Privacy-Preserving for Distributed Data Streams: Towards l-Diversity

Mona Mohamed, Sahar Ghanem, and Magdy Nagi
Computer and Systems Engineering Department, Alexandria University, Egypt

Abstract: *Privacy-preserving data publishing have been studied widely on static data. However, many recent applications generate data streams that are real-time, unbounded, rapidly changing, and distributed in nature. Recently, few work addressed k-anonymity and l-diversity for data streams. Their model implied that if the stream is distributed, it is collected at a central site for anonymization. In this paper, we propose a novel distributed model where distributed streams are first anonymized by distributed (collecting) sites before merging and releasing. Our approach extends Continuously Anonymizing STreaming data via adaptive cLustEring (CASTLE), a cluster-based approach that provides both k-anonymity and l-diversity for centralized data streams. The main idea is for each site to construct its local clustering model and exchange this local view with other sites to globally construct approximately the same clustering view. The approach is heuristic in a sense that not every update to the local view is sent, instead triggering events are selected for exchanging cluster information. Extensive experiments on a real data set are performed to study the introduced Information Loss (IL) on different settings. First, the impact of the different parameters on IL are quantified. Then k-anonymity and l-diversity are compared in terms of messaging cost and IL. Finally, the effectiveness of the proposed distributed model is studied by comparing the introduced IL to the IL of the centralized model (as a lower bound) and to a distributed model with no communication (as an upper bound). The experimental results show that the main contributing factor to IL is the number of attributes in the quasi-identifier (50%-75%) and the number of sites contributed about 1% and this proves the scalability of the proposed approach. In addition, providing l-diversity is shown to introduce about 25% increase in IL when compared to k-anonymity. Moreover, 35% reduction in IL is achieved by messaging cost (in bytes) of about 0.3% of the data set size.*

Keywords: *k-anonymity, l-diversity, data streams and clustering.*

Received April 20, 2017; accepted December 18, 2017
https://doi.org/10.34028/iajit/17/1/7

1. Introduction and Background

Many applications such as social networks, sensor networks, health-care, market-basket analysis, network monitoring and cloud-based services produce big data. Big data with time stamp is called data stream. For organizations to release such data gathering, that contain person specific information, they are obligated by privacy policy to remove all explicit identifiers, such as name, Social Security Number (SSN) and address. Usually released data are outsourced to a third party for research or data-mining purpose. The resulting data after *suppressing* identifying information looks anonymous. However, the data is still vulnerable to linking attacks. That is the remaining data can be used to re-identify individuals by linking to other data.

For example if Table 1 is released by an insurance company that has all explicit identifiers removed. The remaining table has four attributes: the zip-code (part of the address), date-of-birth, gender and diagnosis. The diagnosis is a sensitive attribute that need to be protected such that observers (including attackers) should not be able to link a specific person to his diagnosis. However, if an attacker has background knowledge that a specific person is in the table and is able to obtain that person (zip-code, data-of-birth, gender), he can uniquely link him to his diagnosis. For example, an attacker who knows Ali has visited a hospital and knows Ali's first three attributes are (53723, 02/05/1973, Male) (e.g., obtained from a voter registration table). Then the attacker can search the released table and find a unique record (second row) that can be linked to Ali, and conclude "Ali is diagnosed with heart disease".

The goal of privacy-preserving data publishing is to limit the ability to link released information to other external collections. The attributes that in combination can uniquely identify individuals (with high probability) are called Quasi-Identifier (QI), such as the three attributes (zip-code, birth-date, gender). QI needs a domain expert to decide, in addition policies and contracts can help.

One solution to the linking attack is k-anonymity [18]. To achieve k-anonymity, generalization of QI attributes is applied such that each combination of QI values occurs at least k times. In the generalization process, a QI attribute value is replaced by a less specific but semantically related value. The set of released tuples which have the same generalized QI are called an equivalence class. All equivalence classes should be of size at least *k*. In this approach, a

record QI is indistinguishable from other k - 1 records and the probability of linking attack is reduced to 1/k. Note that, Suppression and generalization reduce the quality of the data and an information loss metric need to be minimized during the anonymization process. Information loss is defined as the amount of uncertainty introduced by the generalization procedure.

For example, Table 2 is 3-anonymized version of Table 1, where the QI is (zip-code, date-of-birth, gender) and k=3. Zip-code is generalized by suppressing the first digit or two. Date-of-birth is generalized by year-of-birth or year interval. Note that, for each combination of QI there are at least 3 occurrences of those values in Table 2.

While k-anonymity protects against identity disclosure where a person cannot be linked to a particular record in the released table. It is insufficient to prevent *attribute disclosure*. For example, in Table 2 the "diagnosis" attribute is sensitive. If an attacker knows that Omar's data is in the table and he is born in 1979 and lives in part of the address (ZIP) 53724. From Table 2, the attacker can conclude that Omar corresponds to one of the first three records, and thus must have "Heart Disease". This is known as homogeneity attack. While Omar can't be linked to exactly one record in the released table (identity disclosure), an attacker still can conclude new information about Omar (attribute disclosure).

Table 1. Health insurance table.

ZIP code	Date of Birth	Gender	Diagnosis
53712	01/01/1970	M	Heart Disease
53723	02/05/1973	M	Heart Disease
57321	01/01/1963	F	Diabetes
53724	02/06/1979	M	Heart Disease
56305	03/04/1963	F	Cancer
56309	09/08/1961	M	Flu
57311	05/05/1963	F	Cancer
57322	03/02/1963	F	Cancer
56306	01/02/1966	M	Diabetes

Table 2. A 3-Anonymous version of Table 1.

ZIP code	Date of Birth	Gender	Diagnosis
537*	1970-1979	M	Heart Disease
537*	1970-1979	M	Heart Disease
537*	1970-1979	M	Heart Disease
5630*	1961-1966	Any	Flu
5630*	1961-1966	Any	Cancer
5630*	1961-1966	Any	Diabetes
573*	1963	F	Diabetes
573*	1963	F	Cancer
573*	1963	F	Cancer

Secondly, suppose that, an attacker can conclude that Mahi corresponds to a record in the last equivalence class in Table 2. Furthermore, suppose that he knows that Mahi has very low risk for Diabetes. This background knowledge enables the attacker to conclude that "Mahi has cancer" leading to a privacy breach.

To address these limitations of k-anonymity, l-diversity has been introduced in [14]. An equivalence class is said to have l-diversity if there are at least l "well-represented" values for the sensitive attribute. A table is said to have l-diversity if every equivalence class of the table has l-diversity [14]. The Simplest definition of "well represented" is to ensure there are at least l distinct values for the sensitive attribute in each equivalence class which we adopt in our work. Other types of l-diversity are entropy l-diversity, and recursive (c, l)-diversity [14].

Research has focused on the privacy of relational and static data sets [3, 6, 9], clustering of data streams [22] and anonymization of incremental data sets [2] which assumes that the whole dataset can be kept for processing. These solutions can't be easily adapted for data streams privacy-preserving. In this work we are motived by the unique requirements to anonymize data streams. Data streams are real time; unbounded/infinite (whole dataset can't be kept); fast growing and rapidly changing (multiple scans are not possible); tuples have to be released before a time constraint expires. In addition, many data stream applications are distributed in nature. For example supermarket chains where check-out scanners, located at different stores, gather data continuously. Another example is international companies that have some data located in different continents. Those companies have various reasons why the data cannot be transmitted to a central site, e.g., limited bandwidth or security aspects. However, the few work that addressed anonymizing data stream has implied that the data streams is to be merged at central location for anonymization.

We claim to be the first to propose a distributed model for anonymizing data streams. In this paper our k-anonymity solution for distributed data streams [15] is extended to provide l-diversity as well.

The main contributions of the paper are:

- An l-diversity protocol for anonymizing distributed data stream is presented.
- Complexity analysis and an experimental study that shows the efficiency and effectiveness of the proposed protocol.

The rest of the paper is organized as follows. Section 2 surveys related work. Section 3 is a preliminary that introduces the proposed model and definitions. In section 4 the main algorithm and its procedures are detailed. Complexity analysis is depicted in section 5. Performance evaluation is presented in section 6. Finally, section 7 concludes the paper and discusses further research.

2. Related Work

Several studies have discussed different techniques for privacy-preserving including k-anonymity [17, 19], perturbation [5], condensation [1] and cryptography [7]. k-anonymity, adopted in this work, for static data

is NP-hard and the research has focused on approximate solutions [6, 10]. Moreover, new approaches have been proposed to overcome *k*-anonymity possible inference attacks such as *l*-diversity [14] and *t*-closeness [12].

There are few work that addressed anonymizing (central) data streams and generally are based on two approaches tree-based and cluster-based. The tree-based approach maintains a tree whose nodes are a set of tuples and their common generalization. Sliding Window Anonymization Framework (SWAF) [20], Stream K-anonYmity (SKY) [11], Continuous Privacy Preserving Publishing of Data Streams (BYJ) [25] and K-anonymIzation Data Stream base on sliding window (KIDS) [24] are tree-based. Compared to the cluster-based approach it owns larger time and space complexity and entails larger information loss and never extended to provide *l*-diversity.

The cluster-based approach for providing *k*-anonymity for data streams continuously anonymize data via adaptive clustering. That is grouping close records and releasing it under the same generalization. The following is a list of cluster-based anonymization work: Continuously Anonymizing STreaming data via adaptive cLustEring (CASTLE) [4], B-CASTLE [21], Fast Anonymizing Algorithm for Numerical Streaming DaTa (FAANST) [23], FADS [8], and FAST [16]. Only CASTLE and Fast clustering-based k-Anonymization approach for Data Streams (FADS) extended their work to provide *l*-diversity on top of *k*-anonymity. The work in this paper extends CASTLE to provide *k*-anonymity and *l*-diversity for distributed data streams.

3. Preliminary

The term data refers to a table of rows (or tuples) and columns (or attributes). k-anonymity for data streams is introduced in CASTLE [4], and termed k_s-anonymity. CASTLE is a cluster-based approach that tries to group close tuples into clusters while minimizing an IL metric. The algorithm maintains a buffer that holds a (count-based) window of the data stream sized by a delay constraint (δ). In addition, it maintains two sets of clusters. The first set (Γ) is the working non-anonymized clusters that are updated, merged and split due to insertion and release of tuples. This set has a size limit of β. The second set (Ω) contains anonymized clusters of good quality (in terms of information loss) that are kept to be reused according to a reuse strategy. The IL threshold of good quality clusters is tuned using a parameter μ. A new arriving tuple is inserted into the cluster that requires the minimum enlargement. Due to the data stream characteristics tuples have to be released before a delay constraint expires. Tuples belonging to the same cluster are released with the same cluster QI generalization. After releasing all tuples belonging to a cluster, the cluster is moved to the anonymized set Ω (for reuse) if it has good quality.

Motivated by the distributed nature of many data stream applications, in [15] we extended k_s-*anonymity* to the distributed data streams case, k_{ds}-*anonymity*. In central CASTLE, it is implied that distributed data streams are merged at a central site for anonymization. In our proposed model, Figure 1, we assume the data streams are generated by *n* sites, and each site is to process/anonymize its own data stream. However, the anonymized streams are merged before releasing. Our novel approach is based on exchanging local view of working clusters. Every site *x* sends its own working non-anonymized clusters Γ_x to each other site (through a relay sever). These messages contain only cluster information (such as ID, size, and generalization) but actual tuples are not sent. In addition, message sending is triggered by merge, split, or release of clusters. The goal is for sites to construct approximately the same view of working non-anonymized clusters. In this case, a released cluster of size at least *k* implies that there is more than *k* tuples belonging to the cluster across all *n* sites. The approach is heuristic that approximates the same global view of clusters at different sites to reduce the information loss.

CASTLE [4] extended its main algorithm to provide data stream l-diversity in addition to k_s-anonymity. Similarly, in this paper we extend our distributed data stream k_{ds}-anonymity algorithm in [15] to provide l-diversity in a distributed scenario.

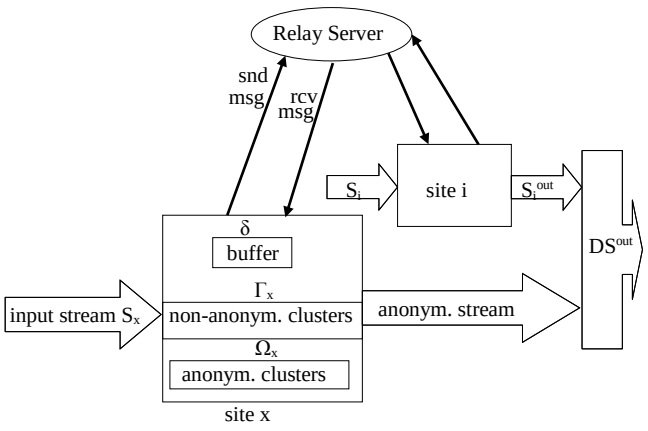


Figure 1. Distributed CASTLE model.

3.1. Network Model

We assume two level hierarchical model where *n* sites connect to a central relay server. The communication between the sites and the server is secured, for example by VPN. In addition, we assume a semi-honest adversary model which follows the protocol but tries to infer valuable information.

3.2. Information Loss Metric (IL)

A Generalized Loss Metric [4] is adopted for

categorical and continuous QI attributes. Let {q_1, …., q_m} be the set of QI attributes.

For a categorical attribute q_i generalized to a node v in the Domain Generalization Hierarchy (DGH_i), the information loss associated with v is defined as follows:

$$VInfoloss(v) = \frac{|S_v| - 1}{|S| - 1} \quad (1)$$

Where S_v is the set of leaf nodes of the subtree rooted at v and S is the set of all the leaf nodes in DGH_i.

For a continuous attribute q_i value generalized to an interval I = [*lower*, *upper*] from the domain [L, U], the information loss associated with I is defined as follows:

$$VInfoloss(I) = \frac{upper - lower}{U - L} \quad (2)$$

Cluster generalization $g = (v_1, \ldots, v_m)$ over a set of tuples is defined such that for each QI attribute q_i, the corresponding range value v_i is defined as follows:

- If q_i is a categorical attribute, v_i is lowest common ancestor (in DGH_i) containing all set of q_i leaves included in the tuples
- If q_i is a continuous attribute, v_i is the minimal subinterval of q_i's domain that contains all q_i's values of the tuples

The total information loss of a tuple QI generalization $g = (v_1, \ldots, v_m)$ is defined as follows:

$$Infoloss(g) = \sum_{i=1}^{m} VInfoloss(v_i) \quad (3)$$

3.3. Cluster Enlargement

Enlargement of cluster C, with respect to a tuple t, is the difference between total IL when applying the new generalization g' resulted from adding t to C and the current generalization g.

3.4. Distance Between Tuples

The distance between two tuples t and t' is calculated as the sum of all corresponding attributes distances such that

- For a categorical attribute, the distance is calculated using *VInfoloss* for the lowest common ancestor of t and t' attribute values.
- For a continuous attribute, the distance is the absolute difference between t and t' attribute values divided by their domain range.
- *Example* 1: To calculate the distance between tuples, assume a tuple QI has two attributes (age; education), where *age* is a continuous attribute in the domain [0-100] and *education* has a DGH illustrated in Figure 2. Consider t_1=(26, Bachelors), t_2 =(29, Doctorate) and t_3 = (28, Masters), Distance (t_1, t_2) = (29-26)/(100-0) + *VInfoloss*(University)=0.03 + ((3-1)/(7-1))=0.363, and Distance (t_1, t_3) = (28-26)/(100-0) + *VInfoloss*(University)= 0.02+((3-1)/(7-1)) = 0.353. Thus, t_3 is closer to t_1 than t_2.

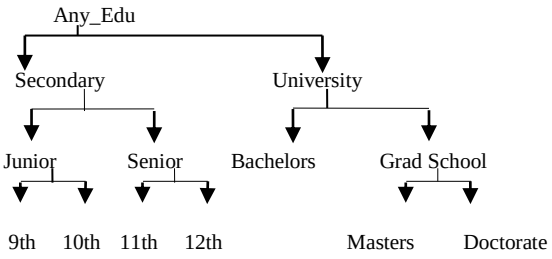


Figure 2. DGH for education Attribute [6].

3.5. K_{ds}-Anonymization of Distributed Data Stream

The input data stream at site x is a sequence of tuples, $DS_x(p_x, pid_x, q_1, q_2, ..., q_m, a_1, a_2, \ldots, a_l)$, where p_x is the tuple position, pid_x is a person ID, (q_1, q_2, ..., q_m) are the QI attributes, and (a_1, a_2, …, a_l) are the remaining attributes including sensitive ones. We assume all explicit identifiers are suppressed and only pid_x is kept.

The output data stream at site x, DS_x^{out}, is an anonymized sequence such that for each tuple $t \in DS_x$ a corresponding anonymized tuple t' in DS_x^{out} is found. In addition, for each QI group (i.e., cluster), the set of included distinct persons (pid_x) are greater than or equal to k. Moreover, a delay constraint (δ) is enforced such that for each new arriving tuple t with position p_x, all tuples with positions less than (p_x - δ) have been in output by DS_x^{out}.

3.6. K_{ds}-Anonymized and l-Diversified Cluster

Consider a single sensitive attribute a_s. Ensuring l-diversity requires that all tuples with the same generalization, i.e., all tuples belonging to the same cluster, have at least l distinct values for a_s. Given a cluster C at site x, let *C.size* denote the number of distinct persons (pid_x) belonging to the cluster. *C.size* is approximated by the sum of the cluster local size (at site x) and the shared size. In addition, let *C.diversity* denote the number of distinct values of a_s for tuples in C. Similarly, *C.diversity* is approximated by the count of distinct a_s values at site x and shared distinct values from one other site.

If, at a given time instant *C.size* is greater than or equal to k, and *C.diversity is* at least l, and all tuples in C are output with C's generalization. Then C is k_{ds}-anonymized and l-diversified cluster.

4. Algorithm and Procedures

4.1. Procedure CentralServer()

The central server waits for all sites to connect before listening to messages, and it is assumed that no site fails. The server relays messages between sites

through a Transmission Control Protocol (TCP) server socket on a predefined port.

4.2. Messages

Figure 3 shows the exchanged message format. A message sent by site *x* contains either a list of newly shared clusters or a cluster update. For each cluster, the message contains:

- Cluster ID which is a concatenation of a site ID and a sequential identifier
- Cluster size that is at least $\lceil k/n \rceil$ for a newly shared cluster, at least *k* for a released cluster, and zero for merged or split clusters
- Cluster QI generalized values
- Array of distinct values of the sensitive attribute a_s in the cluster; for a categorical a_s the array size equals to the number of leaf nodes of the attribute DGH (an entry is set to 1 if the attribute exists otherwise it is set to 0); for a continuous a_s actual distinct values are sent

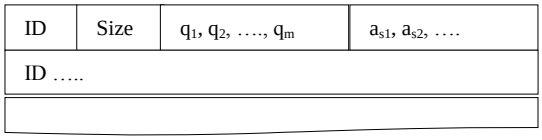


Figure 3. Message format.

A cluster *C* size (*C.size*) is the sum of the cluster local size and its shared size (received in a message). Similarly, a cluster *C* diversity (*C.diversity*) is the count of distinct a_s values in local tuples and in the shared array (received in a message).

Note that a calculated size or diversity at site *x* uses local cluster (size or diversity, respectively) and a shared cluster (size or diversity, respectively) from only one other site. This approximation guarantees that a released cluster will have size > *k* and diversity > *l*. If more sites to be included in the calculation that should provide tighter bounds but increases the messaging and space complexity.

4.3. Main Algorithm: dCASTLE(DS_x, k, l, δ, β, μ, n) @ site x

The software components are depicted in Figure 4.

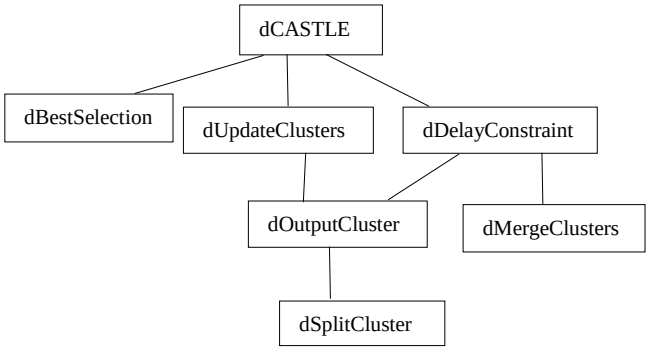


Figure 4. Software components.

The input to the main algorithm distributed CASTLE, dCASTLE, at site *x* is a continuous stream of data DS_x and the following parameters:

- The anonymity parameter *k*.
- The diversity parameter *l*.
- The delay constraint δ.
- A size limit for the working local non-anonymized set of clusters β.
- A parameter used to tune the set size of reused (anonymized) clusters μ.
- Number of sites *n*.

The algorithm produces in output a flow of k_{ds}-anonymized and *l*-diversified tuples.

Every site maintains, as illustrated in Figure 1, a buffer that holds a window of size δ of the data stream, the set of local and shared non-anonymized (i.e., working) clusters (Γ_x) and a subset of good quality anonymized clusters (Ω_x) for reuse. The threshold β is used to limit Γ_x so that no new local cluster will be created if $|\Gamma_x|$ reaches β, however due to exchange of cluster information the maximum size of Γ_x could be larger (β^+) with an upper bound $n\,\beta$.

Using TCP socket, the algorithm spawns a thread to listen to messages from the server. When a message (*msg*) is received, procedure dUpdateClusters(msg) is called.

The algorithm performs the following steps when a new tuple *t* arrives:

- Sends a message *msg* that contains a selected list of recent (not sent before) local non-anonymized clusters from Γ_x with size $\geq \lceil k/n \rceil$ if any
- Calls procedure dBestSelection(t*)* to select the best cluster *C* to insert *t*
- If the procedure returns NULL, creates new cluster to enclose *t* with unique cluster ID and add it to Γ_x else insert *t* in *C*
- Verifies whether the arrival of *t* makes a tuple *t'* with position ($t.p_x$ - δ) that has not been output yet to expire; if yes, calls procedure dDelayConstraint(t')

4.4. Procedure dUpdateClusters(msg)

Procedure *dUpdateClusters(msg)* takes as input a message (*msg*) received from the server and updates Γ_x. The message has one of three types as follows:

- *Type* 1: *msg* contains a list of newly shared clusters (identified by their ID) with size > $\lceil k/n \rceil$ each. Those clusters are added to the set of non-anonymized clusters Γ_x.
- *Type* 2: *msg* contains an update to a cluster with size ≥ *k* and diversity ≥ *l* (in case the cluster has been sent to output). In this case, the cluster is updated and procedure *dOutputCluster(C)* is called.

- *Type* 3: *msg* contains an update to a cluster with size set to zero (in case cluster is merged or split). The cluster shared size is set to zero. However, local tuples are kept in the cluster.

4.5. Procedure dBestSelection(t)

The procedure selects the best cluster to host a new tuple *t*. If it returns NULL, the main algorithm creates a new cluster over *t*. Note that, IL threshold τ is the average information loss of the most recent outputted μ number of clusters, and is calculated in *dOutputCluster*. The procedure is as follows:

Algorithm 1: dBestSelection(t)@ site x

loop through Γ_x and calculate the enlargement due to the insertion of t in each cluster;
select the clusters requiring the minimum enlargement, then select those whose information-loss (IL) $\leq \tau$ and then return the cluster with minimum size;
if no cluster with IL $\leq \tau$ and size $\leq k$ then
  if $|\Gamma_x| < \beta$ then
    return NULL to create a new cluster over t
  else
    return the minimum size cluster among those requiring the minimum enlargement
  end if
end if

4.6. Procedure dDelayConstraint(t)

The main goal of this procedure is to output the tuple *t* hosted in cluster *C* according to the following 5 cases checked in order. Note that, a cluster is considered as an outlier if it is smaller in size than the majority of other clusters.

- *Case* 1: loop through Γ_x and verifies if there is a better cluster *C'* in Γ_x (has less IL) with *C'.size* $\geq k$ and *C'.diversity* $\geq l$; if more than one cluster exists, the minimum IL cluster is chosen to host *t*; the tuple is moved to that cluster *C'* and *dOutputCluster(C')* procedure is called.
- *Case* 2: verifies whether *t* can be output with its hosting cluster *C* with *C.size* $\geq k$ and *C.diversity* $\geq l$; if yes the procedure *dOutputCluster(C)* is called.
- *Case* 3: loop through Ω_x to check the possibility of the reuse strategy, i.e. find clusters in Ω_x that contain *t* and randomly select *C'* and output *t* with *C'* generalization.
- *Case* 4: verifies if *t* is contained in an outlier cluster, then suppress it (i.e., output with the most generalized QI values).
- *Case* 5: verifies whether a merge among the hosting cluster *C* and clusters in Γ_x will result in a cluster with *C.size* $\geq k$ and *C.diversity* $\geq l$; if yes then call *dMergeClusters(C)* followed by *dOutputCluster(C)* else suppress *t*.

4.7. Procedure dMergeClusters(C)

The dMergeClusters(C*)* procedure takes as input the cluster to be merged *C* and proceeds as follows:

Algorithm 2: dMergeClusters(C) @ site x

loop until C.size is at least k and C.diversity is at least l
{
  loop through Γ_x except C
  {
    calculate the enlargement of C if merged with every cluster C' belonging to Γ_x;
    select the cluster C_{min} which brings the minimum enlargement and merges it with C;
    removes C_{min};
    send a message msg to trigger C_{min} removal at other sites (size is set to 0)
  }
}

4.8. Procedure dOutputCluster(C)

Algorithm 3: dOutputCluster(C) @ site x

#initialize a set of clusters SC to be empty
if C.localsize $\geq$ 2 k and C.localdiversity $\geq l$ then*
  SC = dSplitCluster(C)
else SC = C
end if
output all tuples in each C_i in SC
update τ (using μ sized queue)
foreach C_i in SC
{
  if C_i information loss < τ, then
    add C_i to Ω_x
  end if
  delete C_i from Γ_x
}
if(SC == C) then
send a message msg to trigger other sites to output the cluster C
end if

4.9. Procedure dSplitCluster(C)

It is assumed that $k \geq l$. First the procedure creates a set of buckets *BS* according to the tuples values of the sensitive attribute a_s. If $|BS| < l$, this means that the split is not possible and the input cluster *C* is returned. Else generates subclusters C_{sub} by selecting from each bucket $B_j \in$ BS a subset of tuples T_j proportional to the bucket size. The procedure is described as follows:

5. Complexity Analysis

5.1. Space Complexity

Algorithm 4: dSplitCluster(C) @ site x

generate buckets BS by selecting only one tuple from C for each distinct pid_x value, and group only these selected tuples into buckets BS according to their values of the sensitive attribute a_s;
if |BS| < l then
  return C
else continue

end if
initialize a set of clusters SC to be empty;
while $|BS| \geq l$ *and (sum =* $\Sigma_{Bi \in BS} |B_i|) \geq k$
{
random select a bucket from BS and random select a tuple from it and generate C_{sub} *over the tuple and populate it as follows;*
for each B_j *in BS*
{
sort the tuples t_i *in ascending order by their distance* $d_i = (C_{sub}, t_i)$*;*
select the first T_j *(= k * |*B_j*| / sum) tuples and insert it into* C_{sub}*;*
selected tuples are deleted from the bucket; and empty buckets are deleted as well;
}
add C_{sub} *to SC;*
}
loop through each of the remaining tuples t in BS
{
loop through each of created sub-clusters
{
push t into one of the new clusters that requires the minimum enlargement to enclose it;
}
}
insert each tuple t not selected from C, while forming BS, into the unique subcluster that contains a tuple with the same pid_x *of t;*
send a message msg to trigger C removal as a shared cluster at other sites (size is set to 0);
return SC

The space complexity S_{cost} at site x, is the space required to store a window of the data stream, DS_x, plus the space used by the data structure of the non-anonymized (Γ_x) and anonymized (Ω_x) clusters. Let

- D be the number of QI attributes.
- S_t be the space required to store a tuple.
- S_g be the space for a QI attribute generalization.
- S_a be the space required to store a sensitive attribute (for a shared cluster).
- N_c maximum number of clusters in Ω_x.

The data stream window is bounded by δ, the number of non-anonymized clusters is bounded by β^+ (close to β with upper limit $n\beta$), and the number of anonymized clusters is bounded by N_c.

$$S_{\text{cost}} = \delta \times S_t + \beta^+ \times (D \times S_g + k \times S_a) + N_c \times D \times S_g \quad (4)$$

5.2. Time Complexity

Estimating the time complexity requires analyzing the main algorithm and the procedures it performs.

Let t_e be is the time required to calculate the enlargement of one dimension (or the distance between two tuples in one dimension).

Both dDelayConstraint (section 4.6.) and dOutputCluster (section 4.8) do not contain a time consuming calculation such as cluster enlargement or distance between two tuples.

dBestSelection (section 4.5.):the only loop is in step 1 that goes through Γ_x and calculates the enlargement. In the worst case, the cluster is enlarged over all the QI attributes (|QI| = D). Then

$$T_{select} = |\Gamma_x| \times |QI| \times t_e = \beta^+ \times D \times t_e \quad (5)$$

dMergeClusters (section 4.7): there is a nested loops at step 1 and step 2 that go through clusters in Γ_x (excluding C) and calculates the enlargement to reach k and l requirements. In the worst case, each iteration the Γ_x size is decreased by 1.

$$T_{merge} = \sum_{i=1}^{|\Gamma_x|-1} (|\Gamma_x| - i) \times |QI| \times t_e = \frac{\beta^+(\beta^+ - 1)}{2} \times D \times t_e \quad (6)$$

dSplitCluster (section 4.9): assuming $k > l$, in the worst case, the number of tuples in C is δ and each tuple has a distinct pid_x. In step 6 a loop runs through number of buckets $|BS|$ and calculates the enlargement for each tuple (step 7) until k and l requirements are satisfied (step 4). In the worst case every time a new sub-cluster is created, δ is decreased by k. In addition, in steps 11 and 12, the remaining tuples (at most k-1) are pushed to one of created sub-clusters ($\lfloor \delta / k \rfloor$) after calculating the enlargement.

$$T_{split} = \sum_{i=0}^{\delta/k-1} (\delta - ik) \times |QI| \times t_e + (k-1) \times \left\lfloor \frac{\delta}{k} \right\rfloor \times |QI| \times t_e \approx \frac{\delta^2}{2k} \times D \times t_e \quad (7)$$

dCASTLE: Distributed CASTLE iterates on every incoming tuple in the data stream $|DS_x|$, and calls *dBestSelection*, also when an expiring tuple need to be output, *dCASTLE* may call either *dMergeClusters* or *dSplitCluster*. In the worst case, we can say that this happens for every tuple in $|DS_x|$. However, every time a merge is performed, the number of tuples decreases by at least k. Similarly, every time a split is performed, the number of tuples decreases by at least 2k.

$$T_{dCASTLE} = |DS_x| \times T_{select} + \frac{|DS_x|}{k} \times T_{merge} + \frac{|DS_x|}{2k} \times T_{split} = O(|DS_x|) \quad (8)$$

5.3. Messaging Overhead

The number of messages sent is traced at each site x in dCASTLE. For every new tuple, a message is sent by iterating through Γ_x looking for new clusters with size at least k/n, in the worst case, this message is sent for every new tuple. In addition, cluster updates are sent in dOutputCluster (step 9) (in the worst case a message is sent for every k tuples), dMergeClusters (step 6) (in the worst case a message is sent in each iteration of the loop) or dSplitCluster (step 15) (in the worst case a message is sent for every 2k tuples).

$$
\begin{aligned}
nMsg_{new} &= |DS_x| \\
nMsg_{output} &= \frac{|DS_x|}{k} \\
nMsg_{merge} &= \frac{|DS_x|}{k}\ \frac{\beta^+(\beta^+-1)}{2} \\
nMsg_{split} &= \frac{|DS_x|}{2k} \\
nMsgs &= O(|DS_x|)
\end{aligned}
\tag{9}
$$

6. Performance Evaluation

The performance of the proposed algorithm is evaluated using real data set. The "Adult" data set from UC Irvine Machine Learning Repository [13] is used that became a standard for testing anonymity [3, 4, 6, 8, 15, 16, 21].

UCI-Adult contains 30,162 tuples after removing 15,060 tuples having missing values. Quasi-identifier QI attributes are selected from the following 11 attributes: age, final-weight, education-number, capital-gain, capital-loss, hours-per-week, work-class, education, marital-status, occupation, and nation. The first six of them are continuous, and the last five are categorical. The hierarchies for categorical attributes, and the intervals for continuous attributes are adopted from [6].

Intervals for continuous attributes are: age [0-100]; final-weight [0-1500000]; education-number [0-20]; capital-gain [0-100000]; capital-loss [0-5000]; hours-per-week [0-100]. Number of leaves in the DGH of work-class is 8, education is 16, marital-status is 7, occupation is 14, and nation is 40.

First, an experiment is designed to quantify the impact of the different parameters on information loss (section 6.1.). Second, the messaging overhead is compared for both dCASTLE algorithms k-anonymity and l-diversity (section 6.2). Third, both techniques are compared in terms of information loss as well (section 6.3.). Finally, the effectiveness of the proposed l-diversity algorithm is studied by comparing the introduced information loss to a lower and an upper bounds (section 6.4.).

Note that the *k*-anonymity algorithm releases *k*-anonymized clusters, while the *l*-diversity algorithm releases clusters that are *k*-anoymized and *l*-diversified.

6.1. 2^m Factorial Experimental Design

This experiment is designed to study dCASTLE and quantify the effect of different parameters on IL. Positive values indicate positive correlation, and negative values indicate negative correlation.

For *k*-anonymity, there are four parameters (*QI*, *k*, δ, and *n*) and 2^4(=16) experiments with the following low and high value of each variable *QI* = {2, 10}; *k* = {20, 400}; *n* = {2, 10}; δ = {1000, 30000}. . (We omit tables due to space limitation). The experimental results show that when *QI*, *k*, or *n* increases, IL increases. However, when δ increases, IL decreases. *QI* contribution is the highest and equals 58%; *k*= 19%; δ =9%; *n* < 1.5%; for interactions: *QI*-*k*=7%; *QI*- δ =3%; *k*- δ = 2%.

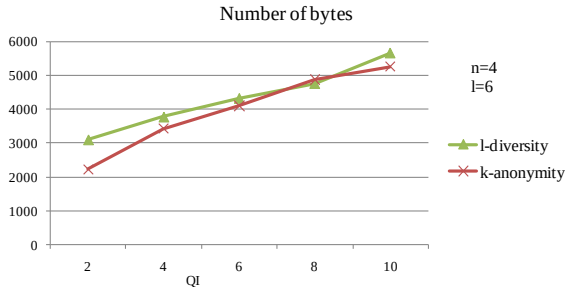
For *l*-diversity, the effect of *l* is studied as well. In this case there are 5 parameters and 2^5 (=32) experiments. *l* high and low values = {2, 10}, and *work-class* (categorical attribute) is the sensitive attribute. Similarly, the contribution of *QI* is the highest and equals to 75%; *k* = 9%; δ =6%; *n* = 1%; *l* ≈ 0. For the interactions: *QI*- δ = 3%; *k*-*QI*=2.5%; *k*- δ = 2%.

6.2. Messaging Overhead

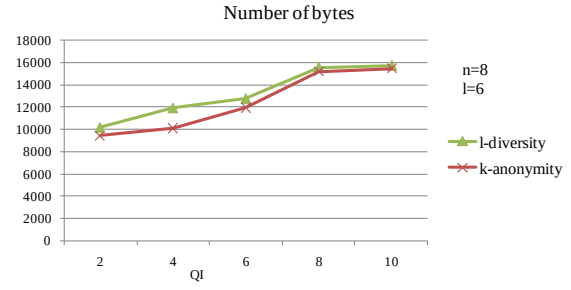
In this experiment, the number of bytes sent by dCASTLE is measured in both the k-anonymity and l-diversity cases, where μ=100 and β=25. For l-diversity, work-class is the sensitive attribute and l=6.

As shown in Figure 5, the number-of-bytes sent increases with *QI* increases (Figure 5(a,b)) as well as with *n* (number-of-sites) increase (Figure 5(g)). However, the number-of-bytes sent decreases with *k* increase (Figure 5(c,d)) and δ increase (Figure 5(e,f)). Moreover, *l*-diversity sends more bytes than *k*-anonymity, and this increase is insignificant most of the time.

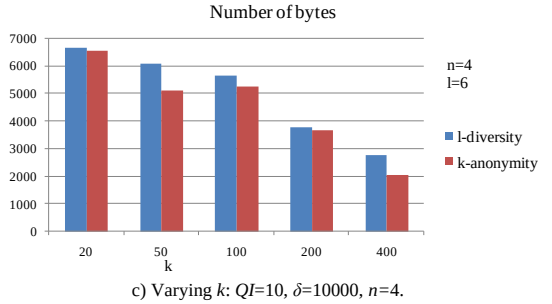
The data set size is 6.6M bytes and the max number-of-bytes sent is this experiment is 20K (Figure 5(d), *n*=10, *l*-diversity). In this case the messaging cost is 0.3% of the data set size.



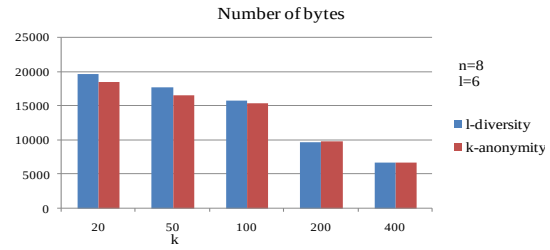
a) Varying *QI*: *k*=100, δ=10000, *n*=4.



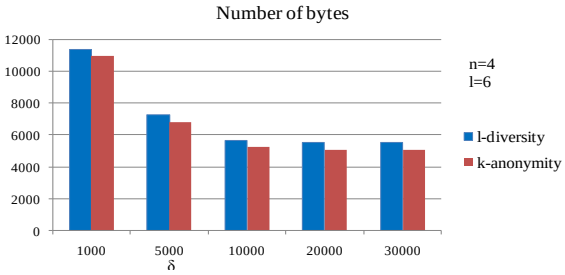
b) Varying *QI*: *k*=100, δ=10000, *n*=8.



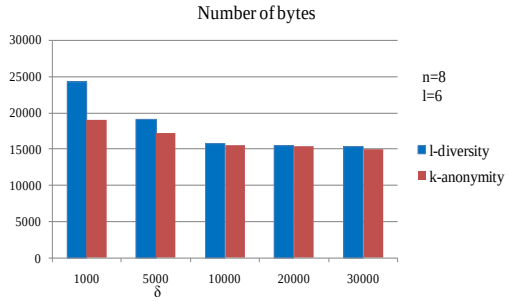
c) Varying *k*: *QI*=10, δ=10000, *n*=4.



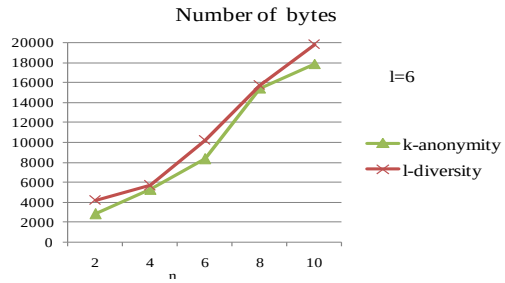
d) Varying k: QI=10, δ=10000, n=8.



e) Varying δ: QI=10, k=100, n=4.



f) Varying δ: QI=10, k=100, n=8.



g) Varying n: QI=10, k=100, δ=10000.

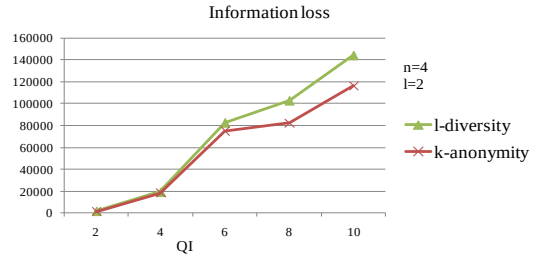
Figure 5. Communication cost.

We recorded the increase of β in *dCASTLE* for the same experiment. The maximum obsereved β^+ is 43 (< 2β).

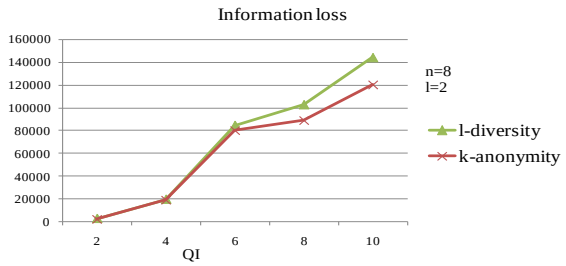
6.3. Comparing *k*-anonymity and *l*-diversity

In this experiment the IL of *dCASTLE* (μ=100, β=25 as tuned in [15]) is compared in the two cases of *k*-anonymity and *l*-diversity. For *l*-diversity, *work-class* is the sensitive attribute, l=2 and we vary values of k, δ, QI and n. Note that, *k*-anonymity resembles l=1 and as l increases the IL is expected to increase as well because *l*-diversity provides more rigorous anonymity.

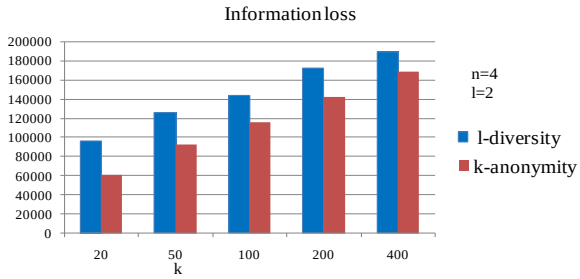
As shown in Figure 6, the IL of *l*-diversity is always more than *k*-anonymity. Similar to the results reported in section 6.1, IL increases with QI Figure 6 (a,b-) and k Figure 6 (c,d) increase. However, IL decreases with δ increase Figure 6 (e,f). In addition, increasing n or l introduces insignificant increase of IL Figure 6 (g,h). The IL increase in *l*-diversity is in the range 20-27%.



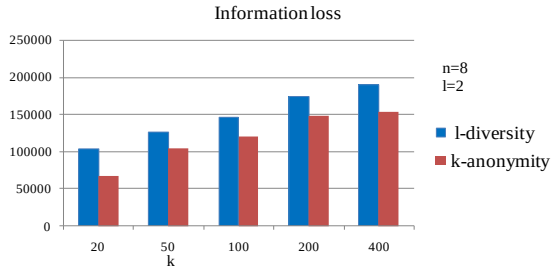
a) Varying QI: k=100, δ=10000, n=4.



b) Varying QI: k=100, δ=10000, n=8.

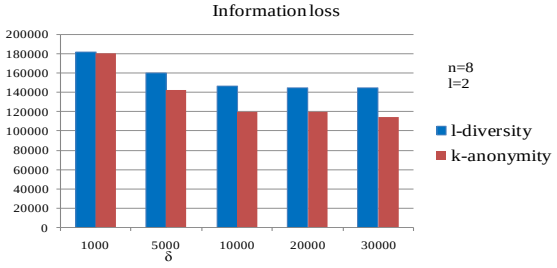


c) Varying k: QI=10, δ=10000, n=4.

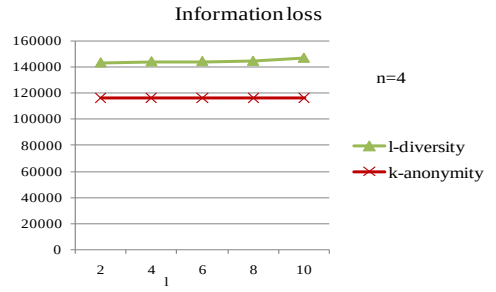


d) Varying k: QI=10, δ=10000, n=8.

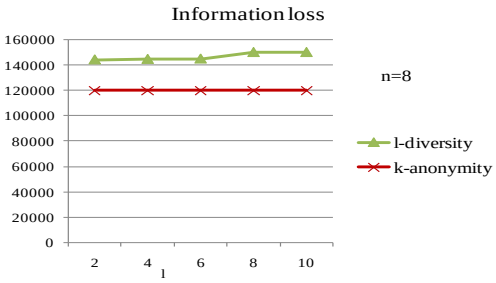
e) Varying δ: QI=10, k=100, n=4.



f) Varying δ: QI=10, k=100, n=8.



g) Varying *l: QI*=10, *k*=100, *δ*=10000, *n*=4



h) Varying *l: QI*=10, *k*=100, *δ*=10000, *n*=8.

Figure 6. Comparing *k*-anonymity and *l*-diversity.

6.4. *l*-diversity Effectiveness

In this experiment the effectiveness of the proposed algorithm is studied by comparing its IL to two other cases. The comparison involves:

1) Centralized CASTLE where all tuples are processed at a central site (Centralized); this case provides IL lower bound.
2) Distributed CASTLE with no communication where tuples are distributed across *n* sites that do not communicate (w/o comm); this case provides IL upper bound.
3) The proposed distributed algorithm where tuples are distributed across *n* sites that communicate a local view of clusters for sites to construct approximately the same global view (Distributed)

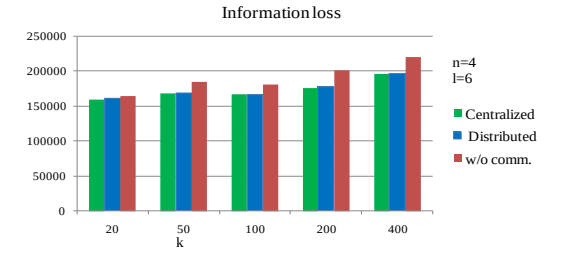
It is assumed that tuples are equally distributed between sites. To resemble the same order of reading from a central site the 1st tuple is assigned to site$_1$, the 2nd tuple is assigned to site$_2$, …,the n^{th} tuple is assigned to site$_n$, the $(n+1)^{th}$ record is assigned to site$_1$, … etc.

The parameters μ and β are tuned in [15] to give the best IL for this data set. For this case μ is set to 100 (for the three approaches), β is set to 25 for distributed case, and β is set to 50 for centralized and w/o comm cases.

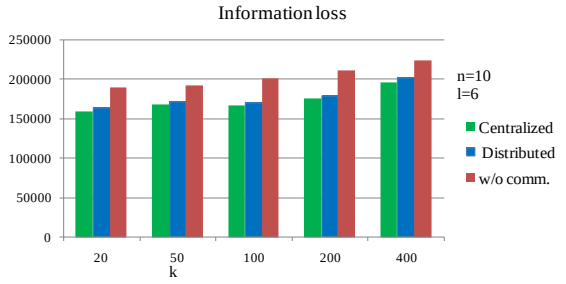
It can be concluded from Figure 7 that the proposed idea of distributed CASTLE provides IL close to the centralized approach most of the time. The IL reduction in the "Distributed" case can reach 100% when compared to the "w/o comm" case. In addition, the "w/o comm" case introduces an increase in IL when compared to the "Distributed" case. This increase can be as large as 35% as reported in this experiment.

a) Varying *QI: k*=100, *δ*=10000, a_s= *work-class*, *l*=2, *n*=8.

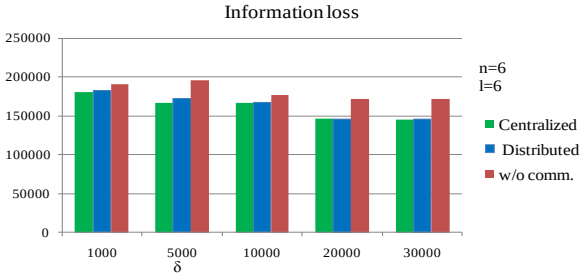
b) Varying *QI: k*=100, *δ*=10000, a_s= *work-class*, *l*=2, *n*=10.



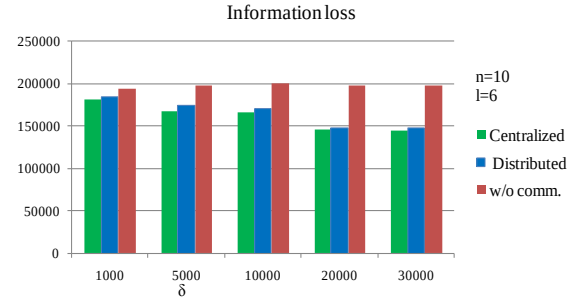
c) *Varying k: QI*=10, *δ*=10000, a_s = *occupation*, *l*=6, *n*=4.



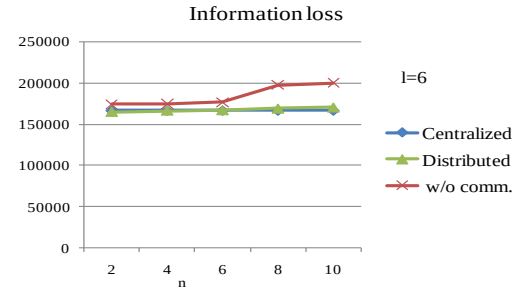
d) Varying *k: QI*=10, *δ*=10000, a_s = *occupation*, *l*=6, *n*=10.



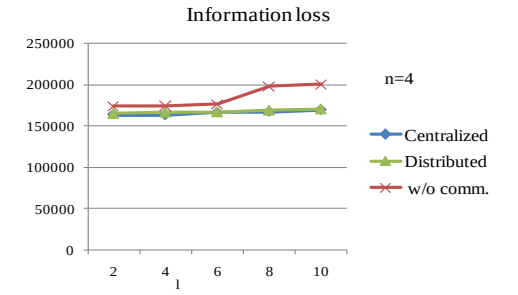
e) Varying *δ: QI*=10, *k*=100, a_s = *occupation*, *l*=6, *n*=6.



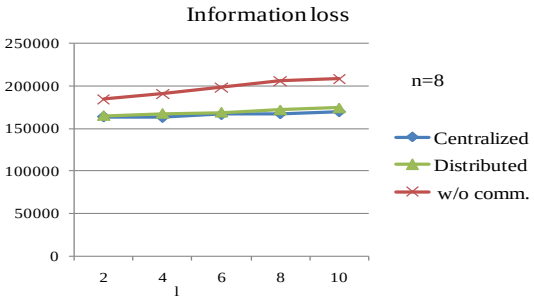
f) Varying *δ: QI*=10, *k*=100, a_s = *occupation*, *l*=6, *n*=10.



g) Varying *n: QI*=10, *k*=100, *δ*=10000, a_s = *occupation*, *l*=6.



h) Varying *l QI*=10, *k*=100, *δ*=10000, a_s =*occupation*, *n*=4.



i) Varying *l QI*=10, *k*=100, *δ*=10000, a_s =*occupation*, *n*=8.

Figure 7. *l*-diversity effectiveness.

7. Conclusions and Future Work

In this paper, a novel distributed model is presented for providing *l*-diversity for applications that generate distributed data streams. Previous research assumes distributed streams are collected at central site for anonymization which is not practical for some real-time applications. Instead, in the proposed model, collecting sites anonymize their streams and communicate their local view to achieve approximately the same global view.

In addition, a distributive *l*-diversity algorithm, dCASTLE, is detailed. The algorithm is cluster-based that groups close tuples while minimizing an information loss metric. The algorithm is heuristic and local views are communicated when triggering events are fired (such as cluster merge, split, or output). The complexity analysis is presented and experiments are performed to study the efficiency and the effectiveness of the proposed algorithm.

First, an experiment on real data set is designed to quantify the effect of different parameters on IL. The main finding is that number of attributes of the QI contributed about 50-75% of IL. The anonymity parameter *k* is the second contributing factor. The number of sites contributed about 1% and this shows the scalability of the algorithm.

Second, *k*-anonymity and *l*-diversity are compared in terms of their messaging cost and information loss. The main finding is that *l*-diversity (*l*=2) introduces about 25% increase in IL when compared to *k*-anonymity. The messaging cost (in bytes) reached 0.3% of the size of the data set.

Third, the proposed distributed approach is compared in terms of IL to the centralized model (as a lower bound) and to a distributed model with no communication (as an upper bound). The main finding is that the distributed approach introduces IL close to the centralized approach most of the time. In addition, about 35% reduction in IL is achieved when compared to the w/o comm case.

For future work, we plan to study providing *t*-closeness as well and experiment with larger data set and larger number of sites. In addition, the data utility will be investigated.

References

[1] Aggarwal C. and Yu P., "A Condensation Approach to Privacy Preserving Data Mining," *in Proceedings of International Conference on Extending Database Technology*, Heraklion, pp. 183-199, 2004.

[2] Aldeen Y., Salleh M., and Aljeroudi Y., "An Innovative Privacy Preserving Technique for Incremental Datasets on Cloud Computing," *Journal of Biomedical Informatics*, vol. 62, pp. 107-116, 2016.

[3] Bayardo R. and Agrawal R., "Data Privacy through Optimal k-Anonymization," *in Proceedings of 21st International Conference on Data Engineering*, Tokoyo, pp. 217-228, 2005.

[4] Cao J., Carminati B., Ferrari E., and Tan K., "CASTLE: Continuously Anonymizing Data Streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 3, pp. 337-352, 2011.

[5] Evfimievski A., Srikant R., Agrawal R., and Gehrke J., "Privacy Preserving Mining of Association Rules," *Information Systems*, vol. 29, no. 4, pp. 343-364, 2004.

[6] Fung B., Wang K., and Yu P., "Top-Down Specialization for Information and Privacy Preservation," *in Proceedings of International Conference on Data Engineering*, Tokoyo, pp. 205-216, 2005.

[7] Goryczka S., Xiong L., and Sunderam V., "Secure Multiparty Aggregation with Differential Privacy: A Comparative Study," *in Proceedings of the Joint EDBT/ICDT Workshops*, Genoa, pp. 155-163, 2013.

[8] Guo K. and Zhang Q., "Fast Clustering-Based Anonymization Approaches with Time Constraints for Data Streams," *Knowledge-Based Systems*, vol. 46, pp. 95-108, 2013.

[9] LeFevre K., DeWitt D., and Ramakrishnan R., "Incognito: Efficient Full-Domain k-Anonymity," *in Proceedings of ACM SIGMOD International Conference on Management of Data*, Baltimore, pp. 49-60, 2005.

[10] LeFevre K. DeWitt D., and Ramakrishnan R., "Mondrian Multidimensional Kanonymity," *in Proceedings of IEEE 22nd International Conference on Data Engineering*, Atlanta, pp. 25, 2006.

[11] Li J., Ooi B., and Wang W., "Anonymizing Streaming Data for Privacy Protection," *in Proceedings of 24th International Conference on Data Engineering*, Cancun, pp. 1367-1369, 2008.

[12] Li N., Li T., and Venkatasubramanian S., "t-Closeness: Privacy Beyond K-Anonymity and L-Diversity," *in Proceedings of 23st IEEE International Conference on Data Engineering*, Istanbul, pp. 106-115, 2007.

[13] Lichman M., UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, Last Visited, 2013.

[14] Machanavajjhala A., Kifer D., Gehrke J., and Venkitasubramaniam M., "l-diversity: Privacy Beyond K-Anonymity," *in Proceedings of 22nd International Conference on Data Engineering*, Atlanta, pp. 24-35, 2006.

[15] Mohamed M., Nagi M., and Ghanem S., "A Clustering Approach for Anonymizing Distributed Data Streams," *in Proceedings of 11th IEEE International Conference on Computer Engineering and Systems*, Cairo, pp. 9-16, 2016.

[16] Mohammadian E., Noferesti M., and Jalili R., "FAST: Fast Anonymization of Big Data Streams," *in Proceedings of International Conference on Big Data Science and Computing*, Beijing, pp. 23-30, 2014.

[17] Sweeney L., "Achieving k-Anonymity Privacy Protection Using Generalization and Suppression," *International Journal of Uncertainty, Fuzziness, Knowledge-Based Systems*, vol. 10, no. 5, pp. 571-588, 2002.

[18] Sweeney L., "K-anonymity: A Model for Protecting Privacy," *International Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, vol. 10, no. 5, pp. 557-570, 2002.

[19] Victor N., Lopez D., and Abawajy J., "Privacy Models For Big Data: A Survey," *International Journal on Big Data Intelligence*, vol. 3, no. 1, pp. 61-75, 2016.

[20] Wang W., Li J., Ai C., and Li Y., "Privacy Protection on Sliding Window of Data Streams," *in Proceedings of International Conference on Collaborative Computing: Networking, Applications and Worksharing*, New York, pp. 213-221, 2007.

[21] Wang P., Lu J., Zhao L., and Yang J., "B-CASTLE: An Efficient Publishing Algorithm for K-Anonymizing Data Streams," *in Proceedings of 2nd WRI Global Congress on Intelligent Systems*, Wuhan, pp. 132-136, 2010.

[22] Yarlagadda A., Jonnalagedda M., and Munaga K., "Clustering Based on Correlation Fractal Dimension Over an Evolving Data Stream," *The International Arab Journal of Information Technology*, vol. 15, no. 1, pp. 1-9, 2018.

[23] Zakerzadeh H. and Osborn S., "FAANST: Fast Anonymizing Algorithm for Numerical Streaming Data," *in Proceedings of Data Privacy Management and Autonomous Spontaneous Security*, Athens, pp. 36-50, 2011.

[24] Zhang J., Yang J., Zhang J., and Yuan Y., "KIDS: K-Anonymization Data Stream Base on Sliding Window," *in Proceedings of 2nd International Conference on Future Computer and Communication*, Wuha, pp. 311-316, 2010.

[25] Zhou B., Han Y., Pei J., Jiang B., Tao Y., and Jia Y., "Continuous Privacy Preserving Publishing Of Data Streams," *in Proceedings of 12th International Conference on Extending Database Technology: Advances in Database Technology*, Saint Petersburg, pp. 648-659, 2009.

Mona Mohamed received her BSc and MSc in Computer Science from Faculty of Engineering in Alexandria University, Egypt, in 2005 and 2011, respectively. She is currently a PhD candidate at the Faculty of Engineering in Alexandria, Egypt. Her main research interests are privacy, data mining and distributed systems.

Sahar Ghanem received her BSc and MSc in Computer Science from Faculty of Engineering in Alexandria University, Egypt, in 1994 and 1997, respectively, and PhD in Computer Science from Old Dominion University in VA, USA, in 2004. She is currently an Associate Professor at the Faculty of Engineering in Alexandria, Egypt. Her main research interests are computer networks, network security, performance evaluation and data mining. She has published about 20 scientific publications in international journals and conference proceedings.

Magdy Nagi received his BSc from the Faculty of Engineering Alexandria University Egypt in 1963. He received his PhD from Karlsruhe University, Karlsruhe, Germany in 1975. He is currently a Professor at the Faculty of Engineering Alexandria University. His main research interests are software engineering, DBMS, data mining and grid computing. He has published more than 35 scientific publications in international journals and conference proceedings.