Journal of Computer and System Sciences 79 (2013) 542–555

Contents lists available at SciVerse ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

An efficient quasi-identifier index based approach for privacy preservation over incremental data sets on cloud

Xuyun Zhang [a,*], Chang Liu [a], Surya Nepal [b], Jinjun Chen [a]

[a] Faculty of Engineering and Information Technology, University of Technology Sydney, PO Box 123, Broadway, NSW 2007, Australia
[b] Centre for Information & Communication Technologies, Commonwealth Scientific and Industrial Research Organisation, Cnr Vimiera and Pembroke Rodas Marsfield, NSW 2122, Australia

ARTICLE INFO

Article history:
Received 21 March 2012
Received in revised form 18 September 2012
Accepted 8 November 2012
Available online 13 December 2012

Keywords:
Cloud computing
Privacy preservation
Incremental data set
Anonymization
Quasi-identifier index

ABSTRACT

Cloud computing provides massive computation power and storage capacity which enable users to deploy applications without infrastructure investment. Many privacy-sensitive applications like health services are built on cloud for economic benefits and operational convenience. Usually, data sets in these applications are anonymized to ensure data owners' privacy, but the privacy requirements can be potentially violated when new data join over time. Most existing approaches address this problem via re-anonymizing all data sets from scratch after update or via anonymizing the new data incrementally according to the already anonymized data sets. However, privacy preservation over incremental data sets is still challenging in the context of cloud because most data sets are of huge volume and distributed across multiple storage nodes. Existing approaches suffer from poor scalability and inefficiency because they are centralized and access all data frequently when update occurs. In this paper, we propose an efficient quasi-identifier index based approach to ensure privacy preservation and achieve high data utility over incremental and distributed data sets on cloud. Quasi-identifiers, which represent the groups of anonymized data, are indexed for efficiency. An algorithm is designed to fulfil our approach accordingly. Evaluation results demonstrate that with our approach, the efficiency of privacy preservation on large-volume incremental data sets can be improved significantly over existing approaches.

1. Introduction

Technically, cloud computing can be regarded as an ingenious combination of a series of developed or developing ideas and technologies, establishing a pay-as-you-go business model by offering IT services using economies of scale [1–3]. Participants in cloud computing business chains can benefit from this novel business model, as they can save huge IT capital investment by facilitating cloud services such as high storage and computation capabilities, and consequently can concentrate on their own core business [4]. Cloud computing also provides attractive features for science applications in academia [5]. Moreover, since cloud is a multi-tenant environment, it is convenient for cloud users to share data and collaborate with each other. Therefore, many companies or organizations have built up IT systems for their business in cloud computing environments.

However, numerous potential cloud customers are still hesitant to take advantage of cloud computing due to security and privacy concerns [6–10]. Privacy protection is one of the concerned issues in this regard. Currently, several large companies

* Corresponding author.
E-mail addresses: xyzhanggz@gmail.com (X. Zhang), changliu.it@gmail.com (C. Liu), Surya.Nepal@csiro.au (S. Nepal), jinjun.chen@gmail.com (J. Chen).

0022-0000/$ – see front matter
http://dx.doi.org/10.1016/j.jcss.2012.11.008

and hospitals have deployed their health services into cloud, e.g., Microsoft HealthVault [11]. The data sets retained in these cloud applications are highly privacy-sensitive. Once an adversary collects these data sets and menaces the privacy-sensitive information, considerable economic loss or severe social reputation impairment will be caused to corresponding individuals. Usually, these data in cloud will be shared and utilized by multiple users for value-added advantages, rather than just for data storage. Encrypting all the data sets [12–15] is a straightforward and effective privacy protection approach. However, processing effectively and efficiently on encrypted data sets on cloud can be quite a challenging task, because most existing applications run on unencrypted data sets. Recent progress has been made in homomorphic encryption research. Theoretically, computation can be performed on encrypted data sets without decrypting them, but the current techniques are rather expensive and impractical with respect to its efficiency [16,17]. Worse still, encryption still fails to protect individual privacy-sensitive information to legal data users even though it can ensure confidentiality against adversaries. As such, data anonymization techniques like generalization [18] and anatomization [19] have been proposed to preserve privacy when privacy-sensitive data are stored in cloud. Data sets are anonymized to satisfy certain privacy requirements such as k-anonymity [20], or l-diversity [21] before they are shared with data users.

The explosive growth of data sets in cloud applications poses a challenge to existing approaches of privacy preservation over incremental data sets. At present, data sets in many cloud applications often grow incrementally over time as new data are collected and added [22,23]. For instance, a cloud health service will update a huge number of personal health records continuously. Since a large amount of data are generated by special devices or cloud users over time, cloud applications are required to handle incremental data efficiently. To cope with updates of large-volume data sets, a variety of incremental or continuous MapReduce variations have been proposed recently [22–25]. MapReduce is a powerful parallel data processing framework on cloud, offering simple programming model for users [26]. Nevertheless, little attention is paid to privacy issues incurred by incremental privacy-sensitive data in such large-volume data scenarios. When an already anonymized data set is added with new data, two possible effects can take place over the whole data set. One is the violation of privacy requirements, i.e., the newly added data fail to satisfy the anonymity requirement even though they are anonymized according to the current anonymity level. The other is over-anonymization, i.e., the newly added data can be exploited to decrease data distortion by specializing the entire data set to a lower anonymity level, which still satisfies the privacy requirement. Therefore, we should adapt anonymized data sets to achieve both privacy requirement compliance and high data utility when data updates occur. Most existing approaches to address this problem are to re-anonymize the whole updated data set from scratch [27–29], thereby suffering from inefficiency and vulnerability to privacy attacks [30]. Thus, incremental approaches have been proposed accordingly [30–32]. Most of these incremental approaches are centralized while data sets are usually distributed in the context of cloud. A distributed approach has been represented in [32] to preserve k-anonymity over distributed data. However, this approach is inefficient with respect to large-volume data sets on cloud, because it accesses all data when performing data updates. Above all, it is still a challenge to efficiently achieve privacy preservation over distributed and incremental data in the presence of data updates.

In this paper, we propose a novel approach to efficiently achieve privacy preservation over distributed and incremental data sets on cloud. To efficiently update anonymized data sets in the presence of new data, indexing structure of quasi-identifiers is established on anonymized data sets. Quasi-identifiers, which represent the groups of anonymized data, are indexed for efficiency. Moreover, similar data records are placed on the same nodes to reduce communication cost across data storage nodes when anonymized data sets are generalized or specialized to achieve anonymity requirements and high data utility. A commonly used privacy model k-anonymity [20] is employed to measure privacy in our research, i.e., privacy requirements are signified by a threshold k. Further, sub-tree generalization scheme [33] is utilized to accomplish data anonymization. An algorithm is designed to fulfil our approach accordingly. Experimental evaluation on real-world data sets demonstrates that with our approach, the efficiency of privacy preservation over incremental data sets can be improved significantly over existing approaches.

The remainder of this paper is organized as follows. The related work is reviewed in the next section. A motivating example and the problem analysis are given in Section 3. In Section 4 we present our quasi-identifier index based approach and its corresponding algorithm in details. The proposed approach is evaluated in Section 5 by conducting experiments on real-world data sets in our cloud environment. Finally, we conclude this paper and discuss our future work in Section 6.

2. Related work

We briefly present a review on recent research about incremental data processing on cloud, privacy-preserving techniques and privacy preservation over incremental data sets.

Plenty of recent research has investigated the issues of processing incremental data on cloud. Recently, MapReduce has been widely revised from a batch processing framework into a more incremental one to analyze huge-volume incremental data on cloud [34]. Kienzler et al. [24] designed a “stream-as-you-go” approach to access and process on incremental data for data-intensive cloud applications via a steam-based data management architecture. Bhatotia et al. [22] extended the traditional Hadoop framework [35] to a novel one named as Incoop by incorporating several techniques like task partition and memorization-aware schedule. Olston et al. [25] presented a continuous workflow system called Nova on top of Pig/Hadoop through stateful incremental data processing. Li et al. [23] proposed a Hadoop-based platform to support incremental one-pass data analytics by employing hash techniques and a frequent key based technique. However, little attention is paid to

the privacy issues caused by incrementally processing huge-volume data. Our research complementarily strives to mitigate such privacy concerns.

As to privacy models and preserving techniques, privacy-preserving data publishing research community has extensively investigated on the issues and made fruitful progress with a variety of privacy models, privacy-preserving methods and algorithms [36]. Privacy principles, such as k-anonymity [20], l-diversity [21] and t-closeness [37] are put forth to model and quantify privacy. Yet most of them are only applied to single data set privacy preservation. For incremental data sets, Byun et al. [38] firstly proposed an anonymization technique for preserving privacy of continuous data publishing by delaying data record publication. Xiao and Tao [27] proposed a privacy notion called m-invariance and developed a corresponding anonymization method by minimally adding counterfeit records. Fung et al. [28] presented a BCF-anonymity privacy model and corresponding anonymization method without delaying data publication or inserting counterfeit data. He et al. [29] proposed a graph based anonymization algorithm to prevent equivalence attacks on incremental anonymized data by exploiting the classic results of the "min-cut/max-flow" problem.

A variety of approaches have been proposed to preserve privacy over incremental data sets. From the perspective of efficiency, Truta and Campan [39] proposed an approach to maintain k-anonymity for incremental data sets by modelling it as a clustering problem. The approach takes less running time than an intuitive method which anonymizes the entire updated data sets from scratch. Pei et al. [30] formally analyzed the effects of incremental updates on k-anonymity requirements and proposed an incremental anonymization algorithm based on monotonic incremental anonymization property. Zhou et al. [31] developed an approach to handle continuous data anonymization by exploiting the statistical distribution of incremental data. The above approaches employ cell generalization scheme [40] or multidimensional generalization scheme [41] to anonymize data sets. However, both schemes suffer from data exploration problem [36]. Dissimilarly, we adopt sub-tree generalization scheme [33] in our research to enable anonymized data sets to be directly processed and analyzed by existing tools. Most of existing approaches to handle updates of anonymized data sets are centralized, i.e., they fail to scale to distributed scenarios like cloud where data sets are in huge volume and are distributed. Doka et al. [32] proposed a distributed system named KANIS to dynamically preserve the privacy of anonymized data set with full-domain generalization scheme [18] in the presence of data updates. This distributed approach is the closest to our research. Nevertheless, it is inefficient for this approach to check the condition whether specialization or generalization is required, because all the data records will be accessed frequently.

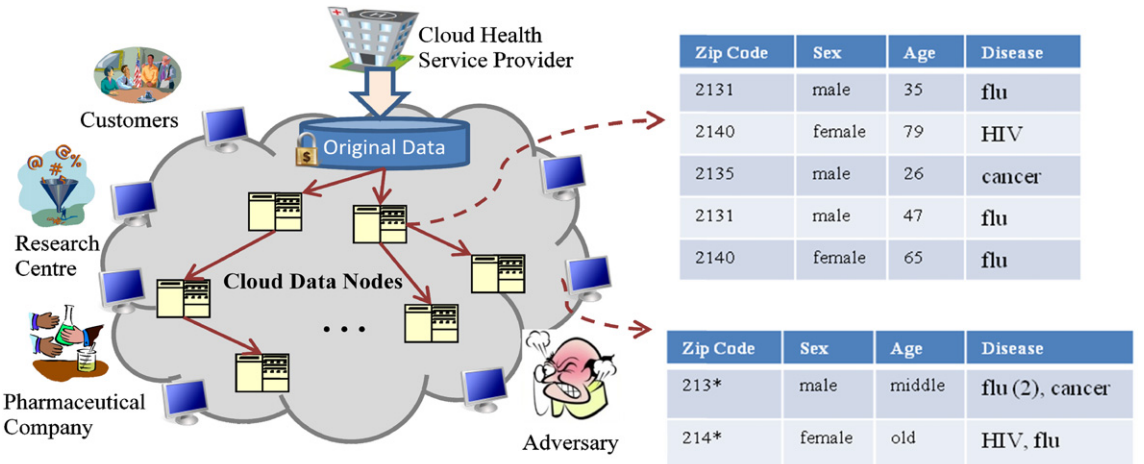
3. Motivating example and problem analysis

In this section, we first demonstrate a motivating example in Section 3.1 to drive our research. We then analyze the privacy-preserving problem of updating incremental data sets in cloud in Section 3.2.

3.1. Motivating example

In a cloud health service application, a health service provider offers health services to cloud customers. With these services, customers can manage their and their families' health. Specifically, customers upload their personal health records into the services and can get helpful information like symptoms analysis, medical diagnosis and health plans. Besides, several hospitals also upload their patients' health information into these services. So, the health service provider can collect huge volume of data and store these privacy-sensitive data sets on cloud to utilize cloud facilities to process these massive data. The data are also analyzed or shared by other institutes like hospitals, pharmacies and even governments. All the storage and computation over these data take place in cloud. Based on the analysis results, customers or patients can be provided with more effective services. However, the data are usually anonymized to preserve the privacy of customers or patients against adversaries and malicious cloud users who may improperly exploit such privacy-sensitive information. Fig. 1 illustrates the scenario of preserving individuals' privacy over incremental data sets in the motivating example.

In the presence of data updates, the privacy requirement of anonymized data sets can be breached or the anonymized data sets can expose more data utility without compromising privacy. As demonstrated in Fig. 1, five data records in an original data set on cloud data nodes are generalized to satisfy 2-anonymity, assuming *Disease* to be the sensitive attribute. The generalized data set has two groups of data records. Suppose a new data record $\langle 2147, female, 45, cancer\rangle$ arrives. According to the current level of generalization, the record will be generalized to $\langle 214*, female, middle, cancer\rangle$. At this point, the 2-anonymity of the entire data set is violated, since the quasi-identifier $\langle 214*, female, middle\rangle$ fails to be merged into any group whose size is not less than two. In this case, the current level of generalization should be adjusted. If the attribute *Sex* is generalized to a higher level, i.e., *any_sex*, the entire data set will satisfy 2-anonymity. On the contrary, if a record $\langle 2135, male, 37, HIV\rangle$ arrives, it can be added to the group with quasi-identifier $\langle 213*, male, middle\rangle$. Then, this group can be further divided into two groups without compromising 2-anonymity, i.e., $\langle 2131, male, middle\rangle$ and $\langle 2135, male, middle\rangle$, respectively. In this case, the over-generalized data set should be specialized to expose more data utility to data users. Furthermore, when data updates take place, the new data should be anonymized as soon as possible to provide information to data users in time. So, we investigate the problem of efficiently updating data sets to comply with anonymity requirements and achieve high data utility in this paper.



Zip Code	Sex	Age	Disease
2131	male	35	flu
2140	female	79	HIV
2135	male	26	cancer
2131	male	47	flu
2140	female	65	flu

Zip Code	Sex	Age	Disease
213*	male	middle	flu (2), cancer
214*	female	old	HIV, flu

Fig. 1. Scenario of privacy preservation over incremental data sets on cloud.

3.2. Problem analysis

3.2.1. Basic denotations

A data provider uploads a privacy-sensitive data set D into a cloud and allows data users to make use of the data. The data set D is usually anonymized in advance to preserve its privacy by data providers before data users access the data. Let D^* denote the anonymized data set of D. In cloud, both D and D^* are stored on data storage nodes. Suppose there are n data nodes for storing D and D^*. These nodes are represented as $DNodes = \{DN_1, DN_2, \ldots, DN_n\}$. Correspondingly, $D = \{D_1, D_2, \ldots, D_n\}$ and $D^* = \{D_1^*, D_2^*, \ldots, D_n^*\}$, where D_i^* is the anonymized data set of D_i, and both data sets are stored on DN_i, $i \in [1, n]$.

A data set D consists of a large number of data records. Usually, explicit identifying information such as name and social security number in data records has been eliminated for protecting individual privacy. The form of a data record can be denoted as $r = \langle v_1, v_2, \ldots, v_m, sv \rangle$, where v_i is an attribute value of an attribute $Attr_i$, and sv is a sensitive value like a disease. The set of sensitive values is denoted as SA. The taxonomy tree of the attribute $Attr_i$, denoted as TT_i, is assumed to be defined for anonymization by domain experts in advance. Both categorical attributes and continuous attributes can have a taxonomy tree [33]. The leaf nodes of TT_i are the original values of $Attr_i$ in a data set D. Let DOM_i be the set of all the domain values in TT_i. The possible value of $Attr_i$ in an anonymized record r^* are taken from DOM_i. Let $qid = \langle q_1, q_2, \ldots, q_m \rangle$ denote a quasi-identifier of an anonymized record r^*, where $q_i \in DOM_i$, $i \in [1, m]$. The set of qids can be then denoted as $QID = \langle Attr_1, Attr_2, \ldots, Attr_m \rangle$. We use QI as the acronym of quasi-identifier.

We employ k-anonymity as the privacy model in our research. Once certain quasi-identifiers are too specific that only a small group of people are linked to them, then these individuals are linked to sensitive information with high confidence, resulting in privacy breach [36]. To combat such privacy breach, the number of the anonymized records that correspond to a quasi-identifier is required to be larger than a threshold. All anonymized records with a quasi-identifier qid constitute a QI-group [42], denoted as $QJG(qid)$. Without loss of generality, we adopt k-anonymity herein as the privacy model. That is, for any $qid \in QID$, the size of $G(qid)$ must be zero or at least k [20]. Assume anonymized data D^* satisfies k-anonymity when a data provider initially stores them on cloud. When $D^* = \{D_1^*, D_2^*, \ldots, D_n^*\}$ is stored on several data nodes, all the data sets D_i^*, $i \in [1, n]$, satisfy k-anonymity as a whole. Note that single data set D_i^* is unnecessary to meet the privacy requirement strictly, since D^* can still be k-anonymous in a global view even though D_i^* violates k-anonymity from a local perspective.

Further, the sub-tree generation scheme is employed to anonymize data records herein. Generalization refers to replacing a group of values with a parent domain value in a taxonomy tree. Specifically, generalization can be mathematically represented as $gen : Child(p) \to p$, where $p \in DOM$, $Child(p) \subseteq DOM$ and $Child(p)$ is the set of child values of p. The reverse process of generalization is specialization which substitutes a domain value with all its child values. Specialization can be formally represented as $spec : p \to Child(p)$, where $p \in DOM$ and $Child(p) \subseteq DOM$. Sub-tree generalization scheme is exploited to generalize or specialize data sets in our research. In sub-tree generalization scheme, all child values of a non-leaf node or none in a domain hierarchy are generalized to the non-leaf node. Consequently, the generalized domain values of an attribute will form a cut through the taxonomy tree. A cut of a tree is a subset of the nodes of the tree that contains exactly one node on each root-to-leaf path. The scheme can make a good tradeoff between data utility and privacy loss, and its anonymized data sets can be directly applicable to most existing data processing or mining tools [33]. Other schemes either suffer from the data exploration problem or expose less data utility than the sub-tree generalization scheme [36].

Let GL denote the generalization level which describes the extent of anonymity. Formally, $GL = \langle Cut_1, Cut_2, \ldots, Cut_m \rangle$, where Cut_i, $i \in [1, m]$, is the current cut of the taxonomy tree TT_i. A cut of a taxonomy tree TT_i, $i \in [1, m]$, is a subset of

values in DOM_i that contains exactly one value in each root-to-leaf path. Intuitively, generalization level of a generalized data set determines the extent of generation over the original data set. In a taxonomy tree TT_i, if certain domain values in a cut Cut_{i_j} are ancestors of certain domain values in another cut Cut_{i_l} while the rest domain values are the same, we denote this relationship between them by $<_C$, i.e., $Cut_{i_l} <_C Cut_{i_j}$. If the domain values in Cut_{i_j} are identical to that in Cut_{i_j}, the relationship can be denoted as $Cut_{i_l} =_C Cut_{i_j}$. If all the cuts in $GL_i = \langle Cut_{i_1}, Cut_{i_2}, \ldots, Cut_{i_m}\rangle$ and $GL_j = \langle Cut_{j_1}, Cut_{j_2}, \ldots, Cut_{j_m}\rangle$ satisfy $Cut_{i_l} \leqslant Cut_{j_l}$, $l \in [1, m]$, we denote this relationship between GL_i and GL_j by $\leqslant_{GL}$, i.e., $GL_i \leqslant_{GL} GL_j$, and the equality holds if and only if all the equations $Cut_{i_l} =_C Cut_{j_l}$ hold. The inequality $GL_i <_{GL} GL_j$ means the generalization level GL_j is higher than GL_i, and GL_j can be achieved by generalizing GL_i, and vice versa.

3.2.2. Problem analysis

The problem is formally described as follows. When new data are added to D, D^* should be updated correspondingly. Let Δ represent the new data. Then the original data is $(D + \Delta)$ after the data update. Correspondingly, the generalized data should be $(D + \Delta)^*$ if $(D + \Delta)$ is generalized from scratch. Let Δ_i denote the part of new data added to the data set D_i. According to the current generalization level, Δ_i will be generalized into Δ_i^*. At this point, the generalized data set on DN_i is $(D_i^* + \Delta_i^*)$ which may NOT be the same as $(D_i + \Delta_i)^*$, i.e., $(D_i^* + \Delta_i^*) \neq (D_i + \Delta_i)^*$. The influence of the new data on generalized data set $(D_i^* + \Delta_i^*)$ is twofold. One impact is that the current k-anonymous state of D^* may be breached, as the size of QI-group $QIG(qid)$ may be less than k, where qid is the quasi-identifier of the new data records generalized according to the current generalization level. In this case, the current generalization level should be adjusted to a higher level to ensure k-anonymity. The other impact is that $(D_i^* + \Delta_i^*)$ may get over-generalized, i.e., it is unnecessary to generalize $(D_i + \Delta_i)$ to the current generalization level because a lower generalization level for $(D_i + \Delta_i)$ can still satisfy the k-anonymity requirement. Since making an anonymized data set as useful as possible for data analysis is another goal when we preserve privacy, it is a necessity to specialize an over-generalized data set into a lower generalization level to expose more data utility to data users. In both cases, we need to dynamically adjust the current generalization level GL_{cur}. That is, a new generalization level GL_{new} should be determined to substitute GL_{cur}. The new generalization level GL_{new} is required to meet k-anonymity and expose high data utility.

Existing approaches fall short of solving the incremental data update problem in an efficient and scalable fashion. Recently, three categories of approach have been proposed to address the privacy-preserving problem of incremental data sets. The first category is to anonymize the entire updated data sets from scratch repeatedly, i.e., to generalize $(D + \Delta)$ to $(D + \Delta)^*$ once new data are added. This category of approach is straightforward yet inefficient and vulnerable to privacy attacks [30]. Given huge volume of data sets in the context of cloud, re-generalizing updated data sets from scratch is inefficient and costly. The second category is to incrementally adapt generalized data sets over time to satisfy a privacy requirement when new data are added. However, the existing approaches on this category only focus on the multidimensional generalization scheme [30,31] or cell generalization scheme [39], and fail to work with respect to the sub-tree generalization scheme. Furthermore, the above approaches are centralized and are incapable of scaling to distributed data sets such as those in cloud. The last category is incremental and distributed for anonymizing full-domain generalization scheme [32]. This distributed approach is the closest to our research. Nevertheless, it is inefficient for this approach to check the condition whether specialization or generalization is required, because this requires all data records to be accessed frequently. Specifically, for each candidate generalization level GL_{cand}, the data nodes DNs have to calculate certain statistical information of any $qid \in QID$ to check whether a candidate GL_{cand} satisfies k-anonymity. All original records will be accessed many times during this period and the update phase. This leads to a high level of inefficiency for data sets in cloud due to the large volume. Furthermore, this approach exhaustively checks all possible candidate GL_{new} when determining the best generalization. Unlike full-domain generalization scheme, the number of candidate GL_{new} in sub-tree generalization scheme will grow exponentially with $|\bigcup_{i=1}^{m} Cut_i|$ even when the number of attributes is constant. In a nutshell, it is a challenge for these existing approaches to scale to large-volume incremental data sets on cloud.

In summary, how to efficiently update the current generalization level GL_{cur} of an anonymized data set when new data are added is the problem we attempt to address in this paper. To this end, we propose an indexed based approach for dynamically preserving privacy of distributed and incremental data sets on cloud, which will be formulated in the next section.

4. Quasi-identifier index based approach for privacy preservation

In this section, we formulate the quasi-identifier index based approach in detail. Section 4.1 sketches out the proposed approach. In Section 4.2, we describe how generalized data sets are indexed and mapped to different data nodes. Based on the established indexes, Section 4.3 elaborates three operations to handle data updates, i.e., inserting newly added data, generalizing and specializing updated data sets.

4.1. Approach overview

In this section we sketch out our approach. At first, large-volume data sets are partitioned into a variety of relatively small data sets which are then stored in cloud data nodes. We partition original generalized data sets according to QI-groups.

Similar QI-groups are mapped into the same data nodes. These QI-groups are then indexed by domain values in their quasi-identifiers in the current generalization level. Statistical information about QI-groups is computed to facilitate data updates when new data arrive. After initialization of original anonymized data sets, we are able to handle data updates when new data are added. The new data are generalized to the current generalization level and added to corresponding QI-groups. Statistical information affected by newly added data is updated accordingly. Then we check whether k-anonymous status is violated and whether anonymized data sets are over-generalized. In the former case generalization will be performed on the data sets, while in the later case specialization will be performed. Finally, statistical information is updated again for the next update. We propose an algorithm named Quasi-identifier Indexed based Privacy Preservation algorithm (QuIPP) to realize our approach described above. Algorithm 1 briefly describes the algorithm. Details for this algorithm will be formulated in the following sections.

Algorithm 1 Quasi-identifier index based privacy preservation.

Description	Dynamically maintain k-anonymity of an anonymized data set and produce high data utility when the data set is updated with new data.
Input	A data set D and its already anonymized data set D^*, where D^* satisfies k-anonymity. Incremental data Δs.
Output	The newly anonymized data sets $(D^* + \Delta^*)$s.
Step 1	Index generalized data set D^*.
1.1 Partition D^* and D into $DNodes = \{DN_1, DN_2, \ldots, DN_n\}$. 1.2 Establish quasi-identifier index for QI-groups by domain value $q \in \bigcup_{i=1}^{m} Cut_i$. 1.3 Count statistical information for links of QI-groups and sort QI-groups in each links in terms of their anonymity.	
Step 2	Handle updates.
2.1 Generalize Δ to current generalization level GL_{cur} and map them to corresponding QI-groups. Update statistical information affected by Δ. Check the state of $(D^* + \Delta^*)$. If it violates k-anonymity, then go to Step 2.2. If the data set is over-generalized, go to Step 2.3. 2.2 Generalize $(D^* + \Delta^*)$ to satisfy k-anonymity via repeating $gen : Child(p) \to p$, where $Child(p) \subseteq \bigcup_{i=1}^{m} Cut_i$, and update affected statistical information. 2.3 Specialize $(D^* + \Delta^*)$ to expose more data utility via repeating $spec : p \to Child(p)$, where $p \in \bigcup_{i=1}^{m} Cut_i$, and update affected statistical information.	

4.2. QI-group indexing for anonymized data sets (Step 1)

In this section we first partition initial data sets into a variety of cloud storage nodes by hashing similar QI-groups into the same data nodes. We then establish quasi-identifier indexes for QI-groups of original anonymized data sets. Meanwhile, certain statistical information is calculated to achieve efficient data updates.

4.2.1. Data set partition (Step 1.1)

It is assumed that a data set D has been anonymized as D^* by sub-tree generalization scheme to satisfy k-anonymity before it is uploaded into cloud. The current generalization level for D^* is GL_{cur}. As mentioned before, both D and D^* will be stored on several data nodes $DNodes = \{DN_1, DN_2, \ldots, DN_n\}$. To facilitate data management, we introduce a controller node, denoted as CN, to keep some information like indexing information for data sets and to launch basic operations. To store D and D^* on $DNode$, it is necessary to partition a huge-volume into small ones and delivery these partitions to individual data nodes. Instead of partitioning D and D^* randomly, we propose to partition them according to QI-groups. Specifically, data records in a QI-group are aggregated as a partition. So, the number of partitions is equal to $|QID|$. These $|QID|$ partitions will be mapped to n data nodes. A hash function $h(qid) : QID \to DNodes$, is adopted to map QI-groups to data nodes.

To improve the performance of indexes of QI-groups, the hash function $h(qid)$ is expected to map similar QI-groups into the same nodes, rather than randomly map QI-groups to data nodes. Intuitively, if QI-groups located in a data node are closer in terms of their quasi-identifiers, partitioning or merging QI-groups in the update phase will become more convenient and efficient compared with the case that these similar QI-groups scatter randomly on data nodes. Moreover, placing QI-groups by this means can also facilitate indexing on QI-groups as fewer indexes across multiple data nodes will be established. As such, we incorporate locality-sensitive hashing (LSH) method [43,44] to hash QI-groups into data nodes. Locality-sensitive hash functions have the salient feature that similar input items can be mapped to the same buckets with high probability.

The distance that indicates closeness between two quasi-identifiers qid^x and qid^y is required to be defined in order to incorporate LSH method. We denote the distance between two quasi-identifiers as $dist(qid^x, qid^y)$. Let $qid^x = \langle q_1^x, q_2^x, \ldots, q_m^x \rangle$ and $qid^y = \langle q_1^y, q_2^y, \ldots, q_m^y \rangle$, then the distance can be defined as:

$$dist\left(qid^x, qid^y\right) \triangleq \sqrt{\sum_{i=1}^{m} d^2\left(q_i^x, q_i^y\right)}, \tag{1}$$

where $d(q_i^x, q_i^y)$ is the distance between q_i^x and q_i^y in the taxonomy tree TT_i. The distance between q_i^x and q_i^y can be defined as the length of the path between two domain nodes with values q_i^x and q_i^y.

With the distance defined above, a hash function can be identified to map similar QI-groups into the same data nodes. Assume d_1, d_2, where $d_1 < d_2$ are two distances. Mathematically, a function family H is called (d_1, d_2, p_1, p_2)-*sensitive* if for every h in H, any two quasi-identifiers $qid^x, qid^y \in QID$: if $dist(qid^x, qid^y) \leqslant d_1$ then $Pr[h(qid^x) = h(qid^y)] \geqslant p_1$; and if $dist(qid^x, qid^y) \geqslant d_2$ then $Pr[h(qid^x) = h(qid^y)] \leqslant p_2$. So far, several LSH families have been discovered. We adopt the LSH family proposed in [43] to hash quasi-identifiers in *QID*. In practice, we utilize an implementation named E^2LSH (Exact Euclidean LSH) [45] to fulfil the hash function $h(qid)$ required in our approach. E^2LSH is based on the LSH scheme proposed in [43]. The interested readers can refer to [43] and [44] for more details for LSH.

4.2.2. QI-group indexing (Step 1.2)

After partitioning similar QI-groups into the same data nodes, we propose to index QI-groups by domain values in quasi-identifiers to improve efficiency when data updates occur. The location of a QI-group can be uniquely identified by *DN* : *offset*. *DN* is the data node where the QI-group locates and *offset* is the exact position of the QI-group in *DN* where all the QI-groups are arranged in a list. Let $GL_{cur} = \langle Cut_1, Cut_2, \ldots, Cut_m \rangle$, and QIG_q denote the QI-groups whose *qid* contains q. For each domain value $q \in \bigcup_{i=1}^{m} Cut_i$, all QIG_q are linked as $Link_q$. Totally, there are $|\bigcup_{i=1}^{m} Cut_i|$ links. Each QI-group will be linked in m links, where m is the number of attributes of a data record. Let *LSet* represent the set of links, i.e., $LSet = \{Link_q \mid q \in \bigcup_{i=1}^{m} Cut_i\}$. Further, each QI-group has to keep m locations for its successors. Heads of links are kept in the controller node *CN*. In fact, all the linked QI-groups are the same to leaf partitions in TIPS (Taxonomy Indexed PartitionS) proposed in [33]. Note that QI-groups are retained in a variety of data nodes in cloud environments. Thus, unlike TIPS, the links in our approach have to maintain the links of QI-groups with the same quasi-identifier because these QI-groups may cross multiple data nodes. The phenomenon that data records in a QI-group scatter across multiple data nodes is caused by data generalization when data sets are updated.

For each link $ln_q \in LSet$, we sort it ascendingly in terms of the size of QI-groups in ln_q. These links are always maintained to keep sorted in this order throughout updates. Without accessing all QI-groups, states of generalized data sets can be examined quickly by these links. That is, only the relevant QI-groups are checked when data sets are generalized, specialized or added with new data. More precisely, the influences of update operations are confined to the QI-groups within a subset of *LSet* rather than all QI-groups. When generalizing a group of domain values or specializing a domain value, we can just conduct generalization or specialization on QI-groups along the corresponding links. Thereafter handle the affected links. Hence, these sorted links can improve both efficiency and scalability when updates occur.

4.2.3. Statistical information calculation (Step 1.3)

When our approach generalizes or specializes the current generalization level GL_{cur} in Steps 2.2 and 2.3 of QuIPP, there are many generalization levels can be chosen as the new generalization level. Intuitively, it is expected that the selected generalization $gen : Child(p) \to p$ can make less data utility loss while more privacy can be gained and the selected specialization $spec : p \to Child(p)$ can produce more data utility while less privacy loss can be caused. Based on [33,36], we employ the information/anonymity trade-off metric as our selection criterion.

For generalization $gen : Child(p) \to p$, the value of Information Loss per Privacy Gain (ILPG) is utilized as the selection criterion, denoted as $ILPG(gen)$. It can be calculated by

$$ILPG(gen) = IL(gen)/\big(PG(gen) + 1\big), \tag{2}$$

where $IL(gen)$ denotes the information loss and $PG(gen)$ denotes the privacy gain incurred by *gen*. Let R_x denote the set of records one of whose attribute values is generalized to x. $|R_x|$ means the number of data records in R_x. Then $IL(gen)$ can be computed by

$$IL(gen) = I(R_p) - \sum_{c \in Child(p)} \big(|R_c|/|R_p|\big) I(R_c), \tag{3}$$

where $I(R_x)$ is the entropy of R_x. The entropy $I(R_x)$ [46] can be computed by

$$I(R_x) = -\sum_{sv \in SA} \big(\#(R_x, sv)/|R_x|\big) \cdot \log_2\big(\#(R_x, sv)/|R_x|\big), \tag{4}$$

where $\#(R_x, sv)$ is the number of the data records with sensitive value sv in R_x. Let $Proj_i(qid)$ denote the ith domain value of *qid*, $i \in [1, m]$. Similarly to [33], the anonymity of domain value q is defined as the minimum group size of QI-groups whose quasi-identifiers contains q, denoted as A_q. Formally, $A_q \triangleq \min_{Proj_i(qid)=q}\{\#(QIG_q)\}$, where $\#(QIG_q)$ is the number data records in QI-groups with identifier *qid*. Then the privacy gain by generalization *gen* can be measured by the increase of anonymity, i.e.,

$$PG(gen) = A_p - \min_{c \in Child(p)} \{A_c\}. \tag{5}$$

We extend the calculation of ILPG on a set of generations. Let $GSet = \{gen_1, gen_2, \ldots, gen_M\}$. We define

$$ILPG(GSet) \triangleq \sum_{i=1}^{M} IL(gen_i) \Big/ \left(\sum_{i=1}^{M} PG(gen_i) + 1 \right). \tag{6}$$

For specialization $spec : p \rightarrow Child(p)$, the value of Information Gain per Privacy Loss (IGPL) is utilized as the selection criterion, denoted as $IGPL(spec)$. Similarly to generalization, $IGPL(spec)$ can be calculated by following formulas.

$$IGPL(spec) = IG(spec)/\big(PL(spec) + 1\big), \tag{7}$$

$$IG(spec) = I(R_p) - \sum_{c \in Child(p)} \big(|R_c|/|R_p|\big) I(R_c), \tag{8}$$

$$PL(spec) = A_p - \min_{c \in Child(p)} \{A_c\}. \tag{9}$$

To calculate ILPG and IGPL, the following statistical information of a domain value p in QI-group QIG_p is counted to facilitate calculating information gain or loss: $|R_p(qid)|$, $|R_c(qid)|$ and $\#(R_p(qid), sv)$, where $c \in Child(p)$. The minimal value of $|R_c(qid)|$, denoted as $r^p_{\min} = \min_{c \in Child(p)}\{|R_c(qid)|\}$, is used to determine whether the specialization $spec : p \rightarrow Child(p)$ will violate k-anonymity. If $r^p_{\min} \geqslant k$, the QI-group is labelled to be valid for p, i.e., $valid_p \leftarrow True$. These data are stored with their corresponding QI-groups. Note that $|R_p(qid)| = |QIG(qid)|$. With these values, we can calculate ILPG and IGPL to guide our selection. These values will be updated after new data are added or update operations has taken place.

4.3. Updates handling (Step 2)

After initial data sets have been stored and indexed, our approach can further cope with data updates efficiently. Firstly, the newly added data are generalized and hashed to corresponding data storage nodes. Then, our algorithm QuiIPP generalizes the updated data sets if k-anonymity requirement is violated. Finally, QuIPP specializes updated data sets if they are over-anonymized.

4.3.1. Data set update (Step 2.1)

When data Δ are added to the current data set, each data record in Δ is generalized to the current generalization level GL_{cur}. They are then hashed to corresponding QI-groups. If there is no existing QI-group in any data node for a record, then a new QI-group is created on that data node. A newly created QI-group $QIG(qid')$ should be linked into links in $\{ln_q \mid q = Proj_i(qid'),\ i \in [1, m]\}$. Let $NLSet$ denote the set of links which link the new QI-groups, i.e., $NLSet = \bigcup_{qid'}\{ln_q \mid q \in \bigcup\{Proj_i(qid') \mid i \in [1, m]\}\}$. The links in $NLSet$ imply that the QI-groups in these links possibly violate k-anonymity. When a record is added to a QI-group $QIG(qid)$, the count $|R_p(qid)|$, $|R_c(qid)|$, $\#(R_p(qid), sv)$, $r^c_{\min}$ and $valid_c$ are updated for each domain values in qid accordingly. Further, let $ALSet$ denote the set of the links affected by the newly added data, i.e., $ALSet = \{ln_q \mid q \in \bigcup\{Proj_i(qid) \mid i \in [1, m]\}\}$, where the QI-group corresponding to qid is added with new records. Note that $NLSet \subseteq ALSet$. After all records in Δ are mapped into the QI-groups in data nodes, the links in $ALSet$ are re-sorted to keep the ascending order.

To determine whether k-anonymity is violated, we just need to check the links in $NLSet$. Let $QIG(qid)$ be the first QI-group in ln_q. If $|QIG(qid)| \geqslant k$, ln_q will be removed from $NLSet$, meaning that the QI-groups containing q comply with k-anonymity. If $|QIG(qid)| < k$, $QIG(qid)$ is said to be a non-k-anonymous QI-group. Let $NKSet$ denote the set of identifiers of non-k-anonymous QI-groups. $NKSet$ can be identified by checking the first QI-group of each link in $NLSet$. If $NKSet$ is not empty, indicating the current generalized data set has violated k-anonymity, we need to generalize the data set to satisfy k-anonymity. The details about how to generalize the data set are formulated in Section 4.3.2.

After handling all the generalizations, we then check whether over-generalization exists in the current anonymized data set. A link $ln_q \in ALSet$ is said to be valid, if all QI-groups in link ln_q are valid for q, i.e., $valid_q = True$. Thus, for a valid link ln_q, the domain value q can be specialized without compromising k-anonymity. If any QI-group in ln_q fails to be valid for q, ln_q is removed from $ALSet$. Through checking all links in $ALSet$, we can identify the candidate links for specialization. Let $CLSet$ denote these valid candidate links, i.e., $CLSet = \{ln_q\}$, where ln_q is a valid link. After identifying $CLSet$, elements in $ALSet$ are removed. Once $CLSet$ is identified, specialization will be conducted on the data set. The details about how to specialize the data set are formulated in Section 4.3.3.

4.3.2. Generalization (Step 2.2)

To fulfil k-anonymity, the data set will be generalized to the new generalization level GL_{new}, where $GL_{cur} < GL_{new}$. Identifying a proper GL_{new} is a key step for generalization. The approach proposed in [32] exhaustively enumerates all the possible generalization levels higher than GL_{cur}. Let $\#(gen)$ denote the number of all possible generalizations: $gen : Child(p) \rightarrow p$, where $Child(p) \subseteq \bigcup_{i=1}^{m} Cut_i$. Then, there will be $2^{\#(gen)}$ generalization level candidates for GL_{new}. Thus, the exhaustive approach suffers from inefficiency.

The purpose of generalization is to merge the records in existing QI-group with that in the new QI-groups whose sizes are less than k. Meanwhile, less data utility should be reduced by generalization. Intuitively, a non-k-anonymous QI-group $QIG(qid')$ should be merged with an existing QI-group whose quasi-identifier qid is similar to qid'. The reason is that if qid is similar to qid', then the extent of generalization required to eliminate the difference between qid and qid' will be relatively small. The nearest quasi-identifiers of a non-k-anonymous QI-group can be identified in the sense of the distance for identifiers defined in Section 4.2.1. Since most similar QI-groups have been hashed into the same data nodes with

LSH, we can find the nearest quasi-identifiers of qid' in the same data node with high probability. As such, the nearest quasi-identifiers of qid' within a data node rather than the globally nearest qids are checked to generate the candidate generalization levels for GL_{new}.

Let qid be one of the nearest quasi-identifiers of $qid' \in NKSet$. We can generate a candidate generalization level $GL_{cand} = \langle Cut_{cand_1}, Cut_{cand_2}, \ldots, Cut_{cand_m} \rangle$ from qid, qid' and $GL_{cur} = \langle Cut_{cur_1}, Cut_{cur_2}, \ldots, Cut_{cur_m} \rangle$. Let pid be the quasi-identifier of a QI-group that consists of the data records in both $QIG(qid)$ and $QIG(qid')$, $pid = \langle p_1, p_2, \ldots, p_m \rangle$. Further, let $q_i = Proj_i(qid)$ and $q'_i = Proj_i(qid')$, where $i \in [1, m]$. If $q_i = q'_i$, then $Cut_{cand_i} = Cut_{cur_i}$, i.e., no generalization is needed on attribute $Attr_i$. Further, we have $p_i \leftarrow q_i$ or q'_i. If $q_i \neq q'_i$, generalization should be conducted on this attribute. Let $p_i \in DOM_i$ be the Lowest Common Ancestor (LCA) of q_i and q'_i, i.e., $p_i = LCA(q_i, q'_i)$. Let $Parent(q_i)$ and $Parent(q'_i)$ denote the parents of q_i and q'_i, respectively. The generalization $gen : Child(Parent(q_i)) \cup Child(Parent(q'_i)) \rightarrow p_i$ is then utilized to generalize attribute $Attr_i$. Then, $Cut_{cand_i} \leftarrow (Cut_{cur_i} - (Child(Parent(q_i)) \cup Child(Parent(q'_i)))) \cup \{p_i\}$ is used to construct Cut_{cand_i}. Let $GSet = \{gen_1, gen_2, \ldots, gen_M\}$ be the set of generalizations required to generate GL_{cand}. $GSet$ is retained with GL_{cand} to calculate $ILPG(GSet)$. After this generalization, the size of $QIG(pid)$ is larger than k. The sizes of other merged QI-groups are also not less than k. Hence, the data set can still be k-anonymous if it is generalized by generalizations in $GSet$.

Usually, there exist several candidate generalization levels for GL_{new}, because more than one quasi-identifier can be the nearest quasi-identifiers of qid'. According to the selection criterion defined in Section 4.2.3, the generalizations in $GSet$ with the lowest $ILPG(GSet)$ will be selected to generalize the data set to generalization level GL_{new}. Let generalization $gen \in GSet$, $gen : Descendant(p_i) \rightarrow p_i$, where $Descendant(p_i)$ is the set of descendants generalized to p_i. For each QI-group $QIG(qid)$ in each link ln_d, where $d \in Descendant(p_i)$, we then count statistics for imaginary ln_{p_i}: $|R_{p_i}|$, $|R_d|$, $\#(R_{p_i}, sv)$ and $\#(R_d, sv)$. Specifically, $|R_{p_i}| \leftarrow |R_{p_i}| + |R_d(qid)|$, $|R_d| \leftarrow |R_d| + |R_d(qid)|$, $\#(R_{p_i}, sv) \leftarrow \#(R_{p_i}, sv) + \#(R_d(qid), sv)$ and $\#(R_d, sv) \leftarrow \#(R_d, sv) + \#(R_d(qid), sv)$, respectively. With $|R_{p_i}|$, $|R_d|$, $\#(R_{p_i}, sv)$ and $\#(R_d, sv)$, Eq. (3) can be utilized to compute $IL(gen)$. To compute $PG(gen)$, we need to obtain the anonymity A_{p_i}. Suppose all the links in $\{ln_q \mid q \in Descendant(p_i)\}$ are merged as one link ln_{p_i} by inserting QI-groups in each links into ln_{p_i}. Two QI-groups in ln_{p_i} are merged if their quasi-identifiers are the same. So we can obtain the minimal QI-group size of the new QI-groups in ln_{p_i}. Note that this process is imaginary and it is unnecessary to merge links in $\{ln_q \mid q \in Descendant(p_i)\}$ substantially. However, this imaginary process helps us to get the minimal QI-group size. Specifically, the sizes of QI-groups in links $\{ln_q \mid q \in Descendant(p_i)\}$ are accumulated if the quasi-identifiers only differ on attribute $Attr_i$. Then A_{p_i} is the minimum of these accumulative values. A_q is the group size of the first QI-group in ln_q. Then, $PG(gen)$ can be calculated by Eq. (5). $ILPG(GSet)$ can be computed by Eq. (6).

After identifying a candidate generalization level for GL_{new}, we can conduct generalizations on the data set. Let $GSet$ be the generalizations required for the new generalization level GL_{new}. For each generalization $gen \in GSet$, $gen : Descendant(p_i) \rightarrow p_i$, a new link ln_{p_i} is created. The quasi-identifiers of QI-groups in links $\{ln_q \mid q \in Descendant(p_i)\}$ are revised by replacing q with p_i. Then all the QI-groups are inserted into ln_{p_i}. Let $QIG_q(qid)$ be the QI-group that will be inserted, and qid' be the quasi-identifier of $QIG_q(qid')$ before generalized to qid. If no QI-group in ln_{p_i} has the quasi-identifier qid, then $QIG_q(qid)$ is added to ln_{p_i}. The count statistics updated as follows. $|R_{p_i}(qid)| \leftarrow |R_p(qid')|$, $|R_d(qid)| \leftarrow |R_p(qid')|$, where d generalized from q is a child of p_i, and $\#(R_{p_i}(qid), sv) \leftarrow \#(R_p(qid'), sv)$. If a QI-group $QIG(qid)$ is already in ln_{p_i}, another QI-group $QIG_q(qid)$ will be merged into $QIG(qid)$ when we insert it into ln_{p_i}. Note that merging these two QI-groups will affect the links in $\{ln_x\}$, where x is any domain value in qid except p_i. Thus, it is necessary to remove QI-group $QIG_q(qid)$ from links in $\{ln_x\}$. The count statistics for $QIG(qid)$ is updated. Specifically, $|R_{p_i}(qid)| \leftarrow |R_{p_i}(qid)| + |QIG_q(qid')|$, $|R_d(qid)| \leftarrow |R_d(qid)| + |QIG_q(qid')|$, where d generalized from q is a child of p_i, and $\#(R_{p_i}(pid), sv) \leftarrow \#(R_{p_i}(pid), sv) + \#(R_p(qid'), sv)$. After all QI-groups are inserted, $r^d_{\min}$ and $valid_d$ for QI-groups in ln_{p_i} are also updated accordingly. Links in $\{ln_{p_i}\} \cup \{ln_x\}$ are re-sorted according to QI-group size. Remove links in $\{ln_q \mid q \in Descendant(p_i)\}$ from $ALSet$.

Since we have placed most similar QI-groups on the same data nodes, the QI-groups merged into the same new QI-groups can locate in the some data nodes with high probability. That is, most merging actions occur within a data node. When multiple QI-groups with the same quasi-identifier across several data nodes, we do not merge them into one data node in order to avoid communication overhead. Instead, one QI-group with their statistical information represents the logically merged QI-group, while the rest are purely linked to the representative after it is removed from its original links. In this situation, a QI-group has a link to connect its records across a variety of data nodes.

We remove the qid' from $NKSet$ and generalize the rest quasi-identifiers in $NKSet$ to GL_{new}. Again, we check each identifier in $NKSet$ whether its QI-group is k-anonymous. If yes, the corresponding identifier is removed from $NKSet$. If $NKSet \neq \Phi$, repeat the generalization process.

4.3.3. Specialization (Step 2.3)

Usually, more than one domain value can be used for specialization, i.e., $|CLSet| > 1$. According to the selection criterions defined in Section 4.1, the specialization with the highest $IGPL(spec)$ will be selected to specialize the data set to generalization level GL_{new} Let $spec : p \rightarrow Child(p)$. For each QI-group $QIG(qid)$ in link ln_p, $|R_p| \leftarrow |R_p| + |R_p(qid)|$, $|R_c| \leftarrow |R_c| + |R_c(qid)|$ and $\#(R_p, sv) \leftarrow \#(R_p, sv) + \#(R_p(qid), sv)$. But we have to access the data records in $QIG(qid)$ to acquire statistical information $\#(R_c(qid), sv)$. After this, $\#(R_c, sv) \leftarrow \#(R_c, sv) + \#(R_c(qid), sv)$. With $|R_p|$, $|R_c|$, $\#(R_p, sv)$ and $\#(R_c, sv)$, Eq. (8) can be utilized to compute $IG(spec)$. According to the definition of A_c, we can derive that $\min_{c \in Child(p)}\{A_c\} = \min_{ln_p}\{r^p_{\min}\}$, because both sides of the equation is the minimum QI-group size after specialization $spec$.

Hence, the privacy loss can be computed by $PL(spec) = A_p - \min_{ln_p}\{r^p_{\min}\}$. This formula gets the same result of Eq. (9). Finally, $IGPL(spec)$ can be computed by Eq. (7).

After identifying a candidate specialization, we then conduct specialization on the data set. Let the candidate specialization be $spec : p \to Child(p)$. New links $\{ln_c \mid c \in Child(p)\}$ are created at first. Each QI-group $QIG(qid)$ in ln_p is divided into several new QI-groups according to $Child(p)$. Each new QI-group is linked to their corresponding links in $\{ln_c \mid c \in Child(p)\}$. Besides, these QI-groups are also linked into other links in which $QIG(qid)$ used to be. Generally speaking, this process is a reverse process of merging QI-groups in generalization. During this partitioning process, the statistical information for new QI-groups is updated. Let $QIG_c(qid')$ be a new QI-group from $QIG(qid)$. Its statistical information can be updated as follows. We can directly get $|R_p(qid')| \leftarrow |R_c(qid)|$. But for $|R_c(qid')|$ and $\#(R_p(qid'), sv)$, we have to access the data records in $QIG_c(qid')$. $r^c_{\min}$ and $valid_c$ can also be updated accordingly. Note that specialization $spec$ will affect the links in $\{ln_x\}$, where x is any domain value in qid except p. Thus, links in $\{ln_c\} \cup \{ln_x\}$ are re-sorted according to QI-group size. Remove ln_p from $CLSet$ and add new links in $\{ln_c\}$ to $CLSet$. For each link ln_x in $CLSet$, ln_x is removed from $CLSet$ if it is invalid. Finally, if $CLSet \neq \Phi$, the algorithm repeats the specialization process.

5. Evaluation

In this section, we evaluate our approach via conducting experiments. We first compare our approach with existing approaches in overall terms in Section 5.1 and then conduct quantitative experiments on real world data sets in Section 5.2.

5.1. Overall comparison

As mentioned in Section 3.2.2, existing approaches can be classified into three categories to maintain privacy requirements. The work in [32,39] shows that incrementally preserving privacy on incremental data sets outperforms the static approach in first category. This is plausible because re-anonymizing entire updated data sets with new data from scratch will be difficult for large-volume incremental data sets in cloud. Therefore, we compare our approach with the second and third categories, i.e., the incremental and distributed approaches. With regard to huge-volume of data sets, the existing approaches have two shortcomings. One is that they access all data sets multiple times when updates occur. The other is that they globally check all the possible generalization or specialization candidates when determining which one should be selected. As a result, these approaches suffer from inefficiency and poor scalability when data updates occur frequently in the sub-tree generalization scheme.

Compared with existing approaches, our approach makes only accesses a relatively small subset of QI-groups when generalizing or specializing anonymized data sets with the established indexing structure of QI-groups. The length of a link is much smaller than the number of all QI-groups, because it is equal to the number of QI-groups having a common domain value in their quasi-identifiers. Further, the statistical information retained with QI-groups reduces the times of accessing data records. More precisely, data records are only accessed in specialization operation when their corresponding QI-groups are partitioned. Hence, we can utilize statistical information directly for data updates and generalization. Furthermore, due to the adoption of LSH method in our approach, the number of links across multiple data nodes can be small with high probability. Thus, an update operation can be accomplished within a data node with high probability, reducing the communication overhead to maintain the links. Existing distributed approach involves all the data nodes because it randomly hashes data records into data nodes [32]. Above all, our approach can be more efficient and scalable compared to the existing approaches. To further evaluate our approach, we conduct experimental evaluation on real-world data sets in the next section.

5.2. Experiment evaluation

5.2.1. Experiment environment

U-Cloud is a cloud computing environment at University of Technology Sydney (UTS). We conduct our experiments on U-Cloud. The system overview of the U-Cloud system is depicted in Fig. 2. The computing facilities of this system are located among several labs in the Faculty of Engineering and IT, UTS. On top of hardware and Linux operating system, we install KVM virtualization software [47] which virtualizes the infrastructure and provides unified computing and storage resources. To create virtualized data centres, we install OpenStack open source cloud environment [48] which is responsible for virtual machine management, resource scheduling, task distribution and interaction with users. Furthermore, Hadoop [35] is installed based on the private cloud built via OpenStack to facilitate MapReduce computing paradigm and massive data processing. Our experiments are conducted in this cloud environment.

5.2.2. Experiment process

We use the Adult data set from UCI Machine Learning Repository [49] for our experiments. This data set is a commonly-used data set in the privacy-preserving data publishing research community. After pre-processing the Adult data set, the sanitized data set consists of 30 162 records. In this data set, each record has 14 attributes. We utilize eight attributes out of them in our experiments. The basic information about attribute taxonomy trees is described in Table 1, in which the number of domain values and level of trees are listed. The attribute *Work Class* is utilized as the sensitive attribute. This

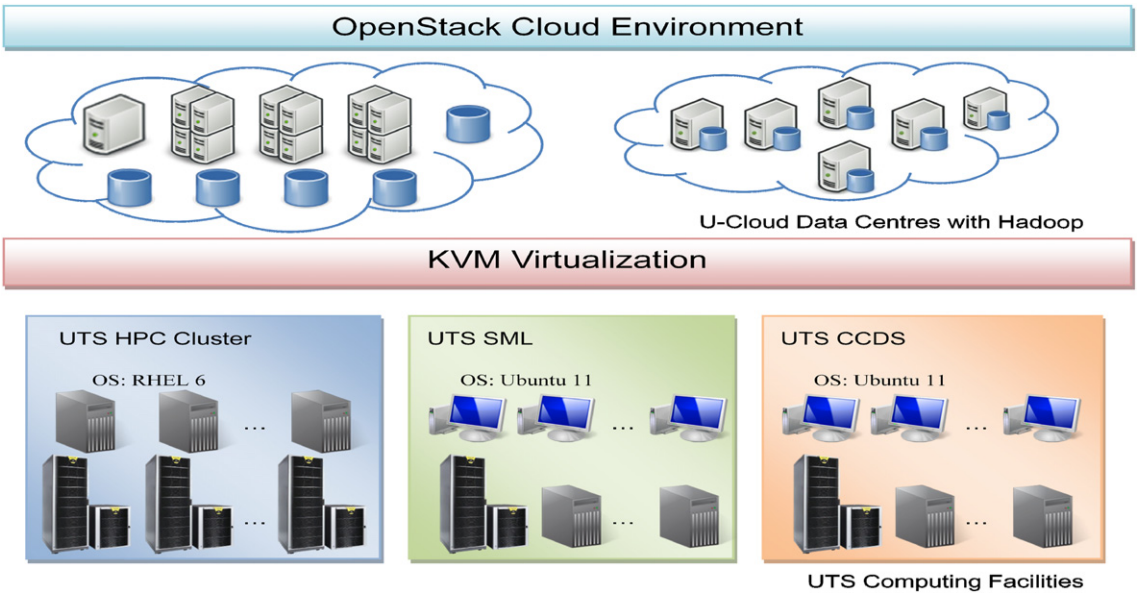


Fig. 2. System overview of U-Cloud.

Table 1
Description of the Adult data set.

Attribute	Age	Education	Marital status	Occupation	Relationship	Race	Sex	Native country
Level	5	4	3	4	3	3	2	4
Domain value	99	26	9	23	8	7	3	50

sensitive attribute has seven different categorical values. First of all, we sample randomly 10 000 data records out of the sanitized data set as the initial data set and anonymize it to a given level of k-anonymity with the sub-tree generalization method proposed in [33]. We then generate ten data sets out of the rest of the sanitized data set as update batches by randomly sampling. The size of the generated update batch ranges from 2000 (2 K) to 20 000 (20 K).

Our approach is implemented as QuIPP algorithm in Java. Initially, the controller node constructs quasi-identifier indexing structure for the current QI-groups in the initial data set. The statistical information is also count in this step. All QI-groups are then hashed into different data nodes. In our experiment setting, the number of data nodes is set to six. The head of each link is retained in the controller node while the statistical information is kept with corresponding QI-groups. A representative existing approach is implemented according to the proposed algorithms in [32]. Rather than full-domain generalization scheme, we use sub-tree generalization scheme in the existing approach for comparison. Then both two approaches are fed with the same update batches. The time for running a data update by QuIPP is denoted as t_I, while the time for the existing approach is denoted as t_E. Both approaches are repeated 50 times for each update batch. The mean of their run time is regarded as the time to complete the update.

5.2.3. Experiment results and analysis

The experimental result on the real-world data sets is depicted in Fig. 3. Fig. 3 illustrates how the difference between t_I and t_E changes with the number of data records in an update batch when k is fixed. Since k is a user defined anonymity threshold, we select four values: 3, 5, 10 and 15, to conduct our experiments. The selection of these specific values is rather random and does not affect our analysis because what we want to see is the trend of t_I against t_E over the size of update batches. We set the number of values as four to informatively conduct the experiments. The interested readers can try other values of k. The conclusions will be similar.

From (a), (b), (c) and (d) in Fig. 3, we can see that both t_I and t_E go up when the size of an update batch is getting larger. That is, the larger an update batch is, the more time cost will be incurred. The increase of t_E is slightly exponential to the size of an update batch. According to analysis in Section 4.3.2, t_E is exponential to the number of domain values in a generalization level. The increase of data records implies the increase in number of domain values because more new QI-groups may be created. Moreover, all the data records are accessed several times, resulting in considerable increase in t_E. For t_I, a larger update batch means more links will be established and the length of links will also become longer. Thus, the time cost increases as well.

Most importantly, we can see from Fig. 3 that the difference between t_I and t_E becomes bigger and bigger when the number of data records increases. That is, more time cost can be reduced by QuIPP than existing approaches when the

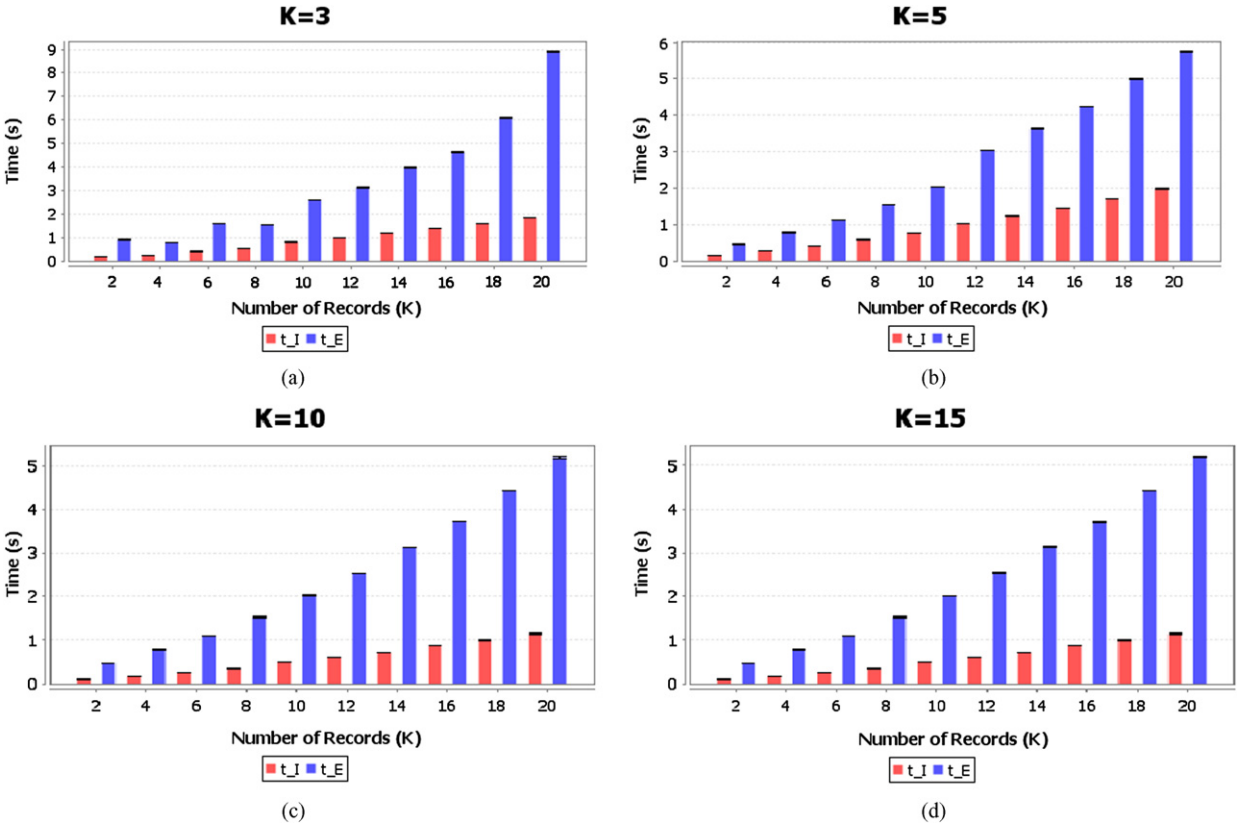


Fig. 3. Update time w.r.t. number of update records.

volume of a data set becomes larger. This trend is the result of the sharp rise in t_E and relatively slower increase in t_I when the number of data records is increased. Usually, the volume of data sets is quite huge in the context of cloud. Hence, this trend demonstrates that our approach can significantly improve the efficiency of privacy preservation on large-volume incremental data sets over existing approaches.

6. Conclusion and future work

In this paper, we have investigated the challenge about how to efficiently update huge-volume incremental data sets to ensure privacy requirements of data owners and simultaneously achieve high data utility to data users. We have proposed an efficient quasi-identifier index based approach for privacy preservation over incremental data sets on cloud. In our approach, QI-groups (QI: quasi-identifier) are indexed by the domain values in the current generalization level, which makes it possible to access only a part of records in a data set in the presence of data updates rather than access all data records as required by existing approaches. To further improve the performance of quasi-identifier indexing, locality-sensitive hashing method is incorporated to place similar QI-groups on the same data storage nodes. Thus, the number of data nodes that a QI-group link across will be reduced considerably with high probability. Based on the established indexes of an anonymized data set, we have designed an efficient quasi-identifier index based privacy preservation algorithm (QuIPP) for our approach. Evaluation results on real-world data sets have demonstrated that with our approach, the efficiency of privacy preservation on large-volume incremental data sets can be improved significantly over existing approaches.

In accordance with various data and computation intensive applications on cloud, processing of huge-volume incremental data sets is becoming an important research area. Privacy preservation for such data sets is one of important yet challenging research issues, and needs thorough investigation. With the contributions of this paper, we plan to investigate privacy-aware efficient scheduling of anonymized data sets in cloud by taking privacy preservation as a metric together with other metrics such as storage and computation in the future.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58.

[2] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Comput. Syst. 25 (6) (2009) 599–616.

[3] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, C. Fu, Cloud computing: A perspective study, New Generation Comput. 28 (2) (2010) 137–146.
[4] L. Wang, J. Zhan, W. Shi, Y. Liang, In cloud, can scientific communities benefit from the economies of scale?, IEEE Trans. Parallel Distrib. Syst. 23 (2) (2012) 296–303.
[5] X. Yang, L. Wang, G. Laszewski, Recent research advances in e-science, Cluster Comput. 12 (4) (2009) 353–356.
[6] H. Takabi, J.B.D. Joshi, G. Ahn, Security and privacy challenges in cloud computing environments, IEEE Secur. Privacy 8 (6) (2010) 24–31.
[7] D. Zissis, D. Lekkas, Addressing cloud computing security issues, Future Generation Comput. Syst. 28 (3) (2011) 583–592.
[8] G. Bernd, W. Tobias, S. Elmar, Understanding cloud computing vulnerabilities, IEEE Secur. Privacy 9 (2) (2010) 50–57.
[9] W.A. Jansen, Cloud hooks: Security and privacy issues in cloud computing, in: Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS'11), 2011, pp. 1–10.
[10] W. Jie, W. Cai, L. Wang, R. Procter, A secure information service for monitoring large scale grids, Parallel Comput. 33 (7–8) (2007) 572–591.
[11] Microsoft HealthVault, http://www.microsoft.com/health/ww/products/Pages/healthvault.aspx, accessed on February 10, 2012.
[12] H. Lin, W. Tzeng, A secure erasure code-based cloud storage system with secure data forwarding, IEEE Trans. Parallel Distrib. Syst. 23 (6) (2011) 995–1003.
[13] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, in: Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM'11), 2011, pp. 829–837.
[14] M. Li, S. Yu, N. Cao, W. Lou, Authorized private keyword search over encrypted data in cloud computing, in: Proceedings of the 31st International Conference on Distributed Computing Systems (ICDCS'11), 2011, pp. 383–392.
[15] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, W. Lou, Fuzzy keyword search over encrypted data in cloud computing, in: Proceedings of the 31st Annual IEEE International Conference on Computer Communications (INFOCOM'11), 2010, pp. 1–5.
[16] C. Gentry, Fully homomorphic encryption using ideal lattices, in: Proceedings of the 41st annual ACM Symposium on Theory of Computing (STOC'09), 2009, pp. 169–178.
[17] M. van Dijk, C. Gentry, S. Halevi, V. Vaikuntanathan, Fully homomorphic encryption over the integers, in: Proceedings of the 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'10), 2010, pp. 24–43.
[18] K. LeFevre, D.J. DeWitt, R. Ramakrishnan, Incognito: Efficient full-domain k-anonymity, in: Proceedings of 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD'05), 2005, pp. 49–60.
[19] X. Xiao, Y. Tao, Anatomy: Simple and effective privacy preservation, in: Proceedings of 32nd International Conference on Very Large Data Bases (VLDB'06), 2006, pp. 139–150.
[20] P. Samarati, Protecting respondents' identities in microdata release, IEEE Trans. Knowl. Data Engrg. 13 (6) (2001) 1010–1027.
[21] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, L-diversity: Privacy beyond k-anonymity, ACM Trans. Knowl. Discov. Data 1 (1) (2007), Article 3.
[22] P. Bhatotia, A. Wieder, R. Rodrigues, U.A. Acar, R. Pasquin, Incoop: Mapreduce for incremental computations, in: Proceedings of Proceedings of the 2nd ACM Symposium on Cloud Computing (SoCC'11), 2011, pp. 1–14.
[23] B. Li, E. Mazur, Y. Diao, A. McGregor, P. Shenoy, A platform for scalable one-pass analytics using mapreduce, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11), 2011, pp. 985–996.
[24] R. Kienzler, R. Bruggmann, A. Ranganathan, N. Tatbul, Stream as you go: The case for incremental data access and processing in the cloud, in: IEEE ICDE International Workshop on Data Management in the Cloud (DMC'12), 2012.
[25] C. Olston, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson, A. Neumann, V.B.N. Rao, V. Sankarasubramanian, S. Seth, C. Tian, T. ZiCornell, X. Wang, Nova: Continuous Pig/Hadoop workflows, in: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'11), 2011, pp. 1081–1090.
[26] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[27] X. Xiao, Y. Tao, M-invariance: Towards privacy preserving re-publication of dynamic datasets, in: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD'07), 2007, pp. 689–700.
[28] B.C.M. Fung, K. Wang, A.W.-C. Fu, J. Pei, Anonymity for continuous data publishing, in: Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology (EDBT'08), 2008, pp. 264–275.
[29] Y. He, S. Barman, J.F. Naughton, Preventing equivalence attacks in updated, anonymized data, in: Proceedings of the IEEE 27th International Conference on Data Engineering (ICDE'11), 2011, pp. 529–540.
[30] J. Pei, J. Xu, Z. Wang, W. Wang, K. Wang, Maintaining k-anonymity against incremental updates, in: Proceedings of the 19th International Conference on Scientific and Statistical Database Management (SSBDM'07), 2007, pp. 5–5.
[31] B. Zhou, Y. Han, J. Pei, B. Jiang, Y. Tao, Y. Jia, Continuous privacy preserving publishing of data streams, in: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (EDBT'09), 2009, pp. 648–659.
[32] K. Doka, D. Tsoumakos, N. Koziris Kanis, Preserving k-anonymity over distributed data, in: The 5th International Workshop on Personalized Access, Profile Management, and Context Awareness in Databases (PersDB'11), 2011.
[33] B.C.M. Fung, K. Wang, P.S. Yu, Anonymizing classification data for privacy preservation, IEEE Trans. Knowl. Data Engrg. 19 (5) (2007) 711–725.
[34] K.H. Lee, Y.J. Lee, H. Choi, Y.D. Chung, B. Moon, Parallel data processing with mapreduce: A survey, ACM SIGMOD Record 40 (4) (2012) 11–20.
[35] Hadoop, http://hadoop.apache.org, accessed on June 01, 2012.
[36] B.C.M. Fung, K. Wang, R. Chen, P.S. Yu, Privacy-preserving data publishing: A survey of recent developments, ACM Computer Survey 42 (4) (2010) 1–53.
[37] N. Li, T. Li, S. Venkatasubramanian, Closeness: A new privacy measure for data publishing, IEEE Trans. Knowl. Data Engrg. 22 (7) (2010) 943–956.
[38] J.-W. Byun, Y. Sohn, E. Bertino, N. Li, Secure anonymization for incremental datasets, in: Proceedings of the VLDB Workshop on Secure Data Management (SDM'06), 2006, pp. 48–63.
[39] T.M. Truta, A. Campan, K-anonymization incremental maintenance and optimization techniques, in: Proceedings of Proceedings of the 2007 ACM Symposium on Applied Computing (SAC'07), 2007, pp. 380–387.
[40] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, A.W.C. Fu, Utility-based anonymization using local recoding, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data (KDD'06), 2006, pp. 785–790.
[41] K. LeFevre, D.J. DeWitt, R. Ramakrishnan, Mondrian multidimensional k-anonymity, in: Proceedings of 22nd International Conference on Data Engineering (ICDE'06), 2006, pp. 25–25.
[42] X. Xiao, Y. Tao, Personalized privacy preservation, in: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data (SIGMOD'06), 2006, pp. 229–240.
[43] A. Alexandr, I. Piotr, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, in: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 2006, pp. 459–468.
[44] M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Proceedings of the 20th Annual Symposium on Computational Geometry (SOCG'04), 2004, pp. 253–262.
[45] E2LSH, LSH algorithm and implementation, http://web.mit.edu/andoni/www/LSH/, accessed on February 10, 2012.
[46] C.E. Shannon, A mathematical theory of communication, SIGMOBILE Mobile Comput. Commun. Review 5 (1) (2001) 3–55.

[47] KVM, http://www.linux-kvm.org/page/Main_Page, accessed on February 10, 2012.
[48] OpenStack, http://openstack.org/, accessed on February 10, 2012.
[49] U.C.I. Machine, Learning repository, ftp://ftp.ics.uci.edu/pub/machine-learning-databases/, accessed on June 01, 2012.