lose connectivity.

Therefore, we call it varied data stream due to the varying sets of QIDs in each tuple of missing data stream. Anonymizing data with missing values is always an interesting topic for researchers [20]. The main challenge for anonymizing incomplete data is handling the missingness in data streams originating from multiple streams *i.e.* IoT devices used by a user. Researchers identified three main methods to handle missingness of static data:

a) **Imputation:** values are calculated to fill the missingness [21];
b) **Marginalization:** ignore missingness while anonymizing [22];
c) **Partitioning:** splits data into disjoint partitions based on tuple's description [21].

However, there has been no substantial work published on handing missingness for data stream anonymization. Incomplete dataset anonymization techniques can be extended to work on varied data streams; however, this will cause more information loss, weak privacy protection and a high computational time.

To the best of our knowledge, there is no known algorithm proposed specifically to anonymize varied data stream. To address this, we are proposing K-VARP (K-anonymity for VARied data stream via Partitioning) for anonymizing varied data streams. Our target is to anonymize and publish varied data streams with minimum delay and less information loss. The K-VARP algorithm uses both partitioning and marginalization methods to anonymize varied data streams under a time based sliding window. As previously discussed, a time based sliding window is the most convenient technique for anonymizing data streams, allowing us to publish data streams with minimum delay and less information loss.

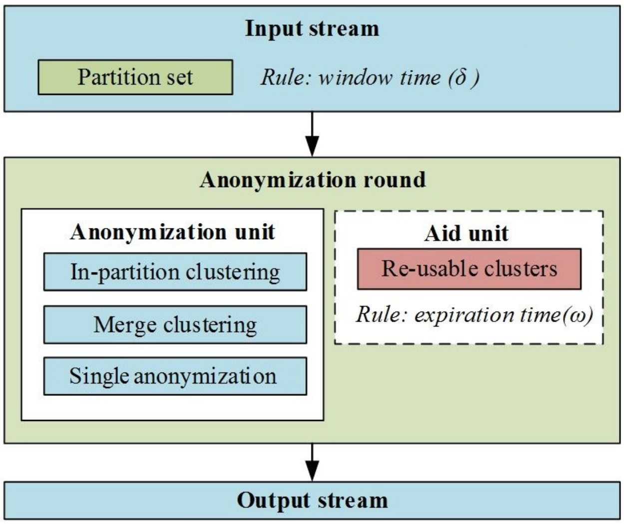


Figure 1: Two phases of K-VARP algorithm with its internal working details.

Our proposed algorithm K-VARP provides privacy preserving capabilities to real world applications that utilizes varied data streams. For example, social network analysis[23, 24], patient monitoring [25, 26] and smart city [27, 28].

An overview of the K-VARP algorithm is illustrated in Fig. 1. K-VARP has two main phases, partitioning and anonymizing. In partitioning, K-VARP assigns receiving tuples to partitions using their QID set with their received timestamp attached. This phase plays the role of a buffer, and helps to store received tuples in an organized form to perform fast and efficient anonymization. In time-based sliding window, the maximum time for each tuple to stay in the buffer is defined by a time constraint, denoted as δ (see Fig (1)). Each expiring tuple has its own anonymization round. The anonymization round is invoked when a tuple is about to expire according to time window criteria δ.

There are three modules to anonymize an expiring tuple t' regarding the size of its partition P', and each of these modules has an option to anonymize an expiring tuple by re-using recently published K-anonymous clusters.

- **In-Partition clustering:** This module is designed to publish cluster with no missing value. However, it is invoked only if partition has enough tuples to form K-anonymous cluster for expiring tuple.
- **Merge clustering:** Partition merging is inevitable when dealing with varied data stream, and this module is designed to merge the most suitable partitions to anonymize expiring tuples with less information loss.
- **Single anonymization:** This module is designed to publish expiring tuple when partition merging is not possible for expiring tuple.

For more details, please refer to Section 4.

Experiments on real datasets demonstrate the efficiency and effectiveness of K-VARP. Efficient merge clustering that uses R-likeness significantly helped to anonymize varied data streams with less information loss. Also, flexible re-using criteria decreases computation time and improves privacy protection. The contributions of the proposed K-VARP algorithm are:

- **Imputation free anonymization for varied data streams:** This is the first substantial effort to anonymize a varied data stream without using imputation to handle missing values in multiple data streams.

- **Transitive merging criteria that consider data distribution of partition:** We have applied customizable merge criteria called R-likeness to calculate distance between tuples and partitions. This measurement is used to identify and merge the closest partitions. Unlike merge operations in conventional methods which merely rely on partition size, K-VARP calculates the attributes distribution in order to merge similar partitions, thus causing less information loss.

- **Flexible re-using strategy to provide better anonymization for less time:** This flexible re-using strategy gives tuples a better opportunity to avoid time consuming clustering operations. Also, tuple anonymized by re-using, ensures privacy. More number of re-using, increases the average number of tuples of each cluster that directly improves the privacy protection of K-anonymity.

The rest of this paper is organized as follows: Section 2 presents the related work. Section 3 introduces the basic concept of varied data stream anonymity and defines the anonymization model for varied data streams. In Section 4, K-VARP is explained in detail. Section 5 compares the experimental result of K-VARP with other widely used anonymization algorithms. The paper is concluded along with future directions in Section 6.

2. Related work

In this section, related work is categorised as anonymization algorithms for data streams and techniques to handle missing data while publishing data for analytical purposes.

2.1. Data stream anonymization

Data stream anonymization must be performed as quickly and efficiently as possible. However, the quality of data stream anonymization dependent on the trade-off between anonymization time and information loss. To process data streams in a dynamic environment, researchers implemented a sliding window technique. Sliding window is a popular anonymization technique for data stream anonymization, which anonymizes the most recent tuples of data and publishes freshly received data. There are two main types of sliding window; time based and count based. In count based sliding window, the anonymization round is invoked when the sliding window size reaches a certain threshold. On the other hand, in time based sliding window, anonymization is controlled by the received time of a tuple in the sliding window.

A well-known anonymization algorithm called CASTLE is proposed by Cao *et al.,* in [14]. CASTLE is a count-based sliding window algorithm, which assigns receiving tuples to immature clusters. When tuples are expiring CASTLE releases them immediately. However, if an expiring tuple is not assigned to a K-anonymous cluster, it performs a merge and split operations to create a K-anonymous cluster. Furthermore, to minimize information loss, CASTLE adopts a cluster re-using strategy to anonymize newly arriving tuples using generalization information of recently published clusters.

Hessam and Sylvia introduced FAANST, a count based sliding window anonymization algorithm for numerical data streams [11]. The main purpose of FAANST is to enhance data quality. To achieve this, the authors proposed information loss constraint for each cluster. It outputs K-anonymized clusters having less than Δ information loss. Since, it uses a count based sliding window, the tuples are only published when the window is filled with a certain number of tuples. FAANST outperforms CASTLE in terms of data quality and time complexity. However, due to the versatility of varied data streams, count based sliding window is not reliable system for handling varied data streams for anonymization. Most count based sliding window algorithms are improved by applying a time-based sliding window. Wang *et al.,* found that CASTLE [14] generated fewer huge clusters when applied on a data stream, resulting in frequent split operations, creating many K-anonymized small clusters [14]. The split and merge operations in CASTLE were time consuming and resulted in higher information loss during re-clustering and publishing. To deal with this B-CASTLE was proposed. It sets a threshold on cluster size, by changing the optimum selection and merging features of CASTLE. B-CASTLE demonstrated a higher efficiency and lower complexity compared to CASTLE.

Guo and Zhang proposed a data stream anonymization algorithm with time constraint called FADS [17]. It resolved the problem of cluster overload found in CASTLE when homogeneous data streams have non-negligible time differences between arriving tuples. FADS considered time delay as the main constraint and set a time constraint on the sliding window, and cluster set. By this, the longest time for a tuple to stay in memory is δ, and re-usable K-anonymized clusters are held for a certain amount of time. The authors noted that the complicated merge and split operations of CASTLE are unnecessary since the cluster size is already constrained by K.

Zhou *et al.,* developed a three-phase method for generalizing streaming data [16]. In the first stage, their algorithm makes a decision about data publishing based on cluster information loss. In the second step, the distribution of the data stream is incorporated in the decision making process of cluster anonymization. In the third step, the effect of cluster anonymization on future tuples is considered. The authors considered that the data publishing based on uncertainty may not be effective because it does not consider the distribution of tuples in a streams. They developed a feature that takes account into the distribution of tuples, which allows a tuple from a sparse area to output before a tuple from a dense area. They adopted the chain sampling method [29] to estimate the density of tuples' area and to reduce computational time.

Moreover, Esmaeil *et al.,* [30], considered that FADS [17] handles tuples in sequence, therefore, it is not a suitable solution for anonymizing data streams. They introduced a parallel algorithm that provides an efficient big data anonymization with multithreaded technique named FAST. The algorithm reads tuples continuously and passes them to new threads until the number of threads reaches the threshold (maximum allowed threads). To publish data, all threads launch a publish function to output the oldest (expiring) tuples from the receiving tuple set. The algorithm finds K-Nearest Neighbours(KNN hereafter) [31] (*i.e.*, tuples) to form a cluster for the oldest tuple. If there is a re-usable cluster which offers less information loss compared to a newly created cluster, then the tuple is published using the re-useable cluster and the re-usable cluster set is updated. In the event that no re-use cluster offers better information loss, the tuple is published with the newly created cluster. It then estimates the other closer K-1 tuples' remaining time. If they have enough time for the next round it remains in memory, if not it suppresses and then outputs the K-1 tuples. The algorithm provides more efficient anonymization when the number of threads are increased, performing more efficiently than FADS in terms of information loss.

Researchers agreed that a time based sliding window is more efficient for anonymizing data streams [17, 19, 30, 32]. Each tuple in a time based sliding window must be processed before expiration - this helps to output the tuple in a similar order to how it was received. In contrast, a count based sliding window has a strict anonymization mechanism. There is no expiration time of a tuple, a single anonymization round starts when a sliding window contains a certain number of tuples. To perform another anonymization round, a count based window has to accumulate the required number of tuples. Therefore, in count based sliding windows some tuples stay longer compared to others before getting anonymized [12, 19, 32, 33].

2.2. Missing data handling in anonymization

Missing data is always a serious problem for data analytics. Before handling missing data, we must understand why data is missing. Graham *et al.,*[21] classified causes of missingness as followings:

- **Not Missing At Random (NMAR):** The cause of missingness is explicit. A direct correlation between missingness and cause of missing is definite when data is missing under NMAR. For example, occasionally we find empty seats on a plane before it's departures. The reason for the empty seats are clear; the seats have not been booked or the commuters have missed their flight.

- **Missing At Random (MAR):** MAR implies that there is a somewhat coherent cause behind the randomness of missingess. Randomness of missing data has happened for a reason but the missingness is random. For example, when spell checking large documents for errors, a reader, in order to have the documents reviewed on time, may inadvertently leave some grammatical errors unchecked

- **Missing Completely At Random (MCAR):** This is the most extreme cause of missingness. There are no reasons to explain what causes missingness in an MCAR situation. Researchers consider MCAR is purely haphazard, such as rolling dice or flipping a coin. These scenarios likened to MCAR.

The most common type of the missingness is MAR. Since IoT is a combination of devices, we consider the missingness of varied data streams as MAR. There is no substantial work published that anonymizes data streams with missing values. However, researchers identified imputation, marginalization and partitioning as three possible approaches to manage the missingness of datasets while anonymizing.

Imputation: By using this method, missing values of varied data streams are replaced by pre-calculated representative values. Sarkar *et al.,* applied a Fuzzy K-means algorithm for dataset containing missing values [20]. To reduce the uncertainty caused by imputation, they attempted to repair the missing values while clustering, instead of preprocessing data before anonymization. The Fuzzy K-means algorithm creates a greater number of independent clusters with different imputation on missing data compared to the K-means algorithm with preprocessing. Imputation has a disadvantage, that when the percentage of missing values in a data stream is relatively high, the uncertainty of anonymized data due to a higher number of imputed values is amplified.

Marginalization: In marginalization, a missing value is handled as a $NULL$ and anonymized as part of the range attribute and the node of categorical attribute in the generalization hierarchy [28]. The major drawback of this type of anonymization is that tuples with different descriptions can be assigned to a same cluster, which can be too sparse to analyze. However, an advantage of this method is that the original data is not disrupted by imputation. Wagstaff *et al.,* noted that, marginalization is a better solution because this method does not add any new data values [22]. If missingness of a published cluster is less than the size of the cluster, then marginalization can provide a fast and more secure anonymization.

Partitioning: Datasets with missing values can be divided into several complete datasets, which can then be published through traditional anonymization approaches.

This strategy is not cost-efficient when a dataset has a relatively high percentage of missing values compared to its size. However, by using this method, we can publish solid clusters with no missing data that do not require any imputation, with privacy protection also secured. Ciglic *et al.,* investigated on anonymization of datasets containing $NULL$ values. They considered three $NULL$ value matching schemes for datasets, and proposed an anonymization system called ANON [34]. In this approach, dataset was divided into separate partitions by the tuples' attribute description and then a *best-first-search* was applied to find the optimal anonymization solution for each partition.

Suppression based methods can be applied to ensure the privacy of a dataset through anonymization. Jia *et al.,* presented a partial suppression algorithm to anonymize datasets in [35]. The authors identified that some $QIDs$ can have sensitive values. This issue is resolved by *global suppression* under ρ-uncertainty [36] privacy model. Moreover, they stated that the *global suppression* algorithm is not efficient and suppressing all sensitive values is unnecessary. Therefore, a partial suppression algorithm is proposed to minimize the suppression while providing privacy preservation. This algorithm is proposed to anonymize datasets; however, suppression based methods can be applied on varied data streams as assistive modules to ensure privacy.

3. Anonymizing varied data streams

In this section, we formally define the data stream anonymization for varied data streams.

Definition 1 (Quasi-identifiers). *Let* Q=$\{q_1,q_2,...,q_n\}$ *be a set of attributes of data streams which need to be anonymized before publishing. We call Q a set of QIDs.*

Definition 2 (Tuple of data stream). *Tuple of data stream is defined as:* $t(id_t, Q_t, ts_t)$*- where* id_t *is the identity of an individual,* $Q_t = \{q_1, q_2, ..., q_m\}$ *is a set of QIDs of a tuple, and* ts_t *is a timestamp at which the tuple is received.*

In a conventional streaming scenario, received data has fixed attributes with no missing values; whereas, in varied data streams one or more random attributes can be missing. In the varied data streaming settings, each tuple can contain a different description of the data. In the following, we defined varied data streams based on definition 1 and definition 2.

Definition 3 (Varied data stream). *Let* Q *be a set of QIDs that can appear in a data stream, where* $Q = \{q_1, q_2, , q_n\}$*. We define a varied data streams as* $VS(id, Q_t, ts)$ *where id is the identity,* Q_t *is the subset of* $Q(Q_t \subseteq Q)$ *that describes a receiving tuple, and* ts *is the arrival timestamp of a tuple.*

A varied data streams consist of tuples with different QID sets. In the following we define cluster, K-anonymous cluster of varied data streams and K-anonymized varied data stream respectively in definition 4, definition 5 and definition 6.

Definition 4 (Cluster of varied data streams). *Cluster is a set of tuples in a varied data stream* VS*. Let* S_c *be a set of tuples,* Q_c *be a set of QIDs that can be found in tuples of* S_c*. Then a cluster of varied data streams C is defined as:* $C(Q_c) = \{t(id_t, Q_t, ts_t) \mid t \in S_c \wedge Q_t \subseteq Q_c\}$

Definition 5 (K-Anonymous cluster). *Let* $C(Q_c)$ *be a cluster* C *built from a varied data stream* VS*. If the number of distinct identities of tuples in* $C(Q_c)$ *is greater than* K *we call* $C(Q_c)$ *a* K*-anonymous cluster.*

Definition 6 (K-anonymized varied data stream). *Let* $VS(id, Q_t, ts)$ *be a varied data stream, and* VS_{out} *be an anonymized stream generated from* VS*.* VS_{out} *is called* K*-anonymized when following the conditions are met:*

a) *For* $\forall t \in VS, \exists t' \in VS_{out}$ *corresponds to* t*.*

b) *For* $\forall t' \in VS_{out}, DI(C(Q'_t)) \geq k$, *when* $C(Q'_t)$ *is the cluster containing* t' *which belongs to* VS_{out}*. DI counts the number of distinct values of the tuples' id in* $C(Q'_t)$*.*

Clusters generated from varied data streams can contain tuples from multiple partitions. Therefore, cluster generalization of varied data streams is different than traditional cluster generalization. Traditional generalization functions create a virtual tuple to represent all tuples of a cluster. In contrast, anonymization on varied data streams can generate cluster with different types of tuples *i.e.*, tuples having different missing attributes. Therefore, we define the following cluster generalization for such clusters.

Definition 7 (Cluster generalization). *Let's assume* $G^*_c(G_1, G_2, ..., G_n)$ *is a generalization of cluster* $C(Q_n)$*, then*

a) $g_i = [r_{i.min}, r_{i.max}]$*, where* $r_{i.min}(r_{i.max})$ *is the minimum(maximum) of the values of all tuples in* C *that have attribute values on* q_i*. If* q_i *is a numeric attribute.*

b) $g_i = H_{i.lowest}$ *where* $H_{i.lowest}$ *is the lowest common ancestor of the* v_{qi} *values of the tuples in cluster* C *that have values on* q_i*. If* q_i *is a categorical attribute.*

The quality of the anonymization algorithm is measured by the average information loss caused by the anonymization of the data stream. The Generalized Loss Metric[37] (*GLM* hereafter) is used by most data streams anonymization algorithms [11, 12, 14, 17, 32, 38] due to its precision and simplicity. Therefore, we define information loss of tuple and average information loss is defined in the definition 8 and definition 9.

Definition 8 (Information loss of tuple). *The information loss of anonymizing a tuple* $t(pid, Q_t)$ *to generalization* $G_t(g_1, g_2, ..., g_m)$ *is:*

$$InfoLoss(t, G_t) = \frac{1}{|G_t|}\left(\sum_{q_i \in Q_t} Loss(v_{qi})\right) \qquad (1)$$

Where $Loss(v_{qi})$ *is the information loss of* t *on QID* q_i *caused by the generalization, which is defined as:*

$$Loss(v_{qi}) = \begin{cases} \frac{r_{i.u} - r_{i.l}}{R_{i.u} - R_{i.l}} & if g_i \in [r_{i.l}, r_{i.u}] \\ \frac{|leaves(H_i)| - 1}{|leaves(DGH_i)| - 1} & if g_i = H \end{cases} \qquad (2)$$

Where $[r_{i.l}, r_{i.u}]$ *is the value domain of a numeric attribute* q_i DGH_i *is the domain graph hierarchy(DGH) of a categorical attribute* q_i*,* $|leaves(H_i)|$ *and* $|leaves(DGH_i)|$ *are the number of nodes of a tree rooted on* H_i *and* DGH_i*.*

Table 1: Example table for information loss measurement

	Age	**Gender**	**Education**
t_1 ***(Tuple)***	20	Male	Bachelor
G_1 ***(Generalization)***	[20-24]	Gender	University

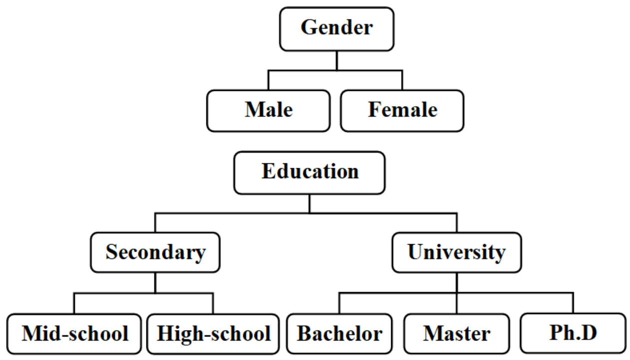


Figure 2: *DGH* of *Gender* and *Education*

In the following example, we demonstrate the calculation of information loss (see eq. 1) caused by cluster generalization (definition 7). Table 1 shows tuple t_1's values and it's generalization G_1. Let us assume that value domain of *Age* is $[0, 100]$, and *DGH* of *Gender* and *Education* is illustrated in Fig 2. Therefore, information loss of t_1 caused by generalization G_1 is: $InfoLoss(t_1, G_1) = (Loss(G_{Age}) + Loss(G_{Gender}) + Loss(G_{Education}))/3$. Using eq. 2, information loss of each *QID* is measured as follows:

$Loss(v_{Age}) = \frac{|24-20|}{|100-0|} = 0.04$

$Loss(v_{Gender}) = \frac{|leaves(Gender)|}{|leaves(Gender)|} = \frac{2}{2} = 1$

$Loss(v_{Education}) = \frac{|leaves(University)|}{|leaves(Education)|} = \frac{3}{7} = 0.428$

Therefore, information loss on t_1 caused by generalization G_1 is $InfoLoss(t_1, G_1) = (0.04 + 1 + 0.428)/3 = 0.489$.

Definition 9 (Average information loss). *The average information loss of a varied data stream of first N tuples is:*

$$AverageInfoLoss(N) = \frac{1}{N}\sum_{i=1}^{N} InfoLoss(t_i, G_i) \quad (3)$$

Where G_i is generalization of a tuple t_i.

4. *K*-anonymity for varied data streams via partitioning (*K*-VARP)

K-VARP anonymizes varied data streams under a time based sliding window. It has five parameters, VS is a varied data stream, K is the K-anonymity, δ is the time constraint of a sliding window, ω is the time constraint for re-using K-anonymous clusters, and R is the R-likeness criteria (please refer to definition 13 for more details). For the sake of simplicity, an abstract of working details of K-VARP is presented in Algorithm-1. Respective procedures of K-VARP are discussed in detail in the following text. K-VARP continuously receives tuples, and newly received tuples are assigned into respective partitions of S_p (definition 10) based on their QID set. Partition set S_p plays the role of a buffer. If there is no appropriate partition for a received tuple, a new partition is created and added to S_p based on their QIDs set Q_t. The received time of each tuple is recorded while assigning each tuple to a partition. Partitioning limits the number of tuples which may be involved in the KNN and this helps to reduce computation time, because, we need to perform a quick-sorting algorithm to find the nearest neighbors, and the time complexity of a quick sort is $O(nlog(n))$[39]. Saving computation time is important in respect of the performance. Also, partitioning helps to localize similar tuples for KNN, and this leads to less information loss and improve the usability of the data.

Algorithm 1 $K - VARP(VS, K, \delta, \omega, R)$

```
Let S_p be a set of partitions which will be used as a buffer, initialized empty;
Let S_k be a set of K-anonymous clusters which will be re-used, initialized empty;
while VS ≠ NULL do
    Read tuple t_i from VS and assign partition of S_p or create new partition for t_i;
    if Oldest tuple in buffer is expiring then
        TriggerPublish();
    end if
end while
while S_p ≠ NULL do
    TriggerPublish();
end while
```

Definition 10 (Partition on Q_p). *Let P be a set of tuples which only contains tuples with same QIDs* $P(Q_p) = \{t_1(pid_1, Q_p, ts_1),\ t_2(pid_2, Q_p, ts_2),\ \ldots,\ t_m(pid_m, Q_p, ts_m)\}$. *We call P is a partition on QIDs set* Q_p.

According to the sliding window time δ, an expiring tuple must get published. Each expiring tuple has its own anonymization round and $TriggerPublish()$ handles anonymization of an expiring tuple. The internal workings of $TriggerPublish()$ is explained in Algorithm-2.

Before anonymizing expiring tuples, $TriggerPublish()$ deletes expired K-anonymous re-usable clusters. As we discussed earlier in Section 1, an expiring tuple is processed through one of the following three procedures regarding their partition size. Let us assume that t' is an expiring tuple, and P' is a partition containing t'.

i. $InPartitionClustering(t', P')$ is a procedure which is called when P' has enough numbers of tuples to produce a K-anonymous cluster.

ii. $MergeClustering(t', P')$ is invoked when there are not enough tuples in P' to produce a cluster with miss-

Algorithm 2 $TriggerPublish()$

```
Delete expiring K-anonymous clusters from S_k using ω;
Let t' be a tuple stored in buffer for δ(expiring tuple) and P' be a partition containing t';
if |P'| ≥ K then
    InPartitionClustering(t', P');
end if
if |S_p| ≥ K then
    MergeClustering(t', P');
else
    SingleAnonymization(t', P');
end if
```

ing data, and the buffer has enough tuples to produce merged K-anonymous clusters.

iii. $SingleAnonymization(t', P')$ is called when the buffer does not have enough tuples to produce K-anonymous cluster. This method tries to anonymize and output t' with re-usable K-anonymous clusters. Otherwise, t' is published with suppression.

Each anonymized tuple is removed from its partition, and each published K-anonymous cluster is added to a re-usable cluster set S_k.

$InPartitionClustering(t', P')$ is illustrated in Algorithm-3. This method finds K-1 number of nearest neighbours of t' using distance function eq. 4. This method does not always generate K-anonymous cluster, if there is a previously anonymized cluster C' which can be used to publish t' with minimal information loss (see eq. 1) then $InPartitionClustering(t', P')$ publishes only t' using C', and t' is removed from P'. In contrast, if t' cannot be published with re-using, then new K-anonymous cluster is created by generalizing t' and it's K-1 neighbours according to definition 7.

Calculating the distance for KNN is the most important part of clustering. Since we are processing tuples with different QID sets, we only measure distance between

Algorithm 3 $InPartitionClustering(t', P')$

```
Find K − 1 nearest tuples to t' from P' and form a virtual cluster C'_p;
Find K-anonymous cluster C_k from S_k defined by P' has minimum information loss;
if C_k ≠ NULL then
    if InfoLoss(C'_p) ≥ Infoloss(C_k) then
        Use cluster generalization of C_k to publish t';
        Remove t' from P'
        RETURN;
    end if
end if
Anonymize and publish all tuples of C'_p and remove published tuples from P';
Add C'_p to S_k;
```

QIDs that are common between tuples. Based on the definition of information loss (see definition 8) we defined the distance between two tuples as follows:

Definition 11 (Distance between 2 tuples). *The distance between two tuples $t_1(pid, Q_1)$ and $t_2(pid, Q_2)$ is measured using QIDs received from both tuples that are the same.*

$$Distance(t_1, t_2) = \frac{\sum_{q_i \in |Q_1 \cap Q_2|} d_i(q_i)}{|Q_1 \cap Q_2|} \tag{4}$$

$$d_i(q_i) = \begin{cases} \frac{|r_{i.1} - r_{i.2}|}{|R_{i.u} - R_{i.l}|} & \textit{if } q_i \textit{ is numerical} \\ \frac{|leaves(H_i)|-1}{|leaves(DGH_i)|-1} & \textit{if } q_i \textit{ is categorical} \end{cases} \tag{5}$$

Where $r_{i.1}(r_{i.2})$ is the value of $t_1.q_i(t_2.q_i)$ if q_i is a numeric attribute, H_i is the lowest common ancestor of $t_1.q_i(t_2.q_i)$ with respect to DGH_i.

In the following example, we will demonstrate the calculation of distance function (see eq. 4) for both numeric and categorical QIDs. In Table 2 we showed two different tuples of varied data stream defined on

Age, Gender, Education, Height and *Weight* QIDs, and Fig. 2 we illustrated the DGH of *Gender* and *Education* attributes. Let us assume that the value domain of *Age, Height* and *Weight* are $[8, 100], [120, 200]$ and $[30, 120]$ respectively.

Table 2: Example table for distance measurement

No	**Age**	**Gender**	**Education**	**Height**	**Weight**
t_1	24	Male	Ph.D	*null*	85
t_2	20	Female	Bachelor	162	*null*

According to the distance function (see eq. 4) the distance of two tuples is calculated on QIDs which are common between tuples. In our example, t_1 and t_2 is common on *Age, Gender* and *Education* QIDs and $|Q_1 \cap Q_2| = 3$. Therefore, distance between t_1 and t_2 is:

$Distance(t_1, t_2) = {}^{(d(q_{Age})+d(q_{Gender})+d(q_{Education}))}/_{3}$

As we stated, value domain of QID *Age* is $[8, 100]$ then distance of QID *Age* is:

$d(q_{Age}) = \frac{|24-20|}{|100-8|} = 0.043$

Considering the DHG of QIDs (see Fig. 2) lowest common ancestor of *'Male'* and *'Female'* is *'Gender'*, and lowest common ancestor of *'Bachelor'* and *'Ph.D'* is *'University'*. Therefore, distance of *Gender* and *Education* QIDs are calculated as follows:

$d(q_{Gender}) = \frac{|leaves(Gender)|}{|leaves(Gender)|} = \frac{2}{2} = 1$

$d(q_{Education}) = \frac{|leaves(University)|}{|leaves(Education)|} = \frac{3}{7} = 0.428$

After calculating distance of each QID using eq. 5, the distance between t_1 and t_2 is:

$Distance(t_1, t_2) = {}^{0.043+1+0.428}/_{3} = 1.471/3 = 0.49$.

The distance measurement has disadvantage when both tuples have too few or zero common QIDs. However, K-VARP prevents this problem by applying two-phase partition selection (see section 4.1 for more details) in $MergeClustering(t', P', R')$.

4.1. Efficient merge-clustering

The $MergeClustering(t', P', R)$ is presented in Algorithm-4. This method performs clustering on multiple partitions. Highlighting part of this method is the partition merging criteria which helps to merge similar partitions to P' until it becomes eligible to produce K-anonymous cluster.

Definition 12 (Jaccard's similarity measurement). *Let P_1 and P_2 be similar partitions defined on Q_1 and Q_2 respectively. Their similarity is measured as:*

$$Jaccard(P_1, P_2) = \frac{|Q_1 \cap Q_2|}{|Q_1 \cup Q_2|} \tag{6}$$

Finding the most suitable candidates to merge in this dynamic environment is always complex. Also, the main challenge is to minimize calculation time and cluster information loss in this *best-from-current* situation. Selection process of the most merge-similar partition has two stages.

First, localization stage: K-VARP uses Jaccard's similarity coefficient (see eq. 6) [40] to find partitions set PS' which is most similar to P' in terms of description. Second, best selection stage: R-likeness is used to find the most suitable partition to merge with P' (see eq. 7).

As we discussed, *IoT Anonymization* [41] has unsupervised simple selection criteria that does not consider data distribution of partitions. It randomly chooses the most similar and biggest partition. Although, this simple selection criteria helps to reduce the impact of imputation in *IoT Anonymization*, it is not a feasible solution. The following example shows the drawback of simple merge selection criteria, and the advantage of R-likeness.

Definition 13 (R-likeness between tuple and partition). *Let R be a likeness measurement radius, P be a partition, t be a tuple. Then R-likeness between t and P is defined as:*

$$Likeness(P, t, R) = \sum_{t_i \in P} Radar(t, t_i, R) \tag{7}$$

$$Radar(t, t_i, R) = \begin{cases} 1 & Distance(t, t_i) \leq R \\ 0 & Distance(t, t_i) > R \end{cases} \tag{8}$$

Algorithm 4 $MergeClustering(t', P', R)$

Find $K-1$ cluster from S_k that can fully generalize t' with low information loss;
if $C_k \neq NULL$ **then**
    Use cluster generalization of C_k to publish t';
    Remove t' from P';
    RETURN;
end if
while $|P'| \geq K$ **do**
    Find non-empty partition P_{sim} from S_p which is most similar to P';
    Merge P_{sim} into P' and remove P_{sim} from S_p;
end while
Find $K-1$ nearest tuple to t' from P' and form a virtual cluster C'_m that has missingness;
Anonymize and publish C'_m;
Remove published tuples from P';
Re-assign remaining tuples of P' to respective partitions;

Algorithm 5 $SingleAnonymization(t', P')$

Find K-anonymous cluster C_k from S_k which covers t' with minimum information loss
if C_k found **then**
    Use cluster generalization of C_k to publish t';
else
    Suppress t' and publish
end if
Remove t' from P'

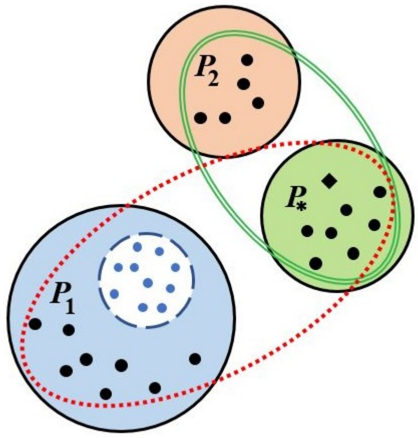


Figure 3: Stage of merge selection.

Let us assume that $K = 10$, and P_* has an expiring tuple (illustrated as a diamond in Fig. 3), shows the merge selection stage after the localization stage. P_1 and P_2 are the most suitable partitions that can be merged with P_*. Also, the circle with the dash represents previously anonymized data points of P_1. According to *IoT Anonymization* P_1 is the most suitable partition to merge; however, if we consider the data distribution of each partition, tuples in P_2 are tightly packed *i.e.*, attribute values of tuples are very similar among available attributes. Thus, P_2 is a better option to merge with P_*. Another supporting point for choosing P_2 is that P_1 has eight tuples and there is a high possibility of executing $InPartitionClustering$ (see Algorithm-3) on P_1.

4.2. Flexible re-using

IoT Anonymization re-using is available only to tuples and clusters having the same QID set which is restrained by a mechanism of imputation. If we publish merged cluster data using imputation, we cannot re-use imputed clusters, since imputed values are published and it is not possible to change information after publication. On the other hand, K-VARP has a flexible re-using strategy; it keeps clusters from $MergeClustering$ (Algorithm-4) for re-use. Therefore, the re-using rule is relaxed. Both expiring tuples and re-using K-anonymous clusters do not need to have exactly the same QID set.

Table 3: 3-anonymous cluster

No	Age	Gender	Height	Weight
t_1	18	Male	[168-175]	80
t_2	20	Male	[168-175]	*null*
t_3	19	*null*	[168-175]	74

It is possible to apply re-using for expiring tuples if all attributes are covered by re-usable cluster generalization. Let us explain the difference between both re-using strategies in the following example. Table 3, demonstrates tuples assigned to a 3-anonymous cluster, *null* represents the missing values. t_2 has a missing value for attribute $Weight$, and t_3 has no value for attribute $Gender$.

Table 4: IoT Anonymization on Table 3

No	Age	Gender	Height	Weight
t_1	[18-20]	Male*(3,2)*	[168-175]	[74-80]*(3,2)*
t_2	[18-20]	Male*(3,2)*	[168-175]	[74-80]*(3,2)*
t_3	[18-20]	Male*(3,2)*	[168-175]	[74-80]*(3,2)*

If we anonymize Table 3 using *IoT Anonymization*, each tuple of a corresponding table will be published using generalization $G_1([18-20], Male\textit{(3,2)}, [168-175], [74-80]\textit{(3,2)})$. In addition, G_1 cannot be re-used due to the imputation on $Gender$ and $Weight$ attributes according to the limitation of *IoT Anonymization*. Also, Table 4 is hard to analyze, because of the uncertainty that was created by imputation.

Table 5: K-VARP on Table 3

No	Age	Gender	Height	Weight
t_1	[18-20]	Male	[168-175]	[74-80]
t_2	[18-20]	Male	[168-175]	*null*
t_3	[18-20]	*null*	[168-175]	[74-80]

On the other hand, if we use K-VARP $G_2([18-20], Male, [168-175], [74-80])$ is a cluster generalization which is kept for re-using. Table 5 demonstrates an anonymized version of Table 3. It is worth mentioning that, this flexible re-using increases the possibility of re-use anonymization which reduces computation time and number of suppressions.

4.3. Complexity analysis of K-VARP

The time complexity of K-VARP is analyzed in this section.

$InPartitionClustering(t', P')$ (Algorithm-3), step 1 needs $O(|P'|log|P'||QID|)$ with quick sort algorithm applied, step 2 requires $O(|S_k||QID|)$, both step 4, 5, 6 and 11 costs $O(|QID|)$ time, and step 10 costs $O(|QID|K)$. Hence, time complexity of procedure $InPartitionClustering(t', P')$ is $O((|P'|log|P'| + |S_k| + K)|QID|) = O(|P'|log|P'||QID|)$.

$MergeClustering(t', P', R)$ (Algorithm-4), step 1 costs $O(|S_k||QID|)$ time, step 3 costs $O(|QID|)$ time. $While\ loop$ from step 7 to 10 runs at most K-1 times, and each iteration costs $O(|S_p||QID|)$, thus, calculation cost of the *while loop* is $O(K|S_p||QID|)$. Using quick sort algorithm, step 11 costs $O(|P'|log|P'||QID|)$, and step 12 to 14 needs $O(|P'||QID|)$ time. Total running time of $MergeClustering(t', P', R)$ is $O((K|S_p| + |P'|log|P'|)|QID|)$.

$SingleAnonymization(t', P')$ (Algorithm-5), step 1 costs $O(|S_k||QID|)$, step 3, 5 and 7 each costs $O(|QID|)$, time complexity of $SingleAnonymization(t', P')$ is $O(|S_k||QID|)$.

In K-VARP (Algorithm-1) buffer size is bounded with sliding window constraint. Therefore, time complexity of $InPartitionClustering(t', P')$ is $O(\delta log\delta|QID|)$, $MergeClustering(t', P', R)$ is $O(\delta log\delta|QID|)$, and $SingleAnonymization(t', P')$ is $O(\delta|QID|)$ if we consider δ in the equation.

In the main procedure of $K-VARP(VS, K, \delta, \omega, R)$, step 4 costs $O(|S_p||QID|)$, and *while loop* is called at most $|VS| - \delta$ times. Thus, time complexity of *while loop* step 3 to 8 is $O(|S_p||VS||QID|)$, after the first loop $\delta - 1$ tuples are left and the time complexity of *while loop* step 9 to 11 is $O(K\delta^2|QID|)$. Therefore, the time complexity of $K-VARP(VS, K, \delta, \omega, R)$ is $O(|VS|K\delta|QID|)$. Size of data stream is potentially infinite and δ and K are considerably smaller compared to $|VS|$, thus, time complexity of K-VARP can be $O(|VS|)$.

5. Experimental evaluation

In order to estimate the performance of K-VARP, we compared it with *IoT Anonymization* [41] and *FADS* [17]. We used the *adult*[1] and *PM2.5 Data of Five Chinese cities*[2] datasets from the *UCI machine learning repository.* The *Adult* dataset is widely used to evaluate efficacy of anonymization algorithms [6, 11, 12, 14, 17, 32, 30, 41, 42, 38]. The selected attributes are: $education, marital-status, work-class, occupation, relationship, race, gender, country$ and $age, final-weight, education-number, capital-gain, capital-loss$ and $hours-per-week$ where eight attributes are categorical and six are numeric. The generalization hierarchy of eight categorical attributes are defined in [16], a brief description of the *Adult* dataset is explained in Table 6.

Table 6: *QID* descriptions of *Adult* dataset

Attribute name	**Type**	**Range**	
		Min	**Max**
Age	*Numeric*	17	90
Final-weight	*Numeric*	13769	1484705
Education-number	*Numeric*	1	16
Capital-gain	*Numeric*	0	99999
Capital-loss	*Numeric*	0	4356
Hours-per-week	*Numeric*	1	99
		Hierarchy tree	
		Height	**Nodes**
Education	*Categorical*	5	26
Marital-status	*Categorical*	4	11
Work-class	*Categorical*	5	13
Country	*Categorical*	4	62
Occupation	*Categorical*	3	15
Relationship	*Categorical*	3	7
Rage	*Categorical*	3	6
Gender	*Categorical*	2	3

PM2.5 Data of Five Chinese cities ($PM2.5$ hereafter) is a dataset of meteorological information of five big cities in China which includes sensory data. We merged five separate city data to create a large IoT data stream. The selected attribures of $PM2.5$ are: $season$, $wind-direction(combined-wind-direction)$ and $first-post(PM2.5)$, $second-post(PM2.5)$, $third-post(PM2.5)$, $dew-point$, $temperature$, $humidity$, $pressure$, $wind-speed(cumulated-wind-speed)$, $h-precipitation(hourly-precipitation)$, $c-precipitation(cumulated-precipitation)$ where two attributes are categorical and ten are numeric. The generalization hierarchy of two categorical attributes are illustrated in Fig. 4, a brief description of the $PM2.5$ dataset is explained in Table 7.

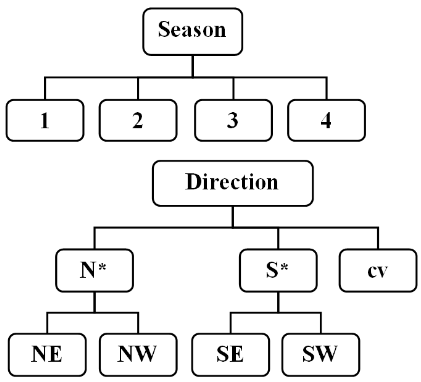


Figure 4: DGH tree of PM2.5 dataset

To compare our algorithm with existing data stream anonymization algorithm, we modified *FADS* [17] for varied data stream anonymization. *FADS* suppresses tuples when it does not find any suitable cluster for an expir-

[1] http://archive.ics.uci.edu/ml/datasets/Adult

[2] https://archive.ics.uci.edu/ml/datasets/PM2.5+Data+of+Five+Chinese+Cities

ing tuple. However, in varied data stream settings, for $FADS$ we hold expired tuples until they get anonymized in clusters, and this causes late anonymization. A late anonymized tuple loses its usability due to the time sensitivity of a sliding window. To measure information loss correctly, we used information loss measurements that applies a late anonymization penalty which is used in [38]. The average information loss of K-VARP is calculated according to definition 9.

Table 7: QID descriptions of $PM2.5$ dataset

Attribute name	**Type**	**Range**	
		Min	**Max**
First-post	*Numeric*	1	1528
Second-post	*Numeric*	1	940
Third-post	*Numeric*	1	968
Dew-point	*Numeric*	-40	28
Temperature	*Numeric*	-25	41
Humidity	*Numeric*	2	100
Pressure	*Numeric*	975	1042
Wind-speed	*Numeric*	0	608
H-precipitation	*Numeric*	0	61.6
C-precipitation	*Numeric*	0	226.4
		Tree	
		Height	**Nodes**
Season	*Categorical*	3	8
Wind-Direction	*Categorical*	2	5

To create a continuous and consistent flow of data streams, tuples were received from the dataset with a delay of 500 microseconds. The experiment parameters are shown in Table 8, where K represents K-anonymity, δ is the time constraint of the sliding window, ω is the time constraint of re-usable K-anonymous clusters, α is a late anonymization penalty coefficient, and R is the R-likeness coefficient.

We randomly added missing values to the original datasets to create a varied data streams. Tuples with

Table 8: Parameters of experiment

Algorithm name	**Parameters**
FADS	K=50, δ=2000,ω=200,α=0.001
IoT Anonymization	K=50, δ=2000, ω=200
K-VARP	K=50, δ=2000, ω=200, R=0.2

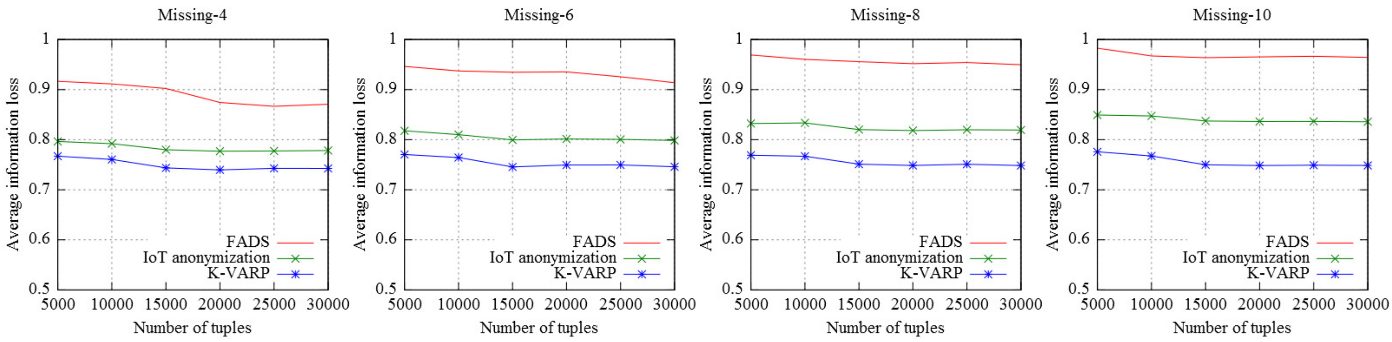
different numbers of missing values are the same in all datasets. As an example, Table 9 shows a description of dataset containing at most three missing values of the *Adult* dataset. The entry with data size of 30000 with maximum $3 - missing$ in each tuple, shows tuples with same number of QIDs all equal to 7500. In experiment graphs the $Missing - X$ indicates that each tuple in the dataset has at most X number of missing values. To maintain the validity of the tests, all the algorithms were implemented in Java. The experiments were conducted on a PC with Intel i5-4590 CPU (3.30GHz) with 8GB RAM and Windows 10x64 with JDK 8.0.

Table 9: Data set description *(At most missing-3 values (Adult dataset))*

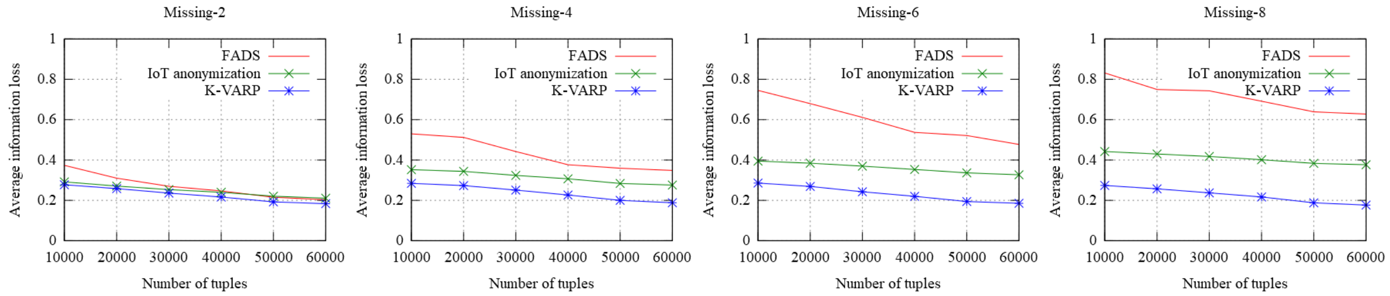
Data size	**Number of tuples with same number of missing values**			
	Missing-0	Missing-1	Missing-2	Missing-3
5000	1250	1250	1250	1250
10000	2500	2500	2500	2500
15000	3750	3750	3750	3750
20000	5000	5000	5000	5000
25000	6250	6250	6250	6250
30000	7500	7500	7500	7500

5.1. Information loss

In Fig. 5 the average information loss of $FADS$, *IoT Anonymization* and K-VARP are illustrated by the varying missingness amount on *Adult* and $PM2.5$ dataset respectively. Fig. 5 shows that K-VARP anonymizes with less information loss compared to the other two ap-

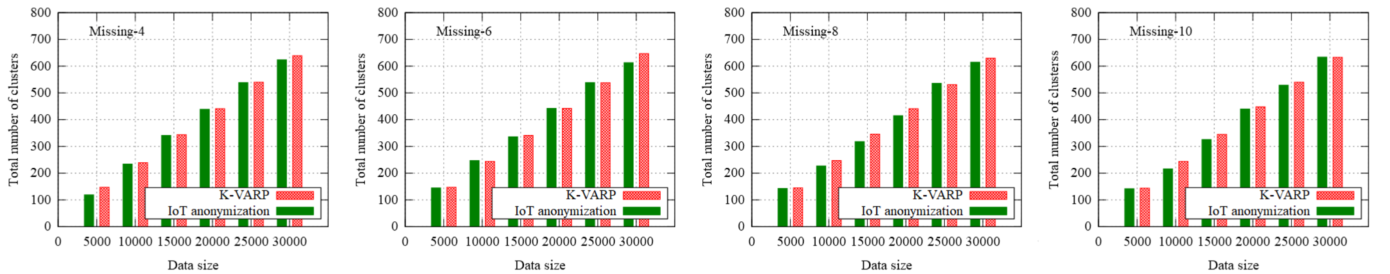


(a) Information loss (Adult dataset)

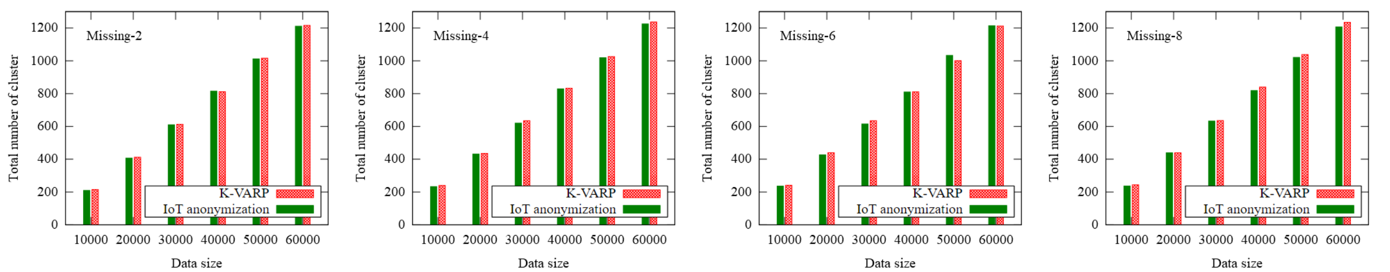


(b) Information loss (PM2.5 dataset)

Figure 5: Information loss varying missingness



(a) Clusters created (Adult dataset)



(b) Clusters created (PM2.5 dataset)

Figure 6: Number of clusters created

proaches. The difference of information loss between K-VARP and others increases when the receiving data has more missing values.

An increase in missingness of data decreases the average number of tuples in partitions. This leads to a greater number of clustering with merging partitions for

IoT Anonymization, and more tuple holding for *FADS*. To achieve less information loss, *IoT Anonymization* has to minimize clustering with merge. Whereas for *FADS*, to reduce the number of late anonymization, the receiving data must be assigned to a fewer number of partitions. However, these conditions are all violated when the missingness of data increases. On the other hand, K-VARP is not sensitive to the decrease of average numbers of tuples in partitions due to its merge-clustering criteria and flexible re-using. This significantly helps to reduce information loss even with the increase of missingness of data, as compared to *IoT Anonymization* and *FADS*.

In Fig. 5(a) we can see that the information loss difference of K-VARP and *IoT Anonymization* is between 3% to 9% which indicates the advantage of K-VARP. However, in Fig. 5(b) we can also see that K-VARP has significantly lower information loss compared to other algorithms, by resulting 10% to 20% less information loss compared to *IoT Anonymization*.

Information loss on two datasets shows similar figure but considerable difference, and this is caused by data distribution of datasets and the curse of dimensionality [43] amplified by the amount of missingness. The *Adult* dataset have fourteen (eight categoric, six numeric) and the $PM2.5$ have twelve (two categoric and ten numeric) number of QIDs, and missingness amounts of *Adult* are higher than $PM2.5$ (see Fig. 5) in the experiments. Therefore, algorithms are expected to result more information loss on *Adult*. Nevertheless, from these results, we can see that K-VARP has performed significantly better which indicates the scalability and efficiency of the algorithm.

5.2. Clustering

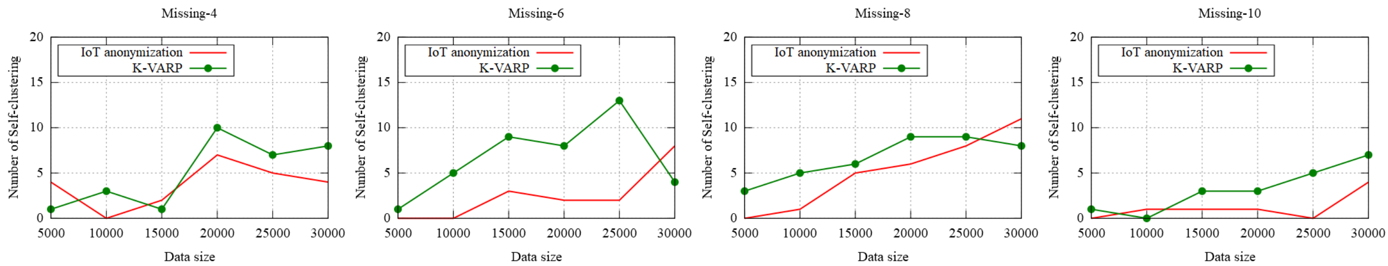
To achieve K-anonymity when a single partition does not contain enough tuples, K-VARP and *IoT Anonymization* merge two or more partitions to create a single cluster satisfying anonymity. For the given datasets (see Table 6 and Table 7) K-VARP and *IoT Anonymization* created an almost similar number of clusters (see Fig 6). Since the experiment is performed on the same datasets under a consistent environment, both algorithms employ a similar time to creating K-anonymous clusters. However, the flexible merging criteria of K-VARP prevented the creation of more clusters, but increased the chance of suppression, resulting in a similar number of clusters.

In Fig. 7 we show the number of self-clustering for K-VARP and *IoT Anonymization*. Despite the few drops, K-VARP performed more self-clustering compared to *IoT Anonymization* in Fig. 7(a), and a fewer number of self-clustering in Fig. 7(b).

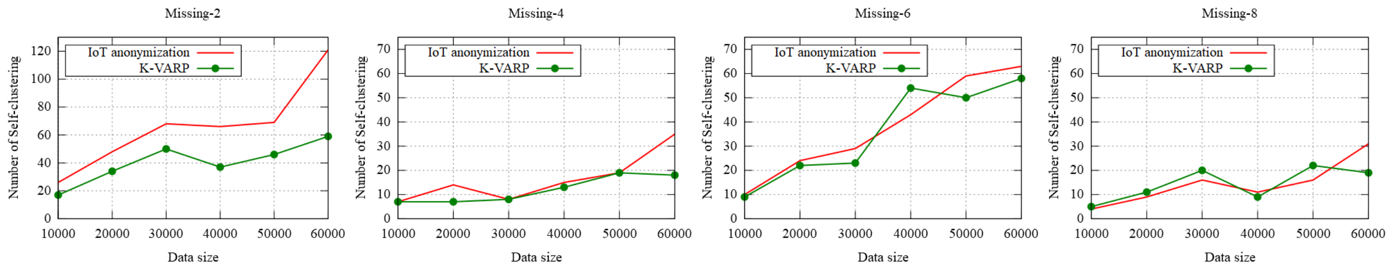
IoT Anonymization has a merge selection criteria that tends to minimize imputation by limiting the number of different partitions involved in merge-clustering. Therefore, the fewer number of clusters merging caused greater numbers of self-clustering. On the other hand, K-VARP only considers the data distribution of partitions when merging clusters, and this merging rule leads to more partitions merging for K-VARP which decreases the number of self-clustering.

K-VARP has a flexible cluster re-using strategy which allows more tuples to be anonymized with re-using. This decreases the number of clustering using KNN and the merging operation of K-VARP, thus decreasing execution time of K-VARP. In contrast to K-VARP, the *IoT Anonymization*'s re-using strategy is very strict only allowing tuples to re-use K-anonymous clusters from their own partitions. This leads to the creation of new clusters for expiring tuples, thus increasing the execution time.

In Fig. 8 we have demonstrated the number of clusters which are re-used during anonymization. When data has lesser amounts of missingness, then, re-using occurs more often. Also, self-clustering occurs more frequent if data has lesser amounts of missing values, and this shortens the time gap between clustering operations, leading to more re-usable K-anonymous clusters being stored, result-

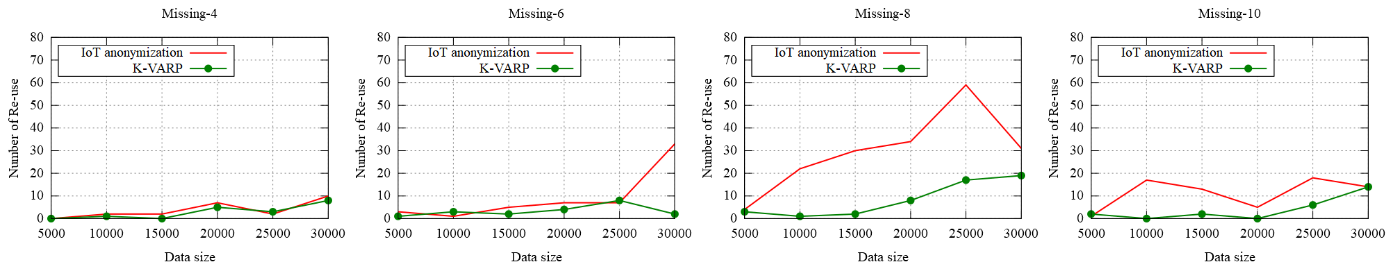


(a) Self-clustering (Adult dataset)

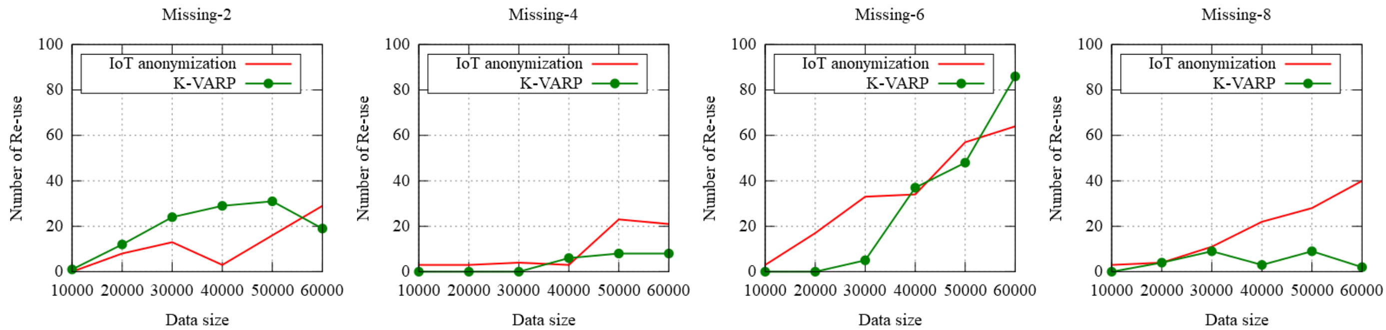


(b) Self-clustering (PM2.5 dataset)

Figure 7: Number of self clustering



(a) Re-using (Adult dataset)



(b) Re-using (PM2.5 dataset)

Figure 8: Number of re-use

ing in more numbers of re-using. Except the unusual result on 8(b), overall result of re-using shows that K-VARP has a lesser number of re-using than *IoT Anonymization*. The greater number of re-using reduces calculation time; however, it is not guaranteed to reduce information loss.

Fig 9. illustrates number of suppression for K-VARP and *IoT Anonymization*. From the figures we can see that, when the number of partitions of varied data streams increases, suppression occurs more often for K-VARP. *IoT Anonymization* tends to combine the biggest partitions when performing merge-clustering for expiring tuples, maintaining less number of partitions in the buffer. On the contrary, K-VARP leaves more number of smaller partitions after merging which leads more number of suppression. The overall figures show more suppression is performed on K-VARP compared to *IoT Anonymization*.

5.3. Runtime

Fig. 10 demonstrates the runtime of K-VARP, *IoT Anonymization* and $FADS$. The improved merging criteria of K-VARP increases the computation time to perform merge clustering. This is because K-VARP's merging stage spends more time calculating R-likeness compared to *IoT Anonymization* and $FADS$. Also, the number of partitions merged in single merge-clustering is generally higher in K-VARP. However, the runtimes of these algorithms are reasonably comparable. The experiment graphs shows that, K-VARP spent approximately zero to five percent more time on anonymization depending on the amount of missingness and the data size. Although, there is a slight increase in runtime, this does not adversely affect the overall performance because information loss is decreasing rapidly. Altogether, our algorithm outperforms conventional algorithms.

6. Conclusion and future direction

In this paper, we presented K-VARP, a novel algorithm to anonymize varied data streams. It uses a time based sliding window technique to partition tuples based on their description. This preliminary operation helps to form clusters faster by localizing tuples and merging the relevant partitions when required. It is necessary to merge similar partitions to anonymize tuples with less uncertainty, and in this situation, a marginalization with flexible re-using strategy is a convenient and scalable approach. K-VARP outperformed both *IoT Anonymization* and $FADS$ conventional data stream anonymization approach is adopted for varied data streams. The results demonstrated the effectiveness of K-VARP as it uses R-likeness to identify similar partitions upon merging. Moreover, a combination of marginalization and flexible re-using has a significant impact for anonymizing varied data streams, K-VARP anonymizes varied data streams with 3-9% less information loss on *Adult* and 10-20% less information loss on $PM2.5$ compared to the other two algorithms while spending similar time for computation. Data usability of K-VARP is better than the other two algorithms because, our proposed algorithm does not impute missing values and impractical late anonymization.

For future work we envision the optimization of merge clustering for partitioning based varied data stream anonymization. Finally, we will study the application of K-VARP for smart and connected spaces that have limited data streams. The challenge will be to maintain data privacy and usability through anonymization while having fewer numbers of streaming tuples.

Acknowledgement

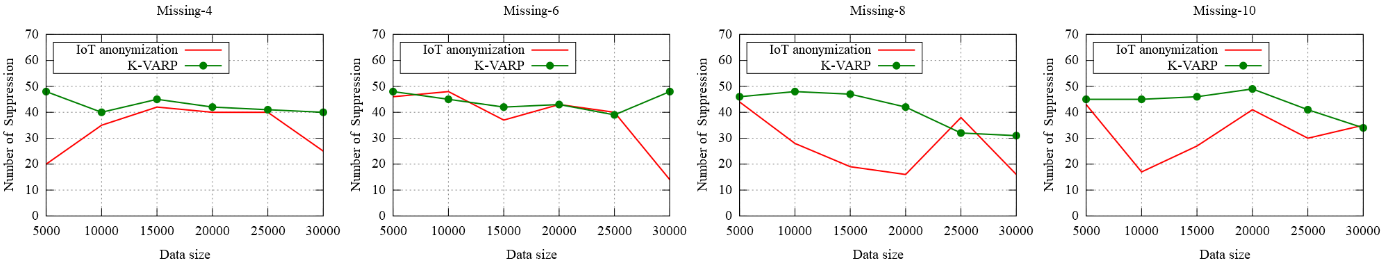
The first author would like to acknowledge the support provided by the EU Erasmus Mundus project gLINK (552099-EM-1-2014-1-UK-ERA) to carry out this research at the University of the West of Scotland, UK.

References

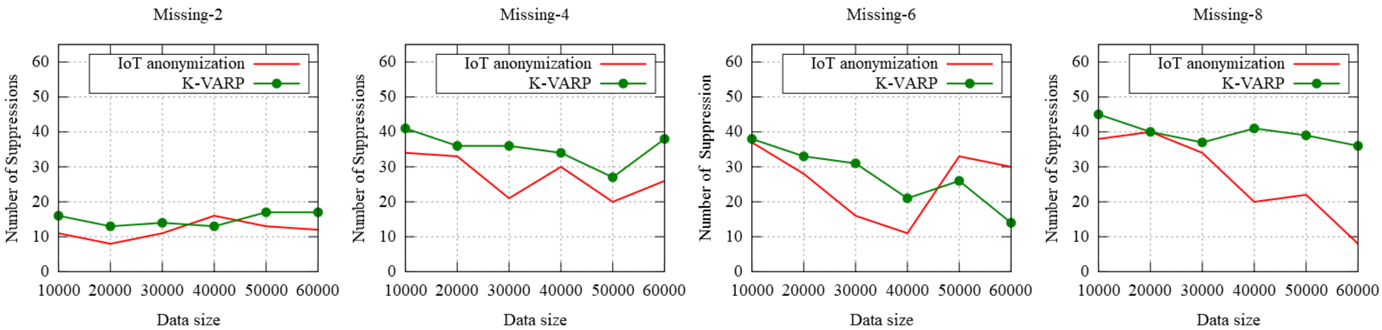
[1] Okada K, Nishi H. Big data anonymization method for demand response services. In: Proceedings on the International Conference on Internet Computing (ICOMP). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp); 2014, p. 1.

[2] Gubbi J, Buyya R, Marusic S, Palaniswami M. Internet of things (iot): A vision, architectural elements, and future directions. Future generation computer systems 2013;29(7):1645–60.

[3] He W, Yan G, Da Xu L. Developing vehicular data cloud services in the iot environment. IEEE Transactions on Industrial Informatics 2014;10(2):1587–95.

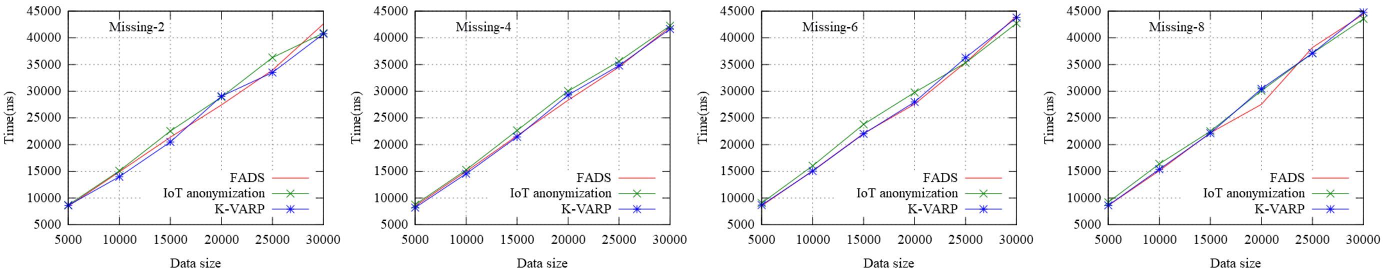


(a) Suppressions(Adult dataset)

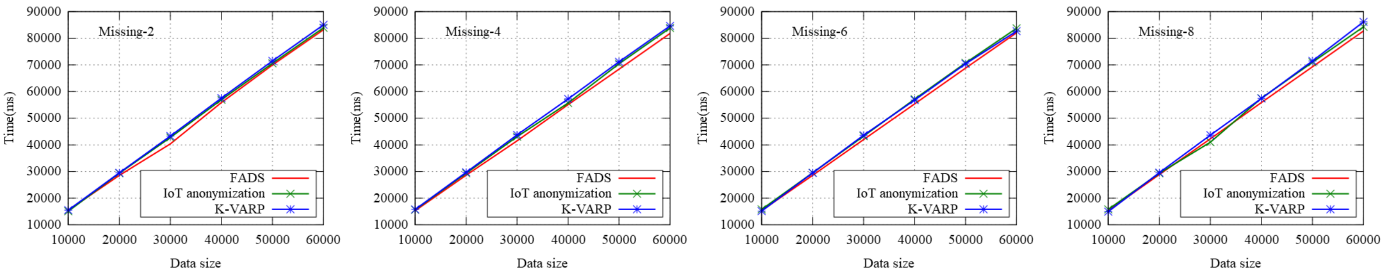


(b) Suppressions(PM2.5 dataset)

Figure 9: Number of suppression



(a) Runtime(Adult dataset)



(b) Runtime(PM2.5 dataset)

Figure 10: Runtime of algorithms

[4] Zhang X, Yang LT, Liu C, Chen J. A scalable two-phase top-down specialization approach for data anonymization using mapreduce on cloud. IEEE Transactions on Parallel and Distributed Systems 2014;25(2):363–73.

[5] Zhang X, Liu C, Nepal S, Yang C, Dou W, Chen J. Combining top-down and bottom-up: scalable sub-tree anonymization over big data using mapreduce on cloud. In: Trust, Security and Privacy in Computing and Communications (TrustCom), 2013

12th IEEE International Conference on. IEEE; 2013, p. 501–8.

[6] Sweeney L. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 2002;10(05):557–70.

[7] LeFevre K, DeWitt DJ, Ramakrishnan R. Incognito: Efficient full-domain k-anonymity. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM; 2005, p. 49–60.

[8] Li N, Li T, Venkatasubramanian S. t-closeness: Privacy beyond k-anonymity and l-diversity. In: Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on. IEEE; 2007, p. 106–15.

[9] Machanavajjhala A, Kifer D, Gehrke J, Venkitasubramaniam M. L-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) 2007;1(1):3.

[10] Truta TM, Vinay B. Privacy protection: p-sensitive k-anonymity property. In: Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on. IEEE; 2006, p. 94–.

[11] Zakerzadeh H, Osborn SL. Faanst: fast anonymizing algorithm for numerical streaming data. In: Data privacy management and autonomous spontaneous security. Springer; 2011, p. 36–50.

[12] Wang P, Lu J, Zhao L, Yang J. B-castle: An efficient publishing algorithm for k-anonymizing data streams. In: Intelligent Systems (GCIS), 2010 Second WRI Global Congress on; vol. 2. IEEE; 2010, p. 132–6.

[13] Li J, Ooi BC, Wang W. Anonymizing streaming data for privacy protection. In: Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. IEEE; 2008, p. 1367–9.

[14] Cao J, Carminati B, Ferrari E, Tan KL. Castle: A delay-constrained scheme for k s-anonymizing data streams. In: Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on. IEEE; 2008, p. 1376–8.

[15] Wang W, Li J, Ai C, Li Y. Privacy protection on sliding window of data streams. In: Collaborative Computing: Networking, Applications and Worksharing, 2007. CollaborateCom 2007. International Conference on. IEEE; 2007, p. 213–21.

[16] Zhou B, Han Y, Pei J, Jiang B, Tao Y, Jia Y. Continuous privacy preserving publishing of data streams. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. ACM; 2009, p. 648–59.

[17] Guo K, Zhang Q. Fast clustering-based anonymization approaches with time constraints for data streams. Knowledge-Based Systems 2013;46:95–108.

[18] Kim S, Sung MK, Chung YD. A framework to preserve the privacy of electronic health data streams. Journal of biomedical informatics 2014;50:95–106.

[19] Al-Hussaeni K, Fung BC, Cheung WK. Privacy-preserving trajectory stream publishing. Data & Knowledge Engineering 2014;94:89–109.

[20] Sarkar M, Leong TY. Fuzzy k-means clustering with missing values. In: Proceedings of the AMIA Symposium. American Medical Informatics Association; 2001, p. 588.

[21] Graham JW. Analysis of missing data. In: Missing data. Springer; 2012, p. 47–69.

[22] Wagstaff K. Clustering with missing values: No imputation required. Classification, Clustering, and Data Mining Applications 2004;:649–58.

[23] Liu Y, Nie L, Han L, Zhang L, Rosenblum DS. Action2activity: Recognizing complex activities from sensor data. In: IJCAI; vol. 2015. 2015, p. 1617–23.

[24] Liu Y, Zhang L, Nie L, Yan Y, Rosenblum DS. Fortune teller: Predicting your career path. In: AAAI; vol. 2016. 2016, p. 201–7.

[25] Yang Y, Zheng X, Guo W, Liu X, Chang V. Privacy-preserving smart iot-based healthcare big data storage and self-adaptive access control system. Information Sciences 2018;.

[26] Kim Y, Lee S. Energy-efficient wireless hospital sensor networking for remote patient monitoring. Information Sciences 2014;282:332–49.

[27] De Paz JF, Bajo J, Rodríguez S, Villarrubia G, Corchado JM. Intelligent system for lighting control in smart cities. Information Sciences 2016;372:241–55.

[28] Tao M, Ota K, Dong M. Locating compromised data sources in iot-enabled smart city: A great-alternative-region-based approach. IEEE Transactions on Industrial Informatics 2018;.

[29] Babcock B, Datar M, Motwani R. Sampling from a moving window over streaming data. In: Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics; 2002, p. 633–4.

[30] Mohammadian E, Noferesti M, Jalili R. Fast: fast anonymization of big data streams. In: Proceedings of the 2014 International Conference on Big Data Science and Computing. ACM; 2014, p. 23.

[31] Bhatia N, et al. Survey of nearest neighbor techniques. arXiv preprint arXiv:10070085 2010;.

[32] Zakerzadeh H, Osborn SL. Delay-sensitive approaches for anonymizing numerical streaming data. International journal of information security 2013;12(5):423–37.

[33] Sakpere AB, Kayem AV. Adaptive buffer resizing for efficient anonymization of streaming data with minimal information loss. In: Information Systems Security and Privacy (ICISSP), 2015 International Conference on. IEEE; 2015, p. 1–11.

[34] Ciglic M, Eder J, Koncilia C. Anonymization of data sets

with null values. In: Transactions on Large-Scale Data-and Knowledge-Centered Systems XXIV. Springer; 2016, p. 193–220.

[35] Jia X, Pan C, Xu X, Zhu KQ, Lo E. ρ-uncertainty anonymization by partial suppression. In: DASFAA (2). 2014, p. 188–202.

[36] Cao J, Karras P, Raïssi C, Tan KL. ρ-uncertainty: inference-proof transaction anonymization. Proceedings of the VLDB Endowment 2010;3(1-2):1033–44.

[37] Iyengar VS. Transforming data to satisfy privacy constraints. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM; 2002, p. 279–88.

[38] Zhang J, Yang J, Zhang J, Yuan Y. Kids: K-anonymization data stream base on sliding window. In: Future Computer and Communication (ICFCC), 2010 2nd International Conference on; vol. 2. IEEE; 2010, p. V2–311.

[39] Cormen TH. Introduction to algorithms. MIT press; 2009.

[40] Leskovec J, Rajaraman A, Ullman JD. Mining of massive datasets. Cambridge university press; 2014.

[41] Ankhbayar Otgonbayar Zeeshan Pervez KD. Toward anonymizing iot data streams via partitioning. In: Mobile Ad Hoc and Sensor Systems (MASS), 2016 IEEE 13th International Conference on. IEEE; 2016, p. 331–6.

[42] Fung BC, Wang K, Yu PS. Top-down specialization for information and privacy preservation. In: Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on. IEEE; 2005, p. 205–16.

[43] Marimont R, Shapiro M. Nearest neighbour searches and the curse of dimensionality. IMA Journal of Applied Mathematics 1979;24(1):59–70.