Fusion:
Privacy-preserving Distributed Protocol for High-Dimensional Data Mashup

Gaby G. Dagher
CSE, Concordia University
Montréal, Québec, Canada
dagher@encs.concordia.ca

Farkhund Iqbal
CTI, Zayed University
Abu Dhabi, United Arab Emirates
Farkhund.Iqbal@zu.ac.ae

Mahtab Arafati
CIISE, Concordia University
Montréal, Québec, Canada
ma_arafa@ciise.concordia.ca

Benjamin C. M. Fung
SIS, McGill University
Montréal, Québec, Canada
ben.fung@mcgill.ca

Abstract—In the last decade, several approaches concerning private data release for data mining have been proposed. Data mashup, on the other hand, has recently emerged as a mechanism for integrating data from several data providers. Fusing both techniques to generate mashup data in a distributed environment while providing privacy and utility guarantees on the output involves several challenges. That is, how to ensure that no unnecessary information is leaked to the other parties during the mashup process, how to ensure the mashup data is protected against certain privacy threats, and how to handle the high-dimensional nature of the mashup data while guaranteeing high data utility.

In this paper, we present Fusion, a privacy-preserving multi-party protocol for data mashup with guaranteed LKC-privacy for the purpose of data mining. Experiments on real-life data demonstrate that the anonymous mashup data provide better data utility, the approach can handle high dimensional data, and it is scalable with respect to the data size.

Keywords—mashup; privacy; anonymization; data mining;

I. INTRODUCTION

As the amount of data available from wide range of domains has increased tremendously in recent years, the demand for data sharing and integration has also risen. The mashup of related data from different sources enables businesses, organizations and government agencies to perform better data analysis and make better decisions. In this paper, we present Fusion, a protocol that enables multiple data providers to engage in a privacy-preserving mashup process to generate an anonymous mashup data with high information utility for data mining tasks such as classification analysis. Throughout the mashup process, a score function needs to be computed between the parties to guide the process. Therefore, we propose a secure protocol for evaluating the score function in a distributed setting. Figure 1 presents an example of a distributed environment for privacy-preserving data mashup. The challenges of mashing-up data from different data providers in a privacy preserving manner are summarized as follows.

A major challenge is privacy concerns. Data providers are often reluctant to share their data due to privacy concerns. We distinguish between two types of concerns. The first is to

Figure 1: Privacy-preserving distributed data mashup

allow data providers to evaluate functions on the collective data while ensuring that no party learns more information about other parties’ data, other than what is revealed in the final mashup data.

Example I.1. Consider the data in Table I, where three data providers: $P_1$, $P_2$ and $P_3$, owns different set of attributes about the same individuals, and $P_2$ owns the Class attribute. Assume that the parties are building a classifier and need to compute information gain $I$ for each attribute. $P_2$ can directly compute the information for attribute Sex since it knows the class values. However, $P_1$ and $P_3$ should be able to compute the information for each of their attributes while the class values remain private (only known to $P_2$).

The second concern is to ensure the final mashup data is anonymized such that potential linkage attacks are thwarted. The adversary can perform two types of linkage attacks: record linkage, where an individual can be linked to a record if the record is very specific, and attribute linkage, where a frequent sensitive value can be inferred about an individual.

Example I.2. In Table I, if the adversary knows (44, 12th, Female) about an individual, then the adversary can link the individual to record #7 and sensitive value $s_2$. On the other hand, if the adversary knows (Bachelor, Male), then he infers sensitive value $s_2$ about
TABLE I: Raw data owned by providers \( P_1 \), \( P_2 \) and \( P_3 \)

<table>
<thead>
<tr>
<th>UID</th>
<th>Data Provider ( P_1 )</th>
<th>Data Provider ( P_2 )</th>
<th>Data Provider ( P_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>54</td>
<td>7th</td>
<td>Bachelor</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>Bachelor</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>39</td>
<td>6th</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>67</td>
<td>Bachelor</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>Bachelor</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>Doctorate</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>44</td>
<td>12th</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>29</td>
<td>Bachelor</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>53</td>
<td>Master</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>46</td>
<td>Master</td>
<td>Yes</td>
</tr>
</tbody>
</table>

the individual with 67% probability.

Another major challenge is data utility. The anonymous mashup data should preserve as much information as possible for the targeted data mining tasks such as classification or clustering analysis. However, since each data provider can own several attributes, each of which is considered a dimension, the resulting mashup data is usually high-dimensional. Many anonymization approaches, such as \( \ell \)-anonymity [2], generate useless anonymous data when applied on high-dimensional data due to the curse of high dimensionality [3]. Therefore, choosing the appropriate anonymization approach is critical for maintaining high data quality.

The contributions of this paper can be summarized as follows:

**Contribution #1.** We present a secure protocol for the distributed evaluation of an information gain-based score function, and show that the protocol is privacy-preserving.

**Contribution #2.** We present a multi-party protocol that applies a hierarchial approach to anonymize high-dimensional data and generate mashup data satisfying LKC-privacy [4].

**Contribution #3.** We performed experimental evaluation on real-life data in different distributed settings. Extensive experimental results suggest that the mashup data provides better data utility, and our approach is scalable w.r.t. the number of records as well as the number of attributes.

II. BUILDING BLOCKS

A. Privacy Model

In recent years, several privacy models [5] [6] have been proposed to prevent linkage attacks against published data. In the problem studied in this paper, each data provider can own many attributes and the result is often a high-dimensional mashup table. Therefore, we choose LKC-privacy [4], a privacy model that was originally designed for preventing linkage attacks on high-dimensional data, i.e., data with a large number of attributes. The authors in [4] have shown that LKC-privacy yields better utility for data mining in the anonymized data in comparison to the traditional privacy models.

Let \( T \) be an attribute table in the form of \( T = (A_1, \ldots, A_m, Sen) \), where each record contains information about a unique individual. \( QID = \{A_1, \ldots, A_m\} \) is a set of quasi-identifier attributes, such as sex and age, that may identify an individual if some combinations of \( QID \) values are specific enough. \( Sen \) is a sensitive attribute that contains some sensitive information about the individuals in the table, such as salaries or diseases. We assume that the adversary looking to identify the record or sensitive value about an individual in \( T \) has a limited prior knowledge \( qid \). More specifically, the adversary knows values from at most \( L \) attributes in \( QID \), where \( |qid| \leq L \). Given \( qid \), the adversary can identify the set of records in \( T \) that satisfy \( qid \), denoted by \( T[qid] \), and launch two privacy attacks:

**Record Linkage Attack.** If the number of records \( |T[qid]| \) is small, the adversary can distinguish the individual’s record, and consequently, the sensitive value. For example, if \( qid = (44, 12th, \text{Female}) \) in Table I, then \( T[qid] = \{UID\#7\} \), \( |T[qid]| = 1 \) and \( Sen = s_2 \).

**Attribute Linkage Attack.** If the number of records \( |T[qid]| \) is large, the adversary can still infer the sensitive value with confidence \( Pr(s(qid)|s_{uid}) = \frac{|T[qid]\cap\{s\}|}{|T[qid]|} \), where \( T[qid]\cap\{s\} \) denotes the set of records containing both \( qid \) and \( s \). For example, if \( qid = \langle \text{Bachelor, Male} \rangle \), then \( T[qid \cap s_2] = \{UID\#5, 8\} \) and \( T[qid] = \{UID\#2, 5, 8\} \). Accordingly, \( Pr(s_2|qid) = \frac{2}{3} = 67\% \).

To prevent such privacy attacks, LKC-privacy requires that in the anonymized table, for every \( qid : |qid| \leq L \), \( qid \) is shared by at least \( K \) records, and the percentage of each sensitive value in every group cannot exceed a certain value \( C \). LKC-privacy guarantees that the probability of a successful record linkage to be \( \leq 1/K \) and the probability of a successful attribute linkage to be \( \leq C \). The following is the formal definition.

**Definition II.1 (LKC-Privacy [4]).** Let \( L \) be the maximum number of \( QID \) values of the adversary’s background knowledge on any individual in a data table \( T \). Let \( S \) be a set of sensitive values. A data table \( T \) satisfies LKC-privacy if, and only if, for any \( qid \) with \( |qid| \leq L \),

1) \( |T[qid]| \geq K \), where \( K > 0 \) is a minimum anonymity threshold, and
2) \( \forall s \subseteq S \), the probability \( P(s|qid) \leq C \), where \( 0 < C \leq 1 \) is a maximum confidence threshold.

LKC-privacy is a generalized privacy model of \( K \)-anonymity [2], confidence bounding [7], and \( \ell \)-diversity [3], which gives the flexibility to the data providers to employ these traditional privacy models.

B. Cryptographic Primitives

**Encryption Scheme.** Our protocol requires an additively homomorphic encryption scheme that allows ciphertexts to be re-randomized without private information. It must also admit distributed key generation (DKG) and distributed decryption, enabling participants to use key shares to perform a decryption operation. The best candidate is a variation of ElGamal [9] called Exponential ElGamal [10]: it is
fast when implemented over elliptic curves, distributed key generation is straightforward, and decryption is feasible for our plaintext space. For brevity, we denote the encryption of a message \( m \) as \([m]_n\).

**Distributed Exponential ElGamal Decryption \([\text{DEG}]\).** Given ElGamal ciphertext \((\alpha, \beta)\), where the secret key \( x \in \mathbb{Z}_p \) is shared between \( n \) parties according to \((k, n)\)-threshold such that \( k \leq n \), each participant \( P_i \) from any group of \( k \) participants \( P_1, \ldots, P_k \) publishes \( \beta^{\alpha x_i} \), where \( x_i \) is a private key share of \( P_i \). The plaintext can then be derived by computing: \( \alpha / \prod_{i=1}^{n} \beta^{\alpha x_i} \). For the purpose of this paper, we assume that \( x \) is shared according to \((2, n)\)-threshold.

### III. Problem Formulation

In this section we formally define the research problem. First, we present an overview of the problem of privacy-preserving data mashup with privacy guarantees on output data in Section III-A. Next, we introduce taxonomy trees in Section III-B and then we describe the utility measures in Section III-C. We then describe the threat model in Section III-D. Finally, we present the problem statement in Section III-E.

#### A. Problem Overview

This paper addresses the problem of integrating distributed person-specific data while preserving both privacy and information utility on the final mashup data. Each party involved in the protocol represents a data provider who is interested in integrating its data with other providers’ data without leaking any unnecessary information. The mashup data is then released to the public for data mining.

We assume that the data being integrated is in the form of a relational table that is vertically partitioned into subtables, each of which is owned by one data provider. Let \( P_1, P_2, \ldots, P_p \) be the group of data providers participating in the protocol. Each party \( P_i \) for \( 1 \leq i \leq p \) owns a table in the form of \( T_i = (UID, EID_i, QID_i, Sen_i, Class) \). \( UID \) is a system-generated unique identifier of an individual, and is shared by all data providers. \( EID_i \) is a set of explicit identifiers containing information that can explicitly identify an individual, and should be removed before the protocol is executed. \( QID_i \) is a set of quasi-identifier attributes, each of which is either categorical or numerical. \( QID_i \) attributes cannot be removed as they are useful for the data mining task, and each attribute can be shared by any number of data providers. We denote by \( \bigcup QID = \bigcup_{i=1}^{p} QID_i \) the union of all quasi-identifier attributes owned by the parties. \( Sen_i \) is a set of sensitive attributes containing some sensitive information about the individuals, and it is shared between all data providers. \( Class \) is a categorical target class attribute for classification analysis. We assume that only one data provider owns (have knowledge to) this attribute. The result of the integration is an anonymous mashup data that satisfies an \( LKC \)-privacy requirements agreed upon by all the parties. Note that increasing the anonymity threshold \( K \), increasing the prior knowledge threshold \( L \), or decreasing the confidence threshold \( C \) imposes a higher level of privacy protection, which in general would result in a lower information utility (data quality) of the anonymized data.

#### B. Taxonomy Trees \([2]\)

A taxonomy tree of an attribute \( A \), denoted by \( T^A \), is a context-specific hierarchical structure that classifies the items in the attribute’s domain. In a taxonomy tree for a categorical attribute, the leaf nodes are the domain items, and the non-leaf nodes represent more generalized concepts of their children. On the other hand, in a taxonomy tree for a numerical attribute, the root node represents the full numerical range of the attribute, and the children nodes represent an optimal split of the parent range. We assume for each attribute \( A \in \bigcup QID \), a context-specific taxonomy tree is defined. Figure 2 presents taxonomy trees for the \( \bigcup QID \) attributes in Table II:

<table>
<thead>
<tr>
<th>UID</th>
<th>Age</th>
<th>Education</th>
<th>Class</th>
<th>Sex</th>
<th>Sen</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>[1-99]</td>
<td>University</td>
<td>No</td>
<td>Any</td>
<td>Sen</td>
<td>[10K-70K]</td>
</tr>
<tr>
<td>3</td>
<td>[1-99]</td>
<td>Secondary</td>
<td>No</td>
<td>Any</td>
<td>Sen</td>
<td>[10K-70K]</td>
</tr>
<tr>
<td>4</td>
<td>[1-99]</td>
<td>University</td>
<td>Yes</td>
<td>Any</td>
<td>Sen</td>
<td>[10K-70K]</td>
</tr>
<tr>
<td>5</td>
<td>[1-99]</td>
<td>University</td>
<td>Yes</td>
<td>Any</td>
<td>Sen</td>
<td>[70K-125K]</td>
</tr>
<tr>
<td>6</td>
<td>[1-99]</td>
<td>University</td>
<td>No</td>
<td>Sen</td>
<td>Sen</td>
<td>[70K-125K]</td>
</tr>
<tr>
<td>7</td>
<td>[1-99]</td>
<td>Secondary</td>
<td>No</td>
<td>Any</td>
<td>Sen</td>
<td>[10K-70K]</td>
</tr>
<tr>
<td>8</td>
<td>[1-99]</td>
<td>University</td>
<td>Yes</td>
<td>Any</td>
<td>Sen</td>
<td>[70K-125K]</td>
</tr>
<tr>
<td>10</td>
<td>[1-99]</td>
<td>University</td>
<td>Yes</td>
<td>Any</td>
<td>Sen</td>
<td>[70K-125K]</td>
</tr>
</tbody>
</table>

#### C. Data Utility Measures

Our idea for generating anonymous mashup data is to anonymize the raw data by performing a sequence of specializations, starting from the topmost general state. To specialize a value \( v \), denoted by \( v \rightarrow \text{child}(v) \), we replace \( v \) by its children values \( \text{child}(v) \). The specialization process can be viewed as pushing the cut of each taxonomy tree downwards. A \( \text{Cut}^A \) of a taxonomy tree \( T^A \) contains exactly one value on each root-to-leaf path. We denote by \( \bigcup \text{Cut} = \bigcup_{\text{A} \in \bigcup QID} \text{Cut}^A \) the union of all cuts. In Figure 2 the dashed curve represents \( \bigcup \text{Cut} \) (also referred to as solution cut) of the \( LKC \) anonymous Table III. The specialization starts from the topmost cut and pushes down the cut iteratively by specializing a value in the current cut until no further specialization that satisfies the \( LKC \)-privacy requirements is possible. We define two utility measures to help us determine at each level the best value \( v \) for specialization.

1. **UM1: Classification Analysis**: We utilize information gain \([1]\) to measure the quality of specialization on a value \( v \) for the purpose of classification analysis. Construction \([1]\) illustrates how the score function \( \text{Score}(v) \) can be computed.
Figure 2: Taxonomy trees: T\(^\text{Age}\), T\(^\text{Education}\), T\(^\text{Sex}\) and T\(^\text{Salary}\) for \(\cup\text{QID}\) attributes in Table II and the union cut \(\cup\text{Cut}\) of the LKC anonymous Table III.

### Classification Analysis

Let \(T[v]\) be the set of records in \(T\) that are generalized to \(v\). The score for a specialization on a value \(v\) can be determined as follows:

1. Compute the entropy of \(T[v]\):
   \[
   E(T[v]) = - \sum_{c \in \text{Class}} \frac{|T[v] \land c|}{|T[v]|} \log_2 \left( \frac{|T[v] \land c|}{|T[v]|} \right) \tag{1}
   \]
   where \(|T[v] \land c|\) denotes the records in \(T[v]\) with class value \(c\).

2. Given that \(|T[v]| = \sum_{c} |T[c]|\), where \(c \in \text{child}(v)\), compute the entropy of \(T[c]\) for each \(c \in \text{child}(v)\):
   \[
   E(T[c]) = - \sum_{c \in \text{Class}} \frac{|T[c] \land v|}{|T[c]|} \log_2 \left( \frac{|T[c] \land v|}{|T[c]|} \right) \tag{2}
   \]

3. Compute the score of specializing \(T[v]\) on value \(v\):
   \[
   \text{Score}(v) = E(T[v]) - \sum_{c \in \text{child}(v)} \frac{|T[c]|}{|T[v]|} E(T[c]) \tag{3}
   \]

#### Construction 1: Utility Measure for Classification Analysis

At any level during the specialization process, the score of every valid attribute for specialization can be computed according to Construction [1] and then the value with the highest score is chosen to perform the actual specialization.

#### UM2: General Analysis
We utilize discernibility cost \([13]\) to measure the quality of specialization on a value \(v\) when the data mining task is unknown. The discernibility cost penalizes each record that is indistinguishable from the rest of the records in a group, and the penalty cost equals the size of the group. That is, each record in an equivalence class \(qid\) is penalized by \(|T[qid]|\), and the total penalty cost of the class is \(|T[qid]|^2\). Hence, the score for specialization on \(v\) considers all \(qid\) combinations that contain \(v\):

\[
\text{Score}(v) = \sum_{qid} |T[qid]|^2 : v \in qid \tag{4}
\]

Similar to the utility measure for classification analysis, we choose the specialization that yields the highest score.

### D. Threat Model

Fusion is secure in the semi-honest adversarial model \([14]\), where each party is trusted to follow the protocol, but during the execution, tries to infer information from other parties. All parties are assumed to be non-colluding and their computational powers are polynomially bounded.

### E. Problem Statement

Let \(P_1, \ldots, P_p\), where \(p > 2\), be a group of data providers respectively owning vertically-partitioned data tables \(T_1, \ldots, T_p\), where any quasi-identifier attribute can be shared by any number of data providers, all sensitive attributes are shared between all data providers and the Class attribute is only owned by one provider. Let \(L, K\) and \(C\) be the prior knowledge threshold, the anonymity threshold and the confidence threshold values agreed upon by all parties. The objective of our work is to propose a protocol for generating an anonymous mashup data table \(\hat{T}\) such that (1) \(\hat{T}\) satisfies LKC-privacy requirements, (2) no party learns unnecessary information about other parties’ data than what is in \(\hat{T}\) (which is LKC-private), (3) \(\hat{T}\) preserves an effective level of information utility for data mining purposes, and (4) the protocol is scalable with respect to high-dimensional data.

### IV. FUSION: DISTRIBUTED PROTOCOL FOR HIGH-DIMENSIONAL DATA MASHUP

#### A. Solution Overview

This paper introduces a multi-party protocol, named Fusion, for integrating distributed person-specific data while preserving both privacy and information utility on the final mashup data. The main idea is to anonymize the raw data by generalizing all raw data records to a topmost general state, and then perform a sequence of specializations such that in each specialization step we choose the specialization with the highest score to maintain the highest possible information utility. The specialization process continues until there is no more specialization that satisfies the LKC-privacy requirements. Our solution consists of two main protocols:

- **Protocol 1**: Distributed Specialization Score (DSS).
- **Protocol 2**: Hierarchical High-dimensional Data Mashup (HHD). This protocol presents a distributed hierarchical approach for integrating high dimensional data...
Distributed Specialization Score (DSS)

Let \(\mathcal{P}^{cls}\) be the party that owns the Class attribute. We assume that each party already received from \(\mathcal{P}^{cls}\) the class values encrypted under the data providers’s distributed public key, as shown in Table III.

1) Following Construction 1, \(\mathcal{P}^{cls}\) directly computes the score for each valid specialization of every attribute in \(\cup QID\) it owns.
2) For each valid specialization \(v \rightarrow child(v)\) of every attribute in \(\cup QID\) owned by \(\mathcal{P}_k : 1 \leq k \leq p\) (except \(\mathcal{P}^{cls}\)), \(\mathcal{P}_k\) does the following:
   a) Choose a party randomly and then together execute Sub-Protocol 1 to compute \([E(T[v])]\).
   b) For each \(c \in child(v)\), it randomly chooses another party and then together execute Sub-Protocol 1 to compute \([E(T[c])]\).
   c) Homomorphically compute the score of specialization over \(v\) as follows:
      \[
      [\text{Score}(v)] = 10^d \times \left( [E(T[v])] - \sum_{c \in child(v)} \left[ \frac{|T[c]|}{|T[v]|} \times [E(T[c])] \right] \right) 
      \]
      where \(d\) is the number of decimal places (precision) agreed upon by all parties.
   d) Request one of the parties to partially decrypt \([\text{Score}(v)]\), and then uses its own share of the secret key to fully decrypt the ciphertext and obtain \(\text{Score}(v)\).
3) All parties engage in a secure circuit evaluation process using Yao's Protocol to determine which party has the specialization value with the highest score.

Protocol 1: Distributed Specialization Score

**Input:** Principle party \(\mathcal{P}_i\), assisting party \(\mathcal{P}_j\), potential specialization value \(x\)

**Output:** Encrypted entropy of \(T(x)\)

1) \(\mathcal{P}_i\) chooses random integer \(r\) from \(\mathbb{Z}_p^*\), and then for each ciphertext \([cls]\in T[x], [\text{Class}]\), where \(T[v], [\text{Class}]\) denotes the set of ciphertexts from the encrypted \([\text{Class}]\) attribute that corresponds to the group of records generalized to \(v\), it performs the following:
   a) Blind \([cls]\) by exponentiating in \(r\): \([cls]' = [cls]^r\).
   b) Partially decrypt \([cls]'\) using its own share of the secret key.
2) \(\mathcal{P}_i\) sends the set of partially decrypted ciphertexts to \(\mathcal{P}_j\) through a secure channel.
3) Using its own share of the secret key, \(\mathcal{P}_j\) decrypts the set of ciphertexts and obtains a set of blinded class values.
4) \(\mathcal{P}_j\) computes the entropy \(E(T[x])\) according to Equation 1 from Construction 1.
5) \(\mathcal{P}_j\) computes the integer value \([E(T[x]) \times 10^d]\), encrypts it using the distributed public key, and then sends the ciphertext \([E(T[x]) \times 10^d]\) to \(\mathcal{P}_i\) through a secure channel.

Sub-Protocol 1. 1: Compute Encrypted Entropy

from multiple data providers, while preserving the data quality for the data mining tasks. The output of this protocol is an LKC-anonymous mashup data.

B. Multi-Party Protocol for Computing Specialization Score

In this section, we present a multi-party protocol for securely determining the best value for specialization. As we discussed in Section II-C, Construction 1 can be used to compute the score of each valid specialization, and then the specialization that yields the highest score is selected. In distributed settings, however, different QID attributes are owned by different parties and the Class attribute is owned by only one party. Therefore, a secure protocol is required to compute the scores and determine the best specialization while ensuring no extra information is leaked to the parties.

Protocol 1 explains how the data providers can securely determine the winner candidate for specialization. As we will see in Section IV-C the table \(T\) is constructed from the leaf partitions of the specialization tree, \(\mathcal{P}^{cls}\), the party that owns the Class attribute, can independently compute the score of its own valid specialization values. On the other hand, any other party that has a valid specialization value must utilize the other parties to achieve that. The intuition is to ask different parties to compute different parts of the each score, and then the party owning the specialization value puts things together by homomorphically computing the total score. Sub-Protocol 1.1 illustrates how \(\mathcal{P}_i\) (the party owning the specialization value \(v\)) and \(\mathcal{P}_j\) (assisting party) compute the entropy of \(T[x]\). The idea is for \(\mathcal{P}_i\) to blind the class ciphertexts corresponding to \(T[x]\) using a random number, decrypt them using its secret decryption share, and then send them to \(\mathcal{P}_j\) who in turn decrypts, counts the equivalent values, and then computes the entropy. Using the same random number to blind each class ciphertext (Step 1 of...
Sub-Protocol 1\(^{[1]}\) ensures the decrypted data is protected but the other party can still count and compute the entropy. We assume that at the beginning of the protocol, all parties agreed on a parameter \(d\) for converting decimal values to integers to be able to encrypt them. Observe that in Equation\(^{[3]}\) we needed to multiply by \(10^d\) to convert \(T[\text{Education}]\) to an integer while maintaining the scale between all computed scores. \(\mathcal{P}_k\) can then perform the multiplication: \(\frac{T[\text{Education}]}{10^d} \times [E(T[\text{Education}])]\). The following example illustrates how to compute the score of a specialization according to Protocol 1\(^{[1]}\).

Example IV.1. Assume that all records in Table III are generalized to Any_Education. Data provider \(\mathcal{P}_1\), who owns the attribute Education, wants to securely compute the score for the specialization Any_Education \(\rightarrow\) {Elementary, Secondary, University} according to \(\text{T_Education}\) from Figure 2.

Step 1. To compute \(E(T[\text{Any_Education}])\), \(\mathcal{P}_1\) blinds all ciphertexts in \(T[\text{Any_Education}]\), \([\text{Class}]\) with a random number \(r_1: [\text{Yes}^s], [\text{No}^s], [\text{No}^r], [\text{Yes}^r], [\text{Yes}^s], [\text{No}^s], [\text{No}^r], [\text{Yes}^r], \) partially decrypts them using its private key share, and the send them to \(\mathcal{P}_3\) (randomly selected). \(\mathcal{P}_3\) then decrypts the ciphertexts using its private key share to obtain the blinded values: [Yes\(^s\)], [No\(^s\)], Yes\(^s\), Yes\(^r\), No\(^s\), No\(^r\), Yes\(^r\), No\(^s\), Yes\(^r\), and computes \(\frac{5}{16} \times \log_2(\frac{5}{16}) - \frac{5}{16} \times \log_2(\frac{5}{16}) = 1\). It then computes the integer \(\lfloor 1 \times 10^2 \rfloor = \lfloor 1 \times 10^2 \rfloor = 100\), and then sends the ciphertext \([100]\) to \(\mathcal{P}_1\).

Step 2. Since the minimum education value in any records \(T[\text{Any_Education}]\) is 7th grade, then no record can be specialized to Elementary. As a result, \(|T[\text{Elementary}]| = 0\) and \(E(T[\text{Elementary}]) = 0\). On the other hand, \(T[\text{Secondary}] = \{\text{UID}\#1, 3, 7, 9\}\) (4 records can be specialized to Secondary). Therefore, \(\mathcal{P}_1\) blinds all ciphertexts in \(T[\text{Secondary}]\), \([\text{Class}]\) with a random number \(r_2: [\text{Yes}^s], [\text{No}^s], [\text{No}^r], [\text{Yes}^r], \) partially decrypts them, and then sends them to \(\mathcal{P}_2\). \(\mathcal{P}_2\) decrypts the ciphertexts and computes \(-\frac{3}{4} \times \log_2(\frac{3}{4}) - \frac{3}{4} \times \log_2(\frac{3}{4}) = 0.8112\). It then computes \([0.8112 \times 10^2] = 81\) and sends \([81]\) to \(\mathcal{P}_1\). Similarly, \(T[\text{University}] = \{\text{UID}\#2, 4, 5, 6, 8, 10\}\) (6 records can be specialized to University). Therefore, \(\mathcal{P}_1\) blinds all ciphertexts in \(T[\text{University}]\), \([\text{Class}]\) with a random number \(r_3: [\text{No}^s], [\text{Yes}^s], [\text{Yes}^s], [\text{No}^r], [\text{Yes}^r], \) partially decrypts them, and then sends them to \(\mathcal{P}_3\). \(\mathcal{P}_3\) decrypts the ciphertexts and computes \(-\frac{3}{4} \times \log_2(\frac{3}{4}) - \frac{3}{4} \times \log_2(\frac{3}{4}) = 0.9183\). It then computes \([0.9183 \times 10^2] = 92\) and sends \([92]\) to \(\mathcal{P}_1\).

Step 3. Since \(|T[\text{Any_Education}]| = 10, \quad |T[\text{Secondary}]| = 4\) and \(|T[\text{University}]| = 6\), \(\mathcal{P}_1\) homomorphically computes the score for specializing on value Any_Education as follows:

\[
\text{Score(Any_Education)} = 10^2 \times |\{\text{Yes}^s\}| - \frac{10^2 \times \log_2(\frac{10^2}{10})}{10} \times 81 - \left[\frac{10^2 \times 5}{10}\right] \times |\{\text{No}^s\}| = 12400.
\]

### Table III: Data owned by \(\mathcal{P}_1, \mathcal{P}_2\) and \(\mathcal{P}_3\) after \(\mathcal{P}_2\) sends encrypted \([\text{Class}]\) attribute to \(\mathcal{P}_1\) and \(\mathcal{P}_3\)

<table>
<thead>
<tr>
<th>UID</th>
<th>Age</th>
<th>Education</th>
<th>Class</th>
<th>Sex</th>
<th>Sen</th>
<th>Salary</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>54</td>
<td>11h</td>
<td>Yes</td>
<td>Male</td>
<td>12</td>
<td>65K</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>26</td>
<td>Bachelor</td>
<td>[No]</td>
<td>No</td>
<td>2</td>
<td>37K</td>
<td>[No]</td>
</tr>
<tr>
<td>3</td>
<td>39</td>
<td>7th</td>
<td>[No]</td>
<td>Female</td>
<td>2</td>
<td>51K</td>
<td>[No]</td>
</tr>
<tr>
<td>4</td>
<td>67</td>
<td>Master</td>
<td>Yes</td>
<td>Female</td>
<td>2</td>
<td>56K</td>
<td>[No]</td>
</tr>
<tr>
<td>5</td>
<td>02</td>
<td>Bachelor</td>
<td>Yes</td>
<td>Male</td>
<td>2</td>
<td>81K</td>
<td>[No]</td>
</tr>
<tr>
<td>6</td>
<td>59</td>
<td>Doctorate</td>
<td>[No]</td>
<td>No</td>
<td>1</td>
<td>107K</td>
<td>[No]</td>
</tr>
<tr>
<td>7</td>
<td>44</td>
<td>12th</td>
<td>[No]</td>
<td>No</td>
<td>2</td>
<td>26K</td>
<td>[No]</td>
</tr>
<tr>
<td>8</td>
<td>29</td>
<td>Bachelor</td>
<td>Yes</td>
<td>Male</td>
<td>2</td>
<td>72K</td>
<td>[No]</td>
</tr>
<tr>
<td>9</td>
<td>35</td>
<td>Master</td>
<td>[No]</td>
<td>Female</td>
<td>2</td>
<td>29K</td>
<td>[No]</td>
</tr>
<tr>
<td>10</td>
<td>46</td>
<td>Master</td>
<td>Yes</td>
<td>Female</td>
<td>2</td>
<td>72K</td>
<td>[No]</td>
</tr>
</tbody>
</table>

### Proposition IV.1. Privacy.

Protocol 1\(^{[1]}\) is privacy-preserving.

Proof: (Sketch) To prove that Protocol 1\(^{[1]}\) is privacy-preserving, we show that the data is protected throughout the protocol execution.

**Encrypted Data.** While encrypted, all ciphertexts exchanged between the parties (encrypted class attributes, entropies and scores) are protected under the CPA-security (Decisional Diffie-Hellman (DDH)) of ElGamal\(^{[9]}\) encryption scheme. The adversary cannot decrypt items arbitrarily, as the decryption key is \((2, n)\)-shared between all data owners, requiring a collusion with another party, which contradicts our non-colluding semi-honest adversarial model.

**Decrypted Data.** In Step 2 of Sub-Protocol 1\(^{[1]}\), \(\mathcal{P}_j\) receives a set of class attribute ciphertexts from \(\mathcal{P}_i\) in order to decrypt and compute the entropy. Decrypting the ciphertexts enables \(\mathcal{P}_j\) to count the equivalent class values. However, since the decrypted data is blinded (exponentiated with a random number), \(\mathcal{P}_j\) cannot determine the actual class values due to the the hardness of computing discrete logarithms. Moreover, using different random numbers for blinding different set of class values prevents \(\mathcal{P}_j\) from comparing two different set of blinded class values it has received from two separate requests. Our protocol, however, leaks partial information about a score to each assisting party, since entropies are computed by assisting parties in clear text. We argue that this leakage is tolerable since assisting party \(\mathcal{P}_j\) can determine neither to which attribute the computed entropy belongs, nor what the underlying class values are.

### C. Multi-Party Protocol for LKK-private Data Mashup Release

Given the distributed data tables \(T_1, \ldots, T_p\), the taxonomy trees for \(\cup QID\), thresholds \(L\), \(K\), and \(C\), the goal is to generate an integrated and anonymous data for data mining while satisfying LKK-privacy. To ensure that no party learns unnecessary information about other parties’ data during the mashup process, we propose a hierarchical approach for specializing the data called Hierarchal High-dimensional Data Mashup (HHDM).

The general idea of our solution is to initially generalize and assign all records to a partition, and then apply a top-
Hierarchal High-dimensional Data Mashup (HHDM)

1) \(P_{cls}\), the party that owns the \textit{Class} attribute, encrypts the class values under the data providers’ distributed public key, and then broadcasts the ciphertexts to the remaining parties.

2) Create an initial partition such that:
   - The hierarchy cut value \(HCut.A\) of every attribute \(A \in QID\) is set to the root of \(T^A\).
   - All records are assigned to the partition.

3) Set the union cut \(UCut\) to the hierarchy cut \(HCut\) of the initial partition.

4) For each value \(v \in UCut\) from a taxonomy tree \(T^A\), \(P^A\) determines whether \(v\) is valid for specialization.

5) While there is at least one value \(v \in UCut\) such that \(v\) is valid for specialization:
   - All parties jointly run Protocol \textit{1} to compute the specialization score for each valid value in \(UCut\), and determine the party that owns the winning value \(w\) with the highest score.
   - The party owning \(w\) runs Sub-Protocol \textit{2.1} to perform the specialization \(w \rightarrow child(w)\).
   - For each value \(v \in UCut\) from a taxonomy tree \(T^A\), \(P^A\) verifies whether \(v\) is valid for specialization.

6) The hierarchy cut \(HCut\) and the record count of each leaf partition constitute the anonymous data for release satisfying the \(LKC\)-private requirements.

\begin{tabular}{|c|c|c|c|c|c|}
\hline
\textbf{P_1} & \textbf{P_2} & \textbf{P_3} & \textbf{Age} & \textbf{Education} & \textbf{Sex} & \textbf{Salary} & \textbf{# of s_i} & \textbf{# of Records} \\
\hline
\textbf{[1-99]} & \textbf{Any_Education} & \textbf{Any_Sex} & \textbf{[10K-125K]} & \textbf{3} & \textbf{10} \\
\hline
\textbf{[1-99]} & \textbf{Secondary} & \textbf{Any_Sex} & \textbf{[10K-125K]} & \textbf{0} & \textbf{4} & \textbf{[10K-70K]} & \textbf{[70K-125K]} & \textbf{[10K-70K]} & \textbf{[70K-125K]} & \textbf{[10K-125K]} & \textbf{3} & \textbf{6} \\
\textbf{[1-99]} & \textbf{Secondary} & \textbf{Any_Sex} & \textbf{[10K-70K]} & \textbf{0} & \textbf{4} & \textbf{[1-99]} & \textbf{University} & \textbf{Any_Sex} & \textbf{[10K-70K]} & \textbf{1} & \textbf{2} & \textbf{1} & \textbf{9} & \textbf{1} & \textbf{2} \\
\textbf{[1-99]} & \textbf{University} & \textbf{Any_Sex} & \textbf{[10K-70K]} & \textbf{1} & \textbf{2} & \textbf{1} & \textbf{9} & \textbf{1} & \textbf{2} \\
\hline
\end{tabular}

Figure 3: Hierarchical high-dimensional data mashup (HHDM) on the data in Table \textit{I}.

Perform Specialization

1) For each partition \(Part\) where \(w \in Part.HCut\):
   a) For each child value \(v \in child(w)\), create a child partition \(CPart\) such that \(CPart.HCut\) is the same as \(Part.HCut\) except that the former, \(w\), is replaced by \(v\).

2) Assign the records in \(Part\) to the child partitions according to Definition \textit{IV.1}

3) Update \(UCut\) by replacing \(w\) by its children \(child(w)\).

Sub-Protocol 2. 1: Perform Specialization

Jointly executed by all data providers, Protocol \textit{2} illustrates how the specialization process is performed in order to generate the \(LKC\)-anonymous table. The parties coordinate their actions using a private broadcast channel called a bulletin board. Initially, all records are assigned to the initial partition. This assignment satisfies Definition \textit{IV.1} since The \(HCut\) of the initial partition contains the most general values (roots) of the taxonomy trees.

Example IV.2. Figure \textit{3} illustrates the specialization process on Table \textit{I} in order to generate an \(LKC\)-anonymous table that satisfies \(L = 2, K = 2\) and \(\chi = 50\%\). The root partition represents the initial partition such that \(HCut = ([1-99], \text{Any_Education}, \text{Any_Sex}, [10K-125K])\) and \(Recs = \{UID\#1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}\). The union cut

down specialization process guided by the taxonomy trees to specialize the records and assign them to disjoint child partitions until no further partitions can be created without violating \(LKC\)-privacy. A \textit{partition} is a data structure that consists of two components: \(HCut\) and \(Recs\). Hierarchy cut \(HCut\) is an ordered set of values \(\langle v_1, \ldots, v_{\#QID} \rangle\), where each value is from a taxonomy tree \(T^A\) of an attribute \(A \in QID\). \(Recs\) contains the unique identifiers \(UIDs\) of the records assigned to the partition.

Definition IV.1. Record Generalization. A record \(R\) can be assigned to a partition \(Part\) if for each attribute \(A \in QID, R.A\) can be generalized to \(Part.HCut.A\), where \(R.A\) and \(Part.HCut.A\) are the values in \(R\) and \(Part.HCut\) that correspond to attribute \(A\), respectively.
is also set to the most general values \( \cup \text{Cut} = \langle [1 - 99), \text{Any}_{\text{Education}}, \text{Any}_{\text{Sex}}, [10K - 125K]\rangle \).

To determine which valid value to specialize on, all parties jointly run Protocol 1[1]. In general, a specialization \( v \rightarrow \text{child}(v) \) involves generating a child partition for each child value in \( \text{child}(v) \). The cut of a taxonomy tree to which \( v \) belongs is pushed downwards, and \( v \) is replaced in the hierarchy cuts of the newly generated partitions by its children values \( \text{child}(v) \). The party that owns the winner value prescribes the actual specialization according to Sub-Protocol 2(1).

**Example IV.3.** In Figure 3, the winner value for the first specialization is \( \text{Any}_{\text{Education}} \). Therefore, part \( \mathcal{P}_1 \), which owns attribute \( \text{Education} \), creates two partitions \( \text{Part}_1 \) and \( \text{Part}_2 \), \( \text{Part}_1.\text{HCut} = \langle [1 - 99), \text{Secondary}, \text{Any}_{\text{Sex}}, [10K - 125K]\rangle \) and \( \text{Part}_2.\text{Recs} = \{ \text{UID} \#1, 3, 7, 9 \} \), and \( \text{Part}_2.\text{HCut} = \langle [1 - 99), \text{University}, \text{Any}_{\text{Sex}}, [10K - 125K]\rangle \) and \( \text{Part}_2.\text{Recs} = \{ \text{UID} \#2, 4, 5, 6, 8, 10 \} \). \( \mathcal{P}_1 \) updates the union cut: \( \cup \text{Cut} = \langle [1 - 99), \text{Secondary}, \text{University}, \text{Any}_{\text{Sex}}, [10K - 125K]\rangle \).

A specialization is valid if after the child partitions are created, the leaf partitions as a whole in the partitioning tree still satisfies LKC-privacy. The specialization process terminates when no more valid specialization is available. The mashup data for the final release can be constructed from the hierarchy cut of the leaf partitions, where each hierarchy cut is duplicated \( |\text{Recs}| \) times (the number of records assigned to the partition).

**Example IV.4.** The output of the specialization process in Figure 3 is: \( \langle [1 - 99), \text{Secondary}, \text{Any}_{\text{Sex}}, [10K - 705K]\rangle \times 4 \), \( \langle [1 - 99), \text{University}, \text{Any}_{\text{Sex}}, [10K - 705K]\rangle \times 2 \), and \( \langle [1 - 99), \text{University}, \text{Any}_{\text{Sex}}, [70K - 125K]\rangle \times 4 \), which is equivalent to the records presented in the LKC-anonymous Table 11.

Based on LKC’s anti-monotonic property [16], once a specialization on a value becomes invalid, further specializations on child(v) will always be invalid. This property significantly reduces the partitioning space, while guaranteeing that the output is suboptimal.

**V. PERFORMANCE EVALUATION**

In this section we evaluate the performance of Fusion. First, we discuss the implementation details, and then we present the experimental results that include mashup utility, attribute scalability and record scalability.

**A. Implementation and Setup**

Fusion is implemented using SCAPI[1] an open-source Java library for implementing secure multiparty computation protocols. We utilize queue-based channels in the communication layer to allow for asynchronous transfer of ciphertexts between parties, where ActiveMQ[2] is used as the messaging broker. The experiments were conducted on a machine equipped with an Intel Core i7 3.8GHz CPU and 16GB RAM, running 64-bit Windows 7.

We utilize a real-life adult data set [17] in our experiments to illustrate the performance of Fusion. The adult data set consists of 45,222 census records containing six numerical attributes, eight categorical attributes, and a class attribute. Table IV lists all the attributes and their types. In our experiments, we model three different distributed settings: 3 parties, 4 parties and 5 parties. We consider attribute \( \text{Marital-status} \) as the sensitive attribute, while we consider the remaining attributes as quasi-identifiers. Table IV illustrates the distribution of each attribute between parties in each of the distributed settings.

**B. Mashup Utility**

Rather than releasing an anonymous mashup data for classification analysis, each data provider could release a classifier of its data. To determine the usefulness of our approach with respect to classification analysis, we utilize C4.5 classifier [13] to compare the classification error of

---

1 SCAPI: https://scapi.readthedocs.org/
2 ActiveMQ: https://activemq.apache.org/
the mashup data with the classification error of the classifier of each party. We use 30,160 records (2/3) to build (train) the classifiers, and 15,062 records (1/3) for testing.

Figure 4 depicts the classification error for each individual party, as well as for the mashup data. The classification error is measured w.r.t. the anonymity threshold $K$, where $K$ linearly increases from 40 and 200. Our approach is robust w.r.t. $L$, since we found out that increasing the prior knowledge of the adversary does not impact the data quality. The classification error for $P_1$ is 17%, $P_2$ is 17.5% and $P_3$ is 18.4%. On the other hand, the mashup classification error decreases from 18.8% to 16.3% as $K$ increases from 40 to 200. Except when $K = 40$, we observe that all data providers benefits from participating in mashup process, where the maximum benefit is as much as 2.1% and the minimum benefit is as low as 0.6%.

C. Scalability

We measure the scalability of Fusion with respect to the number of attributes (Attribute Scalability) and the number of records (Record Scalability) in three distributed settings: 3 parties, 4 parties and 5 parties.

1) Attribute Scalability: Figure 5(a) depicts the runtime from 3 to 13 attributes, for $L = 3$, $K = 40$, $C = 100\%$ and 45,222 records. We observe that the runtime grows sub-linearly when the number of attributes linearly increases, regardless of the number of parties in the setting. We also observe that runtime decreases as the number of parties increases. This is because adding more parties reduces the load on each individual party.

2) Record Scalability: Figure 5(b) depicts the runtime from 200,000 to 1,000,000, for $L = 3$, $K = 40$, $C = 100\%$ and 13 attributes. We observe that it takes up to 195 minutes to run Fusion on a dataset with 1,000,000 records in a 3-party setting. This is mainly due to the fact that we perform a modular exponentiation operation every time a ciphertext is blinded in Sub-Protocol 1.1. However, we also observe that the runtime is still scalable w.r.t. the linear increase in the number of records, regardless of the number of parties in the setting. Similar to Section V-C, we observe that runtime decreases as the number of parties increases.

VI. RELATED WORKS

In this section, we review the literature examining several areas related to our work.

A. Privacy-Preserving Data Publishing (PPDP)

Several privacy models have been proposed in the literature for providing different types of privacy protection, such as $k$-anonymity, $\ell$-diversity, $\epsilon$-differential privacy and $\epsilon$-differential privacy. In this paper, we utilize $\ell$-KC-privacy, a privacy model that generalizes $k$-anonymity, $\ell$-diversity, and $\epsilon$-differential privacy. Unlike our protocol, these protocols are limited to two-parties and not designed to handle high-dimensional data.

There is a related line of work for integrating relational data in a distributed setting: [20], [21] (horizontally partitioned data) and [22], [23], [24] (vertically partitioned data). Alhadidi et al. [20] and Mohammed et al. [23] proposed protocols for securely generating integrated data satisfying $\epsilon$-differential privacy. Unlike our protocol, these protocols are limited to two-parties and not designed to handle high-dimensional data. Fung et al. [25] proposed a privacy-preserving data mashup algorithm for high-dimensional data with $L$-KC-privacy. In contrast to our work, where Class attribute is owned by one party, and a secure protocol is needed to compute the score, the authors relax the problem by assuming that Class attribute is shared between all parties, and the score of each value can be directly computed by the attribute’s owner without any involvement from the other parties. Recently, Arafati et al. [26] proposed a framework for privacy-preserving DaaS mashup to enable secure collaboration between data providers for the purpose of generating a mashup data for data mining. In contrast to our distributed approach, their work assumes the existence of a trusted central party to coordinate the integration process between the parties.

B. Secure Function Evaluation (SFE)

The first two-party protocol was proposed by Yao [15], and later was generalized for secure multi-party computation.
by Goldreich et al. [27]. Our protocol follows the line of research utilizing threshold homomorphic cryptosystems to achieve SFE, also referred to as computing on encrypted data. It was originally introduced in [28] in the semi-honest model, and extended to the malicious model by Schoenmakers and Tuyls [29] (two-parties) and Cramer et al [30] (multi-parties).

VII. CONCLUSIONS AND FURTHER WORK

In this paper, we present a secure protocol for data integration in a distributed setting. The protocol is privacy-preserving, while the output is a mashup data for data mining that satisfy LKC-privacy. We empirically show that the mashup data contain higher information utility, and that the protocol is scalable w.r.t. the number of records as well as the number of attributes in the mashup data. For future work, we plan to address the privacy-preserving data mashup problem in a malicious adversarial model with public verifiability.

REFERENCES


