A Privacy Leakage Upper-bound Constraint based Approach for Cost-effective Privacy Preserving of Intermediate Datasets in Cloud

Xuyun Zhang, Chang Liu, Surya Nepal, Suraj Pandey, Jinjun Chen, Member, IEEE

Abstract—Cloud computing provides massive computation power and storage capacity which enable users to deploy computation and data intensive applications without infrastructure investment. Along the processing of such applications, a large volume of intermediate datasets will be generated, and often stored to save the cost of re-computing them. However, preserving the privacy of intermediate datasets becomes a challenging problem because adversaries may recover privacy-sensitive information by analyzing multiple intermediate datasets. Encrypting ALL datasets in cloud is widely adopted in existing approaches to address this challenge. But we argue that encrypting all intermediate datasets are neither efficient nor cost-effective because it is very time consuming and costly for data-intensive applications to en/decrypt datasets frequently while performing any operation on them. In this paper, we propose a novel upper-bound privacy leakage constraint based approach to identify which intermediate datasets need to be encrypted and which do not, so that privacy-preserving cost can be saved while the privacy requirements of data holders can still be satisfied. Evaluation results demonstrate that the privacy-preserving cost of intermediate datasets can be significantly reduced with our approach over existing ones where all datasets are encrypted.

Index Terms—Cloud Computing, Data Storage Privacy, Privacy Preserving, Intermediate Dataset, Privacy Upper Bound

1 INTRODUCTION

Technically, cloud computing is regarded as an ingenious combination of a series of technologies, establishing a novel business model by offering IT services and using economies of scale [1], [2]. Participants in the business chain of cloud computing can benefit from this novel model. Cloud customers can save huge capital investment of IT infrastructure, and concentrate on their own core business [3]. Therefore, many companies or organizations have been migrating or building their business into cloud. However, numerous potential customers are still hesitant to take advantage of cloud due to security and privacy concerns [4], [5].

The privacy concerns caused by retaining intermediate datasets in cloud are important but they are paid little attention. Storage and computation services in cloud are equivalent from an economical perspective because they are charged in proportion to their usage [1]. Thus, cloud users can store valuable intermediate datasets selectively when processing original datasets in data-intensive applications like medical diagnosis, in order to curtail the overall expenses by avoiding frequent re-computation to obtain these datasets [6], [7]. Such scenarios are quite common because data users often re-analyze results, conduct new analysis on intermediate datasets, or share some intermediate results with others for collaboration. Without loss of generality, the notion of intermediate dataset herein refers to intermediate and resultant datasets [6]. However, the storage of intermediate data enlarges attack surfaces so that privacy requirements of data holders are at risk of being violated. Usually, intermediate datasets in cloud are accessed and processed by multiple parties, but rarely controlled by original dataset holders. This enables an adversary to collect intermediate datasets together and menace privacy-sensitive information from them, bringing considerable economic loss or severe social reputation impairment to data owners. But little attention has been paid to such a cloud-specific privacy issue.

Existing technical approaches for preserving the privacy of datasets stored in cloud mainly include encryption and anonymization. On one hand, encrypting all datasets, a straightforward and effective approach, is widely adopted in current research [8], [9], [10]. However, processing on encrypted datasets efficiently is quite a challenging task, because most existing applications only run on unencrypted datasets. Although recent progress has been made in homomorphic encryption which theoretically allows performing computation on encrypted datasets, applying current algorithms are rather expensive due to their inefficiency [11]. On the other hand, partial information of datasets, e.g., aggregate information, is required to expose to data users in most cloud applications like data mining and analytics. In such cases, datasets are anonymized rather than encrypted to ensure both data utility and privacy preserving. Current privacy-preserving techniques like generalization [12] can withstand most privacy attacks on one single dataset, while preserving privacy for multiple datasets is still a challenge.
ing problem [13]. Thus, for preserving privacy of multiple datasets, it is promising to anonymize all datasets first and then encrypt them before storing or sharing them in cloud. Usually, the volume of intermediate datasets is huge [6]. Hence, we argue that encrypting all intermediate datasets will lead to high overhead and low efficiency when they are frequently accessed or processed. As such, we propose to encrypt part of intermediate datasets rather than all for reducing privacy-preserving cost.

In this paper, we propose a novel approach to identify which intermediate datasets need to be encrypted while others do not, in order to satisfy privacy requirements given by data holders. A tree structure is modeled from generation relationships of intermediate datasets to analyze privacy propagation of datasets. As quantifying joint privacy leakage of multiple datasets efficiently is challenging, we exploit an upper-bound constraint to confine privacy disclosure. Based on such a constraint, we model the problem of saving privacy-preserving cost as a constrained optimization problem. This problem is then divided into a series of sub-problems by decomposing privacy leakage constraints. Finally, we design a practical heuristic algorithm accordingly to identify the datasets that need to be encrypted. Experimental results on real-world and extensive datasets demonstrate that privacy-preserving cost of intermediate datasets can be significantly reduced with our approach over existing ones where all datasets are encrypted.

The major contributions of our research are threefold. Firstly, we formally demonstrate the possibility of ensuring privacy leakage requirements without encrypting all intermediate datasets when encryption is incorporated with anonymization to preserve privacy. Secondly, we design a practical heuristic algorithm to identify which datasets need to be encrypted for preserving privacy while the rest of them do not. Thirdly, experiment results demonstrate that our approach can significantly reduce privacy-preserving cost over existing approaches, which is quite beneficial for the cloud users who utilize cloud services in a pay-as-you-go fashion.

This paper is a significantly improved version of [14]. Based on [14], we mathematically prove that our approach can ensure privacy-preserving requirements. Further, the heuristic algorithm is re-designed by considering more factors. We extend experiments over real datasets. Our approach is also extended to a graph structure.

The remainder of this paper is organized as follows. The related work is reviewed in the next section. A motivating example and problem analysis are given in Section 3. In Section 4, we present the fundamental privacy representation of datasets and derive privacy leakage upper-bound constraints. Section 5 formulates our approach. In Section 6, we evaluate the proposed approach by conducting experiments on both real-world datasets and extensive datasets. Finally, we conclude this paper and discuss our future work in Section 7.

2 RELATED WORK

We briefly review the research on privacy protection in cloud, intermediate dataset privacy preserving and Privacy-Preserving Data Publishing (PPDP).

Currently, encryption is exploited by most existing research to ensure the data privacy in cloud [8], [9], [10]. Although encryption works well for data privacy in these approaches, it is necessary to encrypt and decrypt datasets frequently in many applications. Encryption is usually integrated with other methods to achieve cost reduction, high data usability and privacy protection. Roy et al. [15] investigated the data privacy problem caused by MapReduce and presented a system named Airavat which incorporates mandatory access control with differential privacy. Puttaswamy et al. [16] described a set of tools called Silverline that identifies all functionally encryp-

ble data and then encrypts them to protect privacy. Zhang et al. [17] proposed a system named Sedic which partitions MapReduce computing jobs in terms of the security labels of data they work on and then assigns the computation without sensitive data to a public cloud. The sensitivity of data is required to be labeled in advance to make the above approaches available. Ciriani et al. [18] proposed an approach that combines encryption and data fragmentation to achieve privacy protection for distributed data storage with encrypting only part of datasets. We follow this line, but integrate data anonymization and encryption together to fulfill cost-effective privacy preserving.

The importance of retaining intermediate datasets in cloud has been widely recognized [6], [7], but the research on privacy issues incurred by such datasets just commences. Davidson et al. [19], [20], [21] studied the privacy issues in workflow provenance, and proposed to achieve module privacy preserving and high utility of provenance information via carefully hiding a subset of intermediate data. This general idea is similar to ours, yet our research mainly focuses on data privacy preserving from an economical cost perspective while theirs concentrates majorly on functionality privacy of workflow modules rather than data privacy. Our research also differs from theirs in several aspects such as data hiding techniques, privacy quantification and cost models. But our approach can be complementarily used for selection of hidden data items in their research if economical cost is considered.

The PPDP research community has investigated extensively on privacy-preserving issues and made fruitful progress with a variety of privacy models and preserving methods [13]. Privacy principles such as \( k \)-anonymity [22] and \( l \)-diversity [23] are put forth to model and quantify privacy, yet most of them are only applied to one single dataset. Privacy principles for multiple datasets are also proposed, but they aim at specific scenarios such as continuous data publishing or sequential data releasing [13]. The research in [24], [25] exploits information theory to quantify the privacy via utilizing the maximum entropy principle [26]. The privacy quantification herein is based on the work in [24], [25]. Many anonymization techniques like generalization [12] have been proposed to preserve privacy, but these methods alone fail to solve the problem of preserving privacy for multiple datasets. Our approach integrates anonymization with encryption to achieve privacy preserving of multiple datasets. Moreover, we con-
sider the economical aspect of privacy preserving, adhering to the pay-as-you-go feature of cloud computing.

3 Motivating Example and Problem Analysis

Section 3.1 shows a motivating example to drive our research. The problem of reducing the privacy-preserving cost incurred by the storage of intermediate datasets is analyzed in Section 3.2.

3.1 Motivating Example

A motivating scenario is illustrated in Fig.1 where an online health service provider, e.g., Microsoft HealthVault [27], has moved data storage into cloud for economical benefits. Original datasets are encrypted for confidentiality. Data users like governments or research centres access or process part of original datasets after anonymization. Intermediate datasets generated during data access or process are retained for data reuse and cost saving. Two independently generated intermediate datasets (a) and (b) in Fig.1 are anonymized to satisfy 2-diversity, i.e., at least two individuals own the same quasi-identifier and each quasi-identifier corresponds to at least two sensitive values [23]. Knowing that a lady aged 25 living in 21400 (corresponding quasi-identifier is (214, female, young)) is in both datasets, an adversary can infer that this individual suffers from HIV with high confidence if (a) and (b) are collected together. Hiding (a) or (b) by encryption is a promising way to prevent such a privacy breach. Assume (a) and (b) are of the same size, the frequency of accessing (a) is 10 and that of (b) is 100. We hide (a) to preserve privacy because this can incur less expense than hiding (b).

In most real-world applications, a large number of intermediate datasets are involved. Hence, it is challenging to identify which datasets should be encrypted to ensure that privacy leakage requirements are satisfied while keeping the hiding expenses as low as possible.

3.2 Problem Analysis

3.2.1 Sensitive Intermediate Dataset Management

Similar to [6], data provenance is employed to manage intermediate datasets in our research. Provenance is commonly defined as the origin, source or history of derivation of some objects and data, which can be reckoned as intermediate datasets in our research. Provenance is exploited to build up the generation relationships of datasets [6].

We define several basic notations below. Let \( d_0 \) be a privacy-sensitive original dataset. We use \( D = \{d_1, d_2, \ldots, d_n\} \) to denote a group of intermediate datasets generated from \( d_0 \) where \( n \) is the number of intermediate datasets. Note that the notion of intermediate data herein refers to both intermediate and resultant data [6]. Directed Acyclic Graph (DAG) is exploited to capture the topological structure of generation relationships among these datasets.

**Definition 1** (Sensitive Intermediate Dataset Graph) A DAG representing the generation relationships of intermediate datasets \( D \) from \( d_0 \) is defined as a Sensitive Intermediate dataset Graph, denoted as SIG. Formally, \( SIG = (V, E) \), where \( V = \{d_0\} \cup D \), \( E \) is a set of directed edges. A directed edge \( (d_p, d_c) \) in \( E \) means that part or all of \( d_c \) is generated from \( d_p \), where \( d_p, d_c \in \{d_0\} \cup D \).

In particular, a SIG becomes a tree structure if each dataset in \( D \) is generated from only one parent dataset. Then, we have the following definition for this situation.

**Definition 2** (Sensitive Intermediate Dataset Tree) A SIG is defined as a Sensitive Intermediate dataset Tree (SIT) if it is a tree structure. The root of the tree is \( d_0 \).

A SIG or SIT not only represents the generation relationships of an original dataset and its intermediate datasets, but also captures the propagation of privacy-sensitive information among such datasets. Generally, the privacy-sensitive information in \( d_0 \) is scattered into its offspring datasets. Hence, a SIG or SIT can be employed to analyze privacy disclosure of multiple datasets. In this paper, we first present our approach on a SIT, and then extend it to a SIG with minor modifications in Section 5.

An intermediate dataset is assumed to have been anonymized to satisfy certain privacy requirements. However, putting multiple datasets together may still invoke a high risk of revealing privacy-sensitive information, resulting in violating the privacy requirements. Privacy leakage of a dataset \( d \) is denoted as \( PL(d) \), meaning the privacy-sensitive information obtained by an adversary after \( d \) is observed. The value of \( PL(d) \) can be deduced directly from \( d \), which is described in Section 4.1. Similarly, privacy leakage of multiple datasets in \( D \) is denoted as \( PL_m(D) \), meaning the privacy-sensitive information obtained by an adversary after all datasets in \( D \) are observed. It is challenging to acquire the exact value of \( PL_m(D) \) due to the inference channels among multiple datasets [24].

3.2.2 Privacy-Preserving Cost Problem

Privacy-preserving cost of intermediate datasets stems from frequent en/decryption with charged cloud services. Cloud service vendors have set up various pricing models to support the pay-as-you-go model, e.g., Amazon Web Services pricing model [29]. Practically, en/decryption needs computation power, data storage and other cloud services. To avoid pricing details and focus on the discussion of our core ideas, we combine the prices of various services required by en/decryption into one. This combined price is denoted as \( PR \). \( PR \) indicates the overhead of en/decryption per GB data per execution.
Similar to [6], an attribute vector is employed to frame several important properties of the dataset \(d_i\). The vector is denoted as \((S_i, Flag_i, f_i, P_i)\). The term \(S_i\) represents the size of \(d_i\). The term \(Flag_i\), a dichotomy label, signifies whether \(d_i\) is hidden. The term \(f_i\) indicates the frequency of accessing or processing \(d_i\). If \(d_i\) is labeled as hidden, it will be en/decrypted every time when accessed or processed. Thus, the larger \(f_i\) is, the more cost will be incurred if \(d_i\) is hidden. Usually, \(f_i\) is estimated from the data provenance. The term \(P_i\) is the privacy leakage through \(d_i\), and is computed by \(P_i(d_i)\).

Datasets in \(D\) can be divided into two sets. One is for encrypted datasets, denoted as \(D^\text{enc}\). The other is for un-encrypted datasets, denoted as \(D^\text{une}\). Then, the equations \(D^\text{enc} \cup D^\text{une} = D\) and \(D^\text{enc} \cap D^\text{une} = \emptyset\) hold. We define the pair \((D^\text{enc}, D^\text{une})\) as a global privacy-preserving solution.

The privacy-preserving cost incurred by a solution \((D^\text{enc}, D^\text{une})\) is denoted as \(CP\). With the notation framed above, the cost \(CP\) is defined as a global privacy-preserving solution. The privacy-preserving cost incurred by a solution \((D^\text{enc}, D^\text{une})\) is denoted as \(CP\). With the notation framed above, the cost \(CP\) is defined as follows.

\[
CP = \sum_{d_i \in D^\text{enc}} \sum_{f_i \in F_i} \text{PR} \cdot f_i.
\]

The approach proposed in [25] to compute the probability distribution \(P^*(S, Q)\) of \((s, q)\) when \(d^*\) is observed. More details can be found in Appendix A.1 (All appendices are included in the supplemental file). Then, the privacy quantification of dataset \(d^*\) can be fulfilled. Formally, \(PL\) is defined as:

\[
PL(d^*) = H(S, Q) - H^*(S, Q).
\]

4.1 Single Intermediate Dataset Privacy Representation

The privacy-sensitive information is essentially regarded as the association between sensitive data and individuals [13]. We denote an original sensitive dataset as \(d_o\); an anonymized intermediate dataset as \(d^*\); the set of sensitive data as \(SD\); and the set of quasi-identifiers as \(QI\). Quasi-identifiers, which represent the groups of anonymized data, can lead to privacy breach if they are too specific that only a small group of people are linked to them [13]. Let \(S\) denote a random variable ranging in \(SD\), and \(Q\) be a random variable ranging within \(QI\). Suppose \(s \in SD\) and \(q \in QI\). The joint possibility of an association \((s, q)\), denoted as \(p(S = s, Q = q)\) (abbr. \(p(s, q)\)), is the information that adversaries intend to recover [13]. When an adversary has observed \(d^*\) and a quasi-identifier \(q\), the conditional possibility \(p(S = s|Q = q)\) representing intrinsic privacy-sensitive information of an individual can be inferred. If \(p(S = s|Q = q) = \beta\) is deduced as a high value or even \(1.0\), the privacy of the individual with \(q\) will be awfully breached.

We employ the approach proposed in [25] to compute the probability distribution \(P^*(S, Q)\) of \((s, q)\) in \(d^*\) after observing \(d^*\). More details can be found in Appendix A.1 (All appendices are included in the supplemental file). Then, the privacy quantification of dataset \(d^*\) can be fulfilled. Formally, \(PL\) is defined as:

\[
PL(d^*) = H(S, Q) - H^*(S, Q).
\]

4.2 Joint Privacy Leakage of Multiple Intermediate Datasets

The value of the joint privacy leakage incurred by multiple datasets in \(D = \{d_1, d_2, \ldots, d_n\}, n \in N\), is defined by:

\[
PL_m(D) \triangleq H(S, Q) - H_0(S, Q).
\]

4.3 Privacy Representation and Privacy Leakage Upper-bound Constraint

It is fundamental to measure privacy leakage of anonymized datasets to quantitatively describe how much privacy is disclosed. Privacy quantification of a single dataset is stated in Section 4.1. We point out the challenge of privacy quantification of multiple datasets in Section 4.2 and then derive a privacy leakage upper-bound constraint correspondingly in Section 4.3.
acquiring $PL_m(D_{\text{une}})$ because our goal is to control the privacy disclosure caused by multiple datasets. A promising approach is to substitute the PLC with its sufficient conditions. Specifically, our approach is to replace the exact value of $PL_m(D_{\text{une}})$ with one of its upper bounds which can be calculated efficiently.

### 4.3 Upper-Bound Constraint of Joint Privacy Leakage

We attempt to derive an upper bound of $PL_m(D_{\text{une}})$ that can be easily computed. Intuitively, if an upper bound $B(PL_m(D_{\text{une}}))$ is found, a stronger privacy leakage constraint $B(PL_m(D_{\text{une}})) \leq \epsilon$ can be a sufficient condition of the PLC. Accordingly, $PL_m(D_{\text{une}})$ will never exceed the threshold $\epsilon$ if $B(PL_m(D_{\text{une}})) \leq \epsilon$ holds.

Let $d_u, d_r$ be two datasets whose privacy leakage are $PL_u(d_u)$ and $PL_r(d_r)$, respectively. The joint privacy leakage caused by them together is $PL_m((d_u, d_r))$. As information gain is never negative, $PL_m((d_u, d_r))$ is not less than neither $PL_u(d_u)$ nor $PL_r(d_r)$. Further, $PL_m((d_u, d_r))$ will not exceed the sum of $PL_u(d_u)$ and $PL_r(d_r)$, i.e., $PL_m((d_u, d_r)) \leq PL_u(d_u) + PL_r(d_r)$, where the equality holds if and only if the information provided by $d_u, d_r$ do not overlap. This property of joint privacy leakage can be extended to multiple unencrypted datasets in $D_{\text{une}}$:

$$PL_m(D_{\text{une}}) \leq \sum_{d \in D_{\text{une}}} PL_u(d_r).$$  (7)

Hence, the sum of privacy leakage of unencrypted datasets can be deemed as an upper bound of $PL_m(D_{\text{une}})$. Based on replacing the PLC with such an upper-bound constraint, we propose an approach to address the optimization problem in (3). See Section 5 for details.

### 5 Privacy Leakage Upper-Bound Constraint Based Approach for Privacy Preserving

We propose an upper-bound constraint based approach to select the necessary subset of intermediate datasets that needs to be encrypted for minimizing privacy-preserving cost. In Section 5.1, we specify relevant basic notations and elaborate two useful properties on a SIT. The privacy leakage upper-bound constraint is decomposed layer by layer in Section 5.2. A constrained optimization problem with the PLC is then transformed into a recursive form in Section 5.3. In Section 5.4, a heuristic algorithm is designed for our approach. We extend our approach to a SIG in Section 5.5.

#### 5.1 Basic Notations and Properties on SIT

Let $d_i \in D$ denote an intermediate dataset in a SIT. Its directly generated datasets constitute a set $CD(d_i) = \{d_{i1}, \ldots, d_{ir}\}$, where $d_{i1}, \ldots, d_{ir} \in D$. For any $d \in CD(d_i)$, $PL_d(d) \leq PL_r(d)$ because all information in $d$ is from $d_r$. Further, let $PD(d_r) = \{d_{i1}, \ldots, d_{ir}\}$ be all posterity datasets generated from $d_r$, where $d_{i1}, \ldots, d_{ir} \in D$. Similar to $CD(d_i)$, $PL_d(d) \leq PL_r(d)$ for any $d \in PD(d_r)$. A subtree of a SIT with $d_r \in D$ being its root is denoted as $SDT(d_r)$.

**Lemma 1.** If $d_r \in D_{\text{une}}$, then for any $d \in PD(d_r), d \in D_{\text{une}}$.

The proof of Lemma 1 can be found in Appendix B.1. Lemma 1 defines a useful property named Root Privacy Coverage (RPC) property. This property means that it is unnecessary to check the datasets in $PD(d_r)$ if $d_r$ unencrypted. Thus, $SDT(d_r)$ is somewhat equivalent to $d_r$ from the privacy-preserving perspective. Such a subtree is named as an unencrypted subtree, denoted as $UST$. $UST$s have the RPC property. Due to the RPC property, we only need to consider part of the intermediate datasets rather than all when identifying which datasets should be encrypted.

**Lemma 2.** Given a privacy-preserving solution $(D_{\text{enc}}, D_{\text{une}})$ with the minimal cost, a graph denoted as $EDG = (D_{\text{enc}} \cup \{d_\epsilon\}, E')$ must be a tree structure, where $E' \subseteq E$.

The proof of Lemma 2 can be found in Appendix B.2. This lemma defines a property named Encrypted Dataset Tree (EDT). The property means that all encrypted datasets in a SIT constitute a tree structure if the privacy-preserving cost is the minimal. Consequently, the datasets in $D_{\text{enc}}$ are always connected, regardless of which part of $D$ constitutes $D_{\text{enc}}$. Moreover, any subtree with its root connected to EDG is an UST because of the RPC property. According to the EDT property, the descendant intermediate datasets of a dataset have the possibility to be encrypted if this dataset is encrypted. Hence, it is feasible to construct a privacy-preserving solution with the minimal cost layer by layer from the root of a SIT to leaves.

Let $H$ be the height of a SIT and $L_i$ denote the layer where datasets with depth $i$ locate, $1 \leq i \leq H$. Let $D_i$ denote the set of datasets in $L_i$. The set of encrypted datasets in $L_i$ is denoted as $ED_i$. Further, the set of all children datasets of datasets in $ED_i$ is denoted as $CDE_{i+1}$, i.e., $CDE_{i+1} = \bigcup_{d \in ED_i} CD(d)$. Then, the set of unencrypted datasets in $CDE_i$ is denoted as $UD_i$. Note that $UD_i$ is not required to be equal to the set of unencrypted datasets in $L_i$.

Suppose that the encrypted datasets before the layer $L_i$ have been identified. Then, the encrypted datasets in $L_i$ are selected from $CDE_{i-1}$, rather than from all datasets in $D_i$. A possible choice of encryption is defined as a local encryption solution in $L_i$, denoted as $\pi_i = (ED_i, UD_i)$. The set of all potential local solutions in the layer $L_i$ is denoted as $A_i = \{\pi_{ij}\}$, where $j$ is the number of solutions. Further, a group of local encryption solutions, which are constructed layer by layer, constitute a global encryption solution, denoted as $\pi = \{\pi_{1j_1}, \ldots, \pi_{Hj_H}\}$, where $\pi_{ij} \in A_i, 1 \leq i \leq H, 1 \leq k \leq \Pi_{i=1}^H |A_i|$. $\pi$ can determine a global privacy-preserving solution $(D_{\text{enc}}, D_{\text{une}})$, $D_{\text{enc}} = \bigcup_{i=1}^H ED_i$.

#### 5.2 Recursive Privacy Leakage Constraint Decomposition

To satisfy the PLC, we decompose the PLC recursively into different layers in a SIT. Then, the problem stated in (3) can be addressed via tackling a series of small-scale optimization problems. Let the privacy leakage threshold required in the layer $L_i$ be $\epsilon_i, 1 \leq i \leq H$. The privacy leakage incurred by $UD_i$ in the solution $\pi_i$ can never be larger than $\epsilon_i$, i.e., $PL_m(UD_i) \leq \epsilon_i$. The threshold $\epsilon_i$ can be regarded as the privacy leakage threshold of the remainder part of a SIT after the layer $L_{i-1}$. In terms of the basic idea of our approach, the privacy leakage constraint $PL_m(UD_i) \leq \epsilon_i$ is substituted by one of its sufficient conditions. According to (7), the PLC can be substituted by a set
of privacy leakage constraints, named as PLC:

\[ \sum_{d \in UD_i} P_L(d) \leq \varepsilon_i, 1 \leq i \leq H. \]  
(8)

The above threshold \( \varepsilon_i, 1 \leq i \leq H \), is calculated by

\[
\begin{align*}
\varepsilon_i &= \varepsilon_{i-1} - \sum_{d \in UD_{i-1}} P_L(d), \\
\varepsilon_1 &= \varepsilon.
\end{align*}
\]  
(9)

A local encryption solution in the layer \( L_i \) is feasible if it satisfies the PLC in (8). The set of feasible solutions in \( L_i \) is denoted as \( A_i \equiv \{ \pi_{ij} | \pi_{ij} \in A_i \} \), where \( j \) is the number of feasible solutions. Similarly, a feasible global solution can be denoted as \( \pi^g_i \equiv \{ \pi_{ij1}, \ldots, \pi_{ijH} \} \), where \( \pi_{ij} \in A_i, 1 \leq i \leq H, 1 \leq k \leq \sum_{i \leq H} |A_i| \).

Given a feasible global solution \( \pi^g_i \) for a SIT, we compress the \( \pi^g_i \) into a 'compressed' tree layer by layer from \( L_1 \) to \( L_H \), denoted as \( CT(\pi^g_i) \), where \( H \) is the height of the EDG in the SIT. The construction of \( CT(\pi^g_i) \) is achieved via three steps. Firstly, the datasets in \( ED_i \) are 'compressed' into one encrypted node. According to the EDT property, these compressed nodes together with the original dataset appear to be a string with the length being \( H \). Secondly, all offspring datasets of the datasets in \( UD_i \) are omitted. This will not affect the privacy preserving in terms of the RPC property. Thirdly, the datasets in \( UD_i \) are compressed into one node. Note that the action 'compress' here is imaginary from a logical perspective for a demonstration purpose. Construction process of \( CT(\pi^g_i) \) is illustrated by Fig.2 where the first \( i \) layers have been built up according the above steps.

**Theorem 1.** Under the PLC\(_i\) constraint in (8), a construction process of a compressed tree on a SIT corresponds to a feasible global encryption solution \( \pi^g_i \), i.e., the privacy leakage \( P_L(D) \) with regard to \( \pi^g_i \) can be ensured beneath the given threshold \( \varepsilon \).

The proof of Theorem 1 can be found in Appendix B.3. Complying with the PLC\(_i\), we can obtain a feasible global encryption solution in the construction a compressed tree.

### 5.3 Minimum Privacy-Preserving Cost

Usually, more than one feasible global encryption solution exists under the PLC\(_i\) constraints, because there are many alternative local solutions in each layer. Further, each intermediate datasets has various size and frequency of usage, leading to different overall cost with different solutions. Therefore, it is desired to find a feasible solution with the minimum privacy-preserving cost under privacy leakage constraints. Note that the minimum solution mentioned herein is somewhat pseudo-minimum because an upper bound of joint privacy leakage is just an approximation of its exact value. But a solution can be exactly minimal in the sense of the PLC\(_i\) constraints. We derive the recursive minimal cost formula as follows.

The minimum cost for privacy preserving of the datasets after \( L_{i-1} \) under the privacy leakage threshold \( \varepsilon_i \) is represented as \( CM(\varepsilon_i), 1 \leq i \leq H \). Given a feasible local encryption solution \( \pi_i = (ED_i, UD_i) \) in \( L_i \), the cost incurred by the encrypted datasets in \( L_i \) is denoted as \( C_i(\pi_i) \):

\[
C_i(\pi_i) = \sum_{d \in ED_i} S_k \cdot PR \cdot f_k, 1 \leq i \leq H.
\]  
(10)

Then, \( CM(\varepsilon_i) \) is calculated by the recursive formula:

\[
CM_i(\varepsilon_i) = \min_{\pi_{ij}} \left[ \sum_{d \in ED_{ij}} (S_{ij} \cdot PR \cdot f_{ij}) + \right]
\]

\[
CM_{i+1}(\varepsilon_{i+1} - \sum_{d \in UD_{ij}} P_L(d)) \right]
\]

\[
CM_{H+1}(\varepsilon_{H+1}) = 0.
\]

As a result, \( CM_1(\varepsilon) \) is the minimum privacy-preserving cost required in the optimization problem specified by (3) and (8). The privacy-preserving solution \( D^{enc} D^{unv} \) can be determined during the process of acquiring \( CM_1(\varepsilon) \).

According to the specification of \( CM_1(\varepsilon) \), an optimal algorithm can be designed to identify the optimal privacy-preserving solution. The algorithm details can be found in Appendix C.1. However, this optimal algorithm suffers from poor efficiency due to its huge state-search space if a large number of intermediate datasets are involved. The time complexity is analyzed in Appendix C.2. As such, it is necessary to turn to heuristic algorithms for scenarios where a large number of intermediate datasets are involved, in order to obtain a near-optimal solution with higher efficiency than the optimal one.

### 5.4 Privacy-Preserving Cost Reducing Heuristic Algorithm

In this section, we design a heuristic algorithm to reduce privacy-preserving cost. In the state-search space for a SIT, a state node \( SN_i \) in the layer \( L_i \) herein refers to a vector of partial local solutions, i.e., \( SN_i \) corresponds to \( \{\pi_{ij1}, \ldots, \pi_{ijH}\} \), where \( \pi_{ij} \in A_k, 1 \leq k \leq i \). Note that the state-search tree generated according to A SIT is different from the SIT itself, but the height is the same. Appropriate heuristic information is quite vital to guide the search path to the goal state. The goal state in our algorithm is to find a near-optimal solution in a limited search space.

Heuristic values are obtained via heuristic functions. A heuristic function, denoted as \( f(SN_i) \), is defined to compute the heuristic value of \( SN_i \). Generally, \( f(SN_i) \) consists of two parts of heuristic information, i.e., \( f(SN_i) = g(SN_i) + h(SN_i) \), where the information \( g(SN_i) \) is gained from the start state to the current state node \( SN_i \) and the information \( h(SN_i) \) is estimated from the current state node to the goal state, respectively.

Intuitively, the heuristic function is expected to guide the algorithm to select the datasets with small cost but high privacy leakage to encrypt. Based on this, \( g(SN_i) \) is defined as \( g(SN_i) \leq C_{cur} / (\varepsilon - \varepsilon_{i+1}) \), where \( C_{cur} \) is the privacy-preserving cost that has been incurred so far, \( \varepsilon \) is the initial privacy leakage threshold and \( \varepsilon_{i+1} \) is the privacy leakage threshold for the layers after \( L_i \). Specifically, \( C_{cur} \) is calculated by \( C_{cur} = \sum_{d \in UD_{i+1} \rightarrow ED_i} (S_{ij} \cdot PR \cdot f_j) \). The smaller \( C_{cur} \) is, the smaller total privacy-preserving cost will be. Larger \( (\varepsilon - \varepsilon_{i+1}) \) means more datasets before \( L_{i+1} \) remain unencrypted in terms of the RPC property, i.e., more privacy-preserving expense can saved.
The value of \( h(SN) \) is defined as \( h(SN) = (\varepsilon_{i+1} \cdot C_{\text{des}} \cdot BF_{AVG})/PL_{AVG} \). Similar to the meaning of \((\varepsilon - \varepsilon_{i+1})\) in \( g(SN) \), smaller \( \varepsilon_{i+1} \) in \( h(SN) \) implies more datasets before \( L_{i+1} \) are kept unencrypted. If a dataset with smaller depth in a \( SIT \) is encrypted, more datasets are possibly unencrypted than that with larger depth, because the former possibly has more descendant datasets. For a state node \( SN_i \), the datasets in its corresponding \( ED_k \) are the roots of a variety of subtrees of the \( SIT \). These trees constitute a forest, denoted as \( F \). In \( h(SN_i) \), \( C_{\text{des}} \) represents the total cost of the datasets in \( F_\pi \), and is computed via \( C_{\text{des}} = \sum_{d_i \in ED_k} \sum_{d_j \in ED(d_i)} (S \cdot CR \cdot f_j) \). Potentially, the less \( C_{\text{des}} \) is, the fewer datasets in following layers will be encrypted. \( BF_{AVG} \) is the average branch factor of the forest \( F \), and can be computed by \( BF_{AVG} = N_F/N_i \), where \( N_F \) is the number of edges and \( N_i \) is the number of internal datasets in \( F \). Small \( BF_{AVG} \) means the search space for sequential layers will be smaller, so that we can find a near-optimal solution faster. The value of \( PL_{AVG} \) indicates the average privacy leakage of datasets in \( F \), calculated by \( PL_{AVG} = \sum_{d_i \in ED_k} \sum_{d_j \in ED(d_i)} PL(d_j) / N_i \). Heuristically, the algorithm prefers to encrypt the datasets which incur less cost but disclose more privacy-sensitive information. Thus, higher \( PL_{AVG} \) means more datasets in \( F \) should be encrypted to preserve privacy from a global perspective. Based on the above analysis, the heuristic value of the search node \( SN_i \) can be computed by the formula:

\[
f(SN) = C_{\text{cur}} / (\varepsilon - \varepsilon_{i+1}) + (\varepsilon_{i+1} \cdot C_{\text{des}} \cdot BF_{AVG})/PL_{AVG}. \quad (12)
\]

Based on this heuristic, we design a privacy-preserving cost reduction algorithm, denoted as \( H_{PPCR} \). The basic idea is that the algorithm iteratively selects a state node with the highest heuristic value and then extends its child state nodes until it reaches a goal state node. The privacy-preserving solution and corresponding cost are derived from the goal state.

Algorithm 1 specifies the details of our heuristic algorithm. A priority queue is exploited to keep state nodes. Only the qualified state nodes that are added to the priority queue, i.e., the corresponding partial global solutions are feasible. To avoid the size of the priority queue increase dramatically, the algorithm only retains the state nodes with top \( K \) highest heuristic values. When determining to add child search nodes in layer \( L_{i+1} \) into the priority queue, the algorithm generates a local encryption solution from \( CDE \) at first. The algorithm probably suffers from poor efficiency because it has to check all combinations of datasets in \( CDE \). To circumvent this situation, the algorithm ascendingly sorts the datasets in \( CDE_\pi \) according to the value \( C_k \cdot PL_s(d_k) \), where \( d_k \in CDE_\pi \) and \( C_k = S_k \cdot PR \cdot f_k \). If \( |CDE_\pi| \) is larger than a threshold \( M \), only the first \( M \) datasets in the sorted \( CDE_\pi \) will be examined while the remaining are set to be encrypted. Intuitively, datasets with higher privacy-preserving cost and lower privacy leakage are expected to remain unencrypted. The value \( C_k/PL_s(d_k) \) can help to guide the algorithm to find these datasets with a higher possibility. Hence, the algorithm is guided to approach the goal state in the state space as close as possible. Above all, in the light of heuristic information, the proposed algorithm can achieve a near-optimal solution practically. \( SOR \) and \( SELECT \) are two simple external functions as their names signify.

5.5 Extension to SIG

Although \( SITs \) can suit many applications, \( SIGs \) are also common, i.e., an intermediate dataset can originate from more than one parent dataset. An example of a \( SIG \) is illustrated in Fig.3 (a). The dataset \( d_0 \) is generated from \( d_2, d_3 \) and \( d_4 \), where \( d_4 \) is from \( d_2 \). Thus, it is possible that \( PL_\pi(d_0) \) is larger than \( PL_\pi(d_2) \) or \( PL_\pi(d_3) \), resulting in the failure of Lemma 1 and RPC property. As a result, our approach can not be directly applied to a \( SIG \). However,
we can adapt it to a SIG with minor modifications.

Let \( d_m \) denote a merging dataset that inherits data from more than one predecessor, e.g., \( d_1 \) in Fig.3. As only one root dataset is assumed to exist in a SIG, all paths from the root dataset \( d_0 \) to \( d_m \) must converge at one point beside \( d_m \) itself. Let \( d_s \) denote this source dataset, e.g., \( d_s = d_o \) in Fig.3. The inequality \( PL_s(d_m) \leq PL_s(d_s) \) holds because all privacy information of \( d_m \) comes from \( d_s \). Let \( PD_i(d_s) \) be the set of the offspring datasets of \( d_s \) in the layer \( L_i \), then \( PD_i(d_s) \subseteq PD(d_s) \). Datasets in \( PD_i(d_s) \) are split into \( ED_i \) and \( UD_i \) when determining which datasets are encrypted.

We discuss three cases where the graph structure can affect the applicability of our approach on a SIG. The first one is \( PD_i(d_s) \subseteq UD_i \), i.e., all datasets in \( PD_i(d_s) \) will keep unencrypted, e.g., \( \{d_1, d_2\} \subseteq UD_1 \) shown in Fig.3 (b). All ancestor datasets of \( d_s \) after the layer \( L_i \) will keep unencrypted according to the RPC property. So, the dataset \( d_m \) poses little influence on applying our algorithm to a SIG because \( d_s \) will not be considered in following steps. The second one is \( PD_i(d_s) \subseteq ED_i \), i.e., all datasets in \( PD_i(d_s) \) are encrypted. If \( d_m \) is a child of a dataset in \( PD_i(d_s) \), \( d_m \) is added to \( CDE_{i+1} \) for the next round. Assume the parent dataset is \( d_p \). Then, we delete the edges pointing to \( d_m \) from parents except \( d_p \), e.g., \( \langle d_1, d_2 \rangle \) is retained while \( \langle d_3, d_4 \rangle \) and \( \langle d_4, d_5 \rangle \) are deleted in Fig.3 (c). Logically, \( d_m \) can be deemed as a ‘compressed’ candidate dataset of several imaginary datasets in \( CDE_{i+1} \), which is similar to the construction of a compressed tree. The last one is that \( D_x \subseteq UD_i \) and \( D_y \subseteq ED_i \), where \( D_x \cap D_y = \emptyset \) and \( D_x \cup D_y = PD_i(d_s) \), i.e., part of datasets in \( PD_i(d_s) \) are encrypted while the remainder keep unencrypted. According to the RPC property, it is safe to expose part of privacy information in \( d_m \), where the part of privacy information is from \( D_x \). The edges which point to \( d_m \) from datasets in \( D_x \) and the offspring are deleted, e.g., \( \langle d_3, d_4 \rangle \) in Fig.3 (d). Further, the value of \( PL_s(d_m) \) is substituted by the maximum value of its direct parents who are datasets in \( D_x \) or the offspring of \( D_y \) if \( PL_s(d_m) \) is larger than the maximum.

To make the approach for a SIT available to a SIG as well, three minor modifications are required. The first one is to identify all merging datasets. The second one is to adjust the SIG according to the third case discussed above if \( UD_s \neq \emptyset \) after we get a local solution \( \pi = \langle ED_s, UD_s \rangle \). The third one is to label the datasets that have been processed. In this way, it is unnecessary to explicitly delete edges discussed in the second case.

6 EVALUATION

6.1 Overall Comparison

Encrypting all datasets for privacy preserving is widely adopted in existing research [8], [9], [10]. This category of approach is denoted as ALL_ENC. The privacy-preserving cost of ALL_ENC is denoted as \( C_{ALL} \). For datasets in set \( D \) of a SIT, \( C_{ALL} \) is computed by:

\[
C_{ALL} = \sum_{\beta \in D} S_\beta \cdot PR \cdot f_k.
\]  

Compared with ALL_ENC, our approach only selects necessary intermediate datasets to encrypt while keeping others unencrypted. Intuitively, the cost of our approach is lower than the ALL_ENC approach. To further evaluate how our approach can prevail significantly against the ALL_ENC approach in saving expense, we conduct experiments on real-world datasets and then extend the experiments to datasets of large amounts. To facilitate the comparison, the privacy-preserving cost of H_PPCR is denoted as \( C_{HEU} \). Given a global privacy-preserving solution with the encrypted set \( ED_i \), \( C_{HEU} \) is computed by:

\[
C_{HEU} = \sum_{\beta \in ED} S_\beta \cdot PR \cdot f_k.
\]  

The difference between \( C_{ALL} \) and \( C_{HEU} \) is denoted as \( C_{AV} \). We run both approaches on SITs with different \( \epsilon \) values. According to the analysis in Section 4.2, threshold \( \epsilon \) ranges in the interval \( [E_{min}, E_{max}] \), where \( E_{max} = \log(\langle QI \rangle \cdot |SD|) \) and \( E_{min} = \max_{0 \leq sn \leq \beta} \{PL_s(d_s)\} \). For convenience, we define privacy leakage degree, denoted as \( \epsilon_{PL} = (\epsilon - E_{min})/(E_{max} - E_{min}) \), to indicate privacy leakage incurred by unencrypted intermediate datasets.

6.2 Experiment Evaluation

6.2.1 Experiment Environment

U-Cloud is a cloud computing environment at the University of Technology Sydney (UTS). The system overview of U-Cloud is depicted in Fig.4. The computing facilities of this system are located among several labs at UTS. On top of hardware and Linux operating system, we install KVM virtualization software [30] which virtualizes the infrastructure and provides unified computing and storage resources. To create virtualized data centres, we install OpenStack open source cloud environment [31] for global management, resource scheduling and interaction with users. Further, Hadoop [32] is installed based on the cloud built via OpenStack to facilitate massive data processing. Our experiments are conducted in this cloud environment.

6.2.2 Experiment Process

To demonstrate the effectiveness and scalability of our approach, we run H_PPCR and ALL_ENC on real-world datasets and extensive datasets. We first test them on real-world datasets with \( \epsilon_d \) varying in \([0.05, 0.9] \), then on extensive intermediate datasets with the number ranging in \([50, 1000]\) under certain \( \epsilon_d \). For each group of experiments, we repeat H_PPCR and ALL_ENC 50 times. The mean of cost in each experiment group are regarded as the representative. The margin of error with 95% confidence is also measured and shown in the results.

We first conduct our experiments on the Adult dataset.
which a commonly-used dataset in the privacy research community. Intermediate datasets are generated from the original dataset, and anonymized by the algorithm proposed in [12]. The privacy leakage of each dataset is quantified as formulated in Section 4.1. The details about these datasets are described in Appendix D.1.

Further, we extend experiments to intermediate datasets of large amounts. STTs are generated via a random spanning tree algorithm [34]. The values of data size and usage frequencies are randomly produced in the interval [10, 100] according to the uniform distribution. Without influencing the trend analysis, we set the price $PR$ to be 0.001. The privacy leakage values are generated randomly in an interval $[0, E_{up}]$, where $E_{up}$ is value that is by no means larger than the maximum entropy $E_{max}$ of original datasets. In our experiments $E_{max}$ is set to 18.20, meaning $|QI| \cdot |SD| = 300,000$. The thresholds $M$ and $K$ in $H_{PPCR}$ are set to 20 and 10,000, respectively.

### 6.2.3 Experiment Results and Analysis

The experimental result on real-world datasets is depicted in Fig.5, from which we can see that $C_{HEU}$ is much lower than $C_{ALL}$ with different privacy leakage degree. Even the smallest cost saving of $C_{HEU}$ over $C_{ALL}$ at the left side of Fig.5 is more than 40%. Therefore, Fig. 5 demonstrates our approach $H_{PPCR}$ can save privacy-preserving cost significantly over ALL-ENC approach. Further, we can see that the difference $C_{SAV}$ between $C_{ALL}$ and $C_{HEU}$ increases when the privacy leakage degree increases. This is because looser privacy leakage restraints imply more datasets can remain unencrypted.

With Fig.5 where we reason about the difference between $C_{HEU}$ and $C_{ALL}$ with different privacy leakage degree, Fig.6 illustrates how the difference changes with different numbers of extensive datasets while $e_d$ is certain. In most real-world cases, data owners would like the data privacy leakage to be much low. Thus, we select four low privacy leakage degrees of 0.01, 0.05, 0.1 and 0.2 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 $C_{HEU}$ against $C_{ALL}$. Similarly, we set the number of $e_d$ values as four. As Fig.5 has shown the trend with different $e_d$ values, we do not have to try all values. At the same time, we would like to informatively conduct the experiments. Hence, we select four values. Interested readers can try 3, 5 or other number of values. The conclusions will be similar.

From Fig.6, we can see that both $C_{ALL}$ and $C_{HEU}$ go up when the number of intermediate datasets is getting larger. That is, the larger the number of intermediate datasets is, the more privacy-preserving cost will be incurred. $C_{ALL}$ increases notably because it is proportional to the number of intermediate datasets. Given $e_d$, $C_{HEU}$ also increases with the increase of the number of datasets because more datasets are required to be encrypted. Moreover, Fig.6 also shows that $C_{HEU}$ drops when the privacy leakage degree becomes larger while $C_{ALL}$ keeps invariable. This tendency complies with that shown in Fig.5.

Most importantly, we can see from Fig.6 that the difference $C_{SAV}$ between $C_{ALL}$ and $C_{HEU}$ becomes bigger and bigger when the number of intermediate datasets increases. That is, more expense can be reduced when the number of datasets becomes larger. This trend is the result of the dramatic rise in $C_{ALL}$ and relatively slower increase in $C_{HEU}$ when the number of datasets is getting larger. In the context of Big Data, the number and sizes of datasets and their intermediate datasets are quite large in cloud. Thus, this trend means our approach can reduce the privacy-preserving cost significantly in real-world scenarios.

As a conclusion, both the experimental results demonstrate that privacy-preserving cost intermediate datasets can be saved significantly through our approach over existing ones where all datasets are encrypted.

### 7 Conclusions and Future Work

In this paper, we have proposed an approach that identifies which part of intermediate datasets needs to be encrypted while the rest does not, in order to save the privacy-preserving cost. A tree structure has been modeled from the generation relationships of intermediate datasets to analyze privacy propagation among datasets. We have modeled the problem of saving privacy-preserving cost as a constrained optimization problem which is addressed by decomposing the privacy leakage constraints. A practical heuristic algorithm has been designed accordingly. Evaluation results on real-world datasets and larger extensive datasets have demonstrated the cost of preserving privacy in cloud can be reduced significantly with our approach over existing ones where all datasets are encrypted.
In accordance with various data and computation intensive applications on cloud, intermediate dataset management is becoming an important research area. Privacy preserving for intermediate datasets is one of important yet challenging research issues, and needs intensive investigation. With the contributions of this paper, we are planning to further investigate privacy-aware efficient scheduling of intermediate datasets in cloud by taking privacy preserving as a metric together with other metrics such as storage and computation. Optimized balanced scheduling strategies are expected to be developed towards overall highly efficient privacy aware dataset scheduling.

REFERENCES


Xuyun Zhang He is currently working towards the PhD degree at University of Technology, Sydney. His research interests include cloud computing, privacy and security, MapReduce and OpenStack.

Chang Liu He is currently a PhD student at University of Technology Sydney, Australia. His research interests include cloud computing, resource management, cryptography and data security.

Surya Nepal Dr Surya Nepal is a Principal Research Scientist at CSIRO ICT Centre, Australia. His main research interest includes security, privacy and trust in emerging science areas such as Service Oriented Architectures (SOA), Web Services, Cloud Computing and Social Computing. He has several journal and conference papers in these areas to his credit.

Suraj Pandey Dr Suraj Pandey holds a research scientist position at IBM Research Australia. His research spans many areas of high performance computing, including data intensive applications, workflow scheduling, resource provisioning, and accelerating scientific applications, evident from his publications in ERA A* and ERA A ranked journals and conferences.

Jinjun Chen Dr Jinjun Chen is an Associate Professor and the Director of Lab of Cloud Computing and Distributed Systems at University of Technology Sydney, Australia. His research interests include cloud computing, social computing, green computing, service computing, e-science/e-research, workflow management. His research results have been published in more than 100 papers in high quality journals and conferences, including TOSEM, TSE, and ICSE.