m-Privacy for Collaborative Data Publishing

Slawomir Goryczka
Emory University
Email: sgorycz@emory.edu

Li Xiong
Emory University
Email: lxiong@emory.edu

Benjamin C. M. Fung
Concordia University
Email: fung@ciise.concordia.ca

***Abstract*—In this paper, we consider the *collaborative data publishing* problem for anonymizing horizontally partitioned data at multiple data providers. We consider a new type of "insider attack" by colluding data providers who may use their own data records (a subset of the overall data) in addition to the external background knowledge to infer the data records contributed by other data providers. The paper addresses this new threat and makes several contributions. First, we introduce the notion of m-privacy, which guarantees that the anonymized data satisfies a given privacy constraint against any group of up to m colluding data providers. Second, we present heuristic algorithms exploiting the equivalence group monotonicity of privacy constraints and adaptive ordering techniques for efficiently checking m-privacy given a set of records. Finally, we present a data *provider-aware* anonymization algorithm with adaptive m-privacy checking strategies to ensure high utility and m-privacy of anonymized data with efficiency. Experiments on real-life datasets suggest that our approach achieves better or comparable utility and efficiency than existing and baseline algorithms while providing m-privacy guarantee.**

I. Introduction

There is an increasing need for sharing data that contain personal information from distributed databases. For example, in the healthcare domain, a national agenda is to develop the Nationwide Health Information Network (NHIN)[1] to share information among hospitals and other providers, and support appropriate use of health information beyond direct patient care with privacy protection.

Privacy preserving data analysis and data publishing [1], [2], [3] have received considerable attention in recent years as promising approaches for sharing data while preserving individual privacy. When the data are distributed among multiple data providers or data owners, two main settings are used for anonymization [2], [4]. One approach is for each provider to anonymize the data independently (anonymize-and-aggregate, **Figure 1A**), which results in potential loss of integrated data utility. A more desirable approach is *collaborative data publishing* [5], [6], [2], [4], which anonymizes data from all providers as if they would come from one source (aggregate-and-anonymize, **Figure 1B**), using either a trusted third-party (TTP) or Secure Multi-party Computation (SMC) protocols to do computations [7], [8].

Problem Settings. We consider the collaborative data publishing setting (**Figure 1B**) with horizontally partitioned data across multiple data providers, each contributing a subset of records T_i. As a special case, a data provider could be the data owner itself who is contributing its own records. This is a very common scenario in social networking and recommendation systems. Our goal is to publish an anonymized view of the integrated data such that a data recipient including the data providers will not be able to compromise the privacy of the individual records provided by other parties. Considering different types of malicious users and information they can use in attacks, we identify three main categories of attack scenarios. While the first two are addressed in existing work, the last one receives little attention and will be the focus of this paper.

[1]http://www.hhs.gov/healthit/healthnetwork/background/

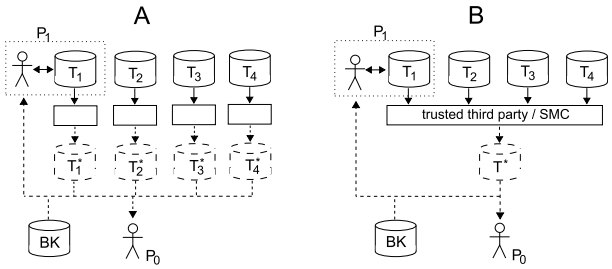

Fig. 1. Distributed data publishing settings.

Attacks by External Data Recipient Using Anonymized Data. A data recipient, e.g. P_0, could be an attacker and attempts to infer additional information about the records using the published data (T^*) and some background knowledge (BK) such as publicly available external data. Most literature on privacy preserving data publishing in a single provider setting considers only such attacks [2]. Many of them adopt a weak or relaxed *adversarial* or *Bayes-optimal privacy* notion [9] to protect against specific types of attacks by assuming limited background knowledge. For example, k-anonymity [10], [11] prevents identity disclosure attacks by requiring each equivalence group, records with the same quasi-identifier values, to contain at least k records. Representative constraints that prevent attribute disclosure attacks include l-diversity, which requires each equivalence group to contain at least l "well-represented" sensitive values [9], and t-closeness [12], which requires the distribution of a sensitive attribute in any equivalence group to be close to its distribution in the whole population. In contrast, differential privacy [1], [3] publishes statistical data or computational results of data and gives unconditional privacy guarantees independent of attackers background knowledge.

Attacks by Data Providers Using Intermediate Results and Their Own Data. We assume the data providers are semi-honest [7], [8], commonly used in distributed computation

setting. They can attempt to infer additional information about data coming from other providers by analyzing the data received during the anonymization. A trusted third party (TTP) or Secure Multi-Party Computation (SMC) protocols (e.g. [5]) can be used to guarantee there is no disclosure of *intermediate* information *during* the anonymization. However, either TTP or SMC do not protect against data providers to infer additional information about other records using the anonymized data and their own data (discussed below). Since the problem is orthogonal to whether a TTP or SMC is used for implementing the algorithm, without loss of generality, we have assumed that all providers use a TTP for anonymization and note that an SMC variant can be implemented.

Attacks by Data Providers Using Anonymized Data and Their Own Data. Each data provider, such as P_1 in **Figure 1**, can also use anonymized data T^* and his own data (T_1) to infer additional information about other records. Compared to the attack by the external recipient in the first attack scenario, each provider has additional data knowledge of their own records, which can help with the attack. This issue can be further worsened when multiple data providers collude with each other. In the social network or recommendation setting, a user (having an account herself) may attempt to infer private information about other users using the anonymized data or recommendations assisted by some background knowledge and her own account information. Malicious users may collude or even create artificial accounts as in a shilling attack [13].

We define and address this new type of "insider attack" by data providers in this paper. In general, we define an m-*adversary* as a coalition of m colluding data providers or data owners, who have access to their own data records as well as publicly available background knowledge BK and attempts to infer data records contributed by other data providers. Note that 0-adversary can be used to model the external data recipient, who has only access to the external background knowledge. Since each provider holds a subset of the overall data, this inherent data knowledge has to be explicitly modeled and checked when the data are anonymized using a weak privacy constraint and assuming no instance level knowledge.

We illustrate the m-adversary threats with an example shown in **Table I**. Assume that hospitals P_1, P_2, P_3, and P_4 wish to collaboratively anonymize their respective patient databases T_1, T_2, T_3, and T_4. In each database, **Name** is an identifier, {**Age**, **Zip**} is a quasi-identifier (QI), and **Disease** is a sensitive attribute. T_a^* is one possible QI-group-based anonymization using existing approaches that guarantees k-anonymity and l-diversity ($k = 3$, $l = 2$). Note that l-diversity holds if each equivalence group contains records with at least l different sensitive attribute values. However, an attacker from the hospital P_1, who has access to T_1, may remove all records from T_a^* that are also in T_1 and find out that there is only one patient between 20 and 30 years old. Combining this information with background knowledge BK, P_1 can identify Sara's record (highlighted in the table) and her disease Epilepsy. In general, multiple providers may collude with each other, hence having access to the union of their data, or a user may have access to multiple databases, e.g. a physician switching to another hospital, and use the increased data knowledge to infer data at other nodes.

T_1

Name	Age	Zip	Disease
Alice	24	98745	Cancer
Bob	35	12367	Asthma
Emily	22	98712	Asthma

T_2

Name	Age	Zip	Disease
Dorothy	38	98701	Cancer
Mark	37	12389	Flu
John	31	12399	Flu

T_3

Name	Age	Zip	Disease
Sara	20	12300	Epilepsy
Cecilia	39	98708	Flu

T_4

Name	Age	Zip	Disease
Olga	32	12337	Cancer
Frank	33	12388	Asthma

T_a^*

Provider	*Name*	Age	Zip	Disease
P_1	*Alice*	[20-30]	*****	Cancer
P_1	*Emily*	[20-30]	*****	Asthma
P_3	***Sara***	**[20-30]**	*****	**Epilepsy**
P_1	*Bob*	[31-35]	*****	Asthma
P_2	*John*	[31-35]	*****	Flu
P_4	*Olga*	[31-35]	*****	Cancer
P_4	*Frank*	[31-35]	*****	Asthma
P_2	*Dorothy*	[36-40]	*****	Cancer
P_2	*Mark*	[36-40]	*****	Flu
P_3	*Cecilia*	[36-40]	*****	Flu

T_b^*

Provider	*Name*	Age	Zip	Disease
P_1	*Alice*	[20-40]	*****	Cancer
P_2	*Mark*	[20-40]	*****	Flu
P_3	*Sara*	[20-40]	*****	Epilepsy
P_1	*Emily*	[20-40]	987**	Asthma
P_2	*Dorothy*	[20-40]	987**	Cancer
P_3	*Cecilia*	[20-40]	987**	Flu
P_1	*Bob*	[20-40]	123**	Asthma
P_4	*Olga*	[20-40]	123**	Cancer
P_4	*Frank*	[20-40]	123**	Asthma
P_2	*John*	[20-40]	123**	Flu

TABLE I
m-ADVERSARY AND m-PRIVACY EXAMPLE.

Contributions. In this paper, we address the new threat by m-adversaries and make several important contributions. First, we introduce the notion of m-privacy that explicitly models the inherent data knowledge of an m-adversary and protects anonymized data against such adversaries with respect to a given privacy constraint. For example, an anonymization satisfies m-privacy with respect to l-diversity if the records in each equivalence group excluding ones from any m-adversary still satisfy l-diversity. In our example in **Table I**, T_b^* is an anonymization that satisfies m-privacy ($m = 1$) with respect to k-anonymity and l-diversity ($k = 3$, $l = 2$).

Second, to address the challenges of checking a combinatorial number of potential m-adversaries, we present heuristic algorithms for efficiently verifying m-privacy given a set of records. Our approach utilizes effective pruning strategies exploiting the equivalence group monotonicity property of privacy constraints and adaptive ordering techniques based on a novel notion of privacy fitness. Finally, we present a data *provider-aware* anonymization algorithm with adaptive strategies of checking m-privacy to ensure high utility and m-privacy of sanitized data with efficiency. We experimentally show the feasibility and benefits of our approach using real-world dataset.

II. m-PRIVACY DEFINITION

We first formally describe our problem setting. Then we present our m-privacy definition with respect to a given privacy constraint to prevent inference attacks by m-adversary, followed by its properties.

Let $T = \{t_1, t_2, \ldots\}$ be a set of records horizontally distributed among n data providers $P = \{P_1, P_2, \ldots, P_n\}$, such that $T_i \subseteq T$ is a set of records provided by P_i. We assume A_S is a sensitive attribute with domain D_S. If the records contain multiple sensitive attributes then a new sensitive attribute A_S can be defined as a Cartesian product of all sensitive attributes. Our goal is to publish an anonymized table T^* while preventing any m-adversary from inferring A_S for any single record.

A. m-Privacy

To protect data from external recipients with certain background knowledge BK, we assume a given privacy requirement C, defined by a conjunction of privacy constraints: $C_1 \wedge C_2 \wedge \ldots \wedge C_w$. If a set of records T^* satisfies C, we say $C(T^*) = true$. Any of the existing privacy principles can be used as a component constraint.

In our example (**Table I**), the privacy constraint C is defined as $C = C_1 \wedge C_2$, where C_1 is k-anonymity with $k = 3$, and C_2 is l-diversity with $l = 2$. Both anonymized tables, T_a^* and T_b^* satisfies C, although as we have shown earlier, T_a^* may be compromised by an m-adversary such as P_1.

We now formally define a notion of m-privacy with respect to a privacy constraint C, to protect the anonymized data against m-adversaries in addition to the external data recipients. The notion explicitly models the inherent data knowledge of an m-adversary, the data records they jointly contribute, and requires that each equivalence group, excluding *any* of those records owned by an m-adversary, still satisfies C.

Definition 2.1: (m-PRIVACY) Given n data providers, a set of records T, and an anonymization mechanism $\mathcal{A}$, an m-adversary I ($m \leqslant n-1$) is a coalition of m providers, which jointly contributes a set of records T_I. Sanitized records $T^* = \mathcal{A}(T)$ satisfy m-privacy, i.e. are m-private, with respect to a privacy constraint C, if and only if,

$$\forall I \subset P, |I| = m, \forall T' \subseteq T : T' \supseteq T \setminus T_I, C(\mathcal{A}(T')) = true$$

Observation 2.1: For all $m \leqslant n-1$, if T^* is m-private, then it is also $(m-1)$-private. If T^* is not m-private, then it is also not $(m+1)$-private.

Note that this observation describes monotonicity of m-privacy with respect to number of adversaries, which is independent from the privacy constraint C and records. In the next section we investigate monotonicity of m-privacy with respect to records with given value of m.

m-Privacy and Weak Privacy Constraints. Given a weak privacy constraint C that does not consider instance level background knowledge, such as k-anonymity, l-diversity, and t-closeness, a T^* satisfying C will only guarantee 0-privacy w.r.t. C, i.e. C is not guaranteed to hold for each equivalence group after excluding records belonging to any malicious data provider. Thus, each data provider may be able to breach privacy of records provided by others. In our example from **Table I**, T_a^* satisfies only 0-privacy w.r.t. C = k-anonymity $\wedge$ l-diversity ($k = 3$, $l = 2$), while T_b^* satisfies 1-privacy w.r.t. the same C.

m-Privacy is defined w.r.t. a privacy constraint C, and hence will inherit strengths and weaknesses of C. For example, if C is defined by k-anonymity, then ensuring m-privacy w.r.t. C will not protect against homogeneity attack [9] or deFinetti attack [14]. However, m-privacy w.r.t. C will protect against a privacy attack issued by any m-adversary, if and only if, C protects against the same privacy attack by any external data recipient. m-Privacy constraint is orthogonal to the privacy constraint C being used.

m-Privacy and Differential Privacy. Differential privacy [1], [3], [15] does not assume specific background knowledge and guarantees privacy even if an attacker knows all records except the victim record. Thus, any statistical data (or records synthesized from the statistical data) satisfying differential privacy also satisfies $(n-1)$-privacy, i.e. maximum level of m-privacy, when any $(n-1)$ providers can collude.

While m-privacy w.r.t. any weak privacy notion does not guarantee unconditional privacy, it offers a practical tradeoff between preventing m-adversary attacks with bounded power m and the ability to publish generalized but truthful data records. In the rest of the paper, we will focus on checking and achieving m-privacy w.r.t. weak privacy constraints.

B. Monotonicity of Privacy Constraints

Generalization based monotonicity has been defined for privacy constraints in the literature (Definition 2.2) [9], [12] and has been used for designing efficient generalization algorithms to satisfy a privacy constraint ([11], [16], [9], [12]). In this paper we will refer to it as *generalization monotonicity*.

Definition 2.2: (GENERALIZATION MONOTONICITY OF A PRIVACY CONSTRAINT [9], [12]) A privacy constraint C is generalization monotonic if and only if for any set of anonymized records T^* satisfying C, all its further generalizations satisfy C as well.

Generalization monotonicity assumes that original records T have been already anonymized and uses them for further generalizations. In this paper, we also introduce more general, record-based definition of monotonicity in order to facilitate the analysis and design of efficient algorithms for checking m-privacy.

Definition 2.3: (EQUIVALENCE GROUP MONOTONICITY OF A PRIVACY CONSTRAINT, EG MONOTONICITY) A privacy constraint C is EG monotonic if and only if any set of anonymized records T^* satisfies C, then all supersets of T^* with the same QI attribute satisfy C as well,

$$C(T^*)holds \Leftrightarrow \forall \widetilde{T^*} \supset T^* : QI(T^*) = QI(\widetilde{T^*}), C(\widetilde{T^*})holds$$

EG monotonicity is more restrictive than generalization monotonicity. If a constraint is EG monotonic, it is also generalization monotonic. But vice versa does not always hold. k-Anonymity and l-diversity that requires l distinct values

of sensitive attribute in an equivalence group are examples of EG monotonic constraints, which are also generalization monotonic. Entropy l-diversity [9] and t-closeness [12] are examples of generalization monotonic constraints that are not EG monotonic at the same time. For example, consider a subset of two anonymized records with 2 different sensitive values satisfying entropy l-diversity ($l = 2$), i.e. distribution of sensitive attribute values in the group is uniform. Entropy l-diversity is not EG monotonic because it will not hold if we add a record that will change the distribution of sensitive values (and entropy) significantly. However, it is generalization monotonic because it will still hold if any other subgroup satisfying entropy l-diversity ($l = 2$) is added (generalized) into the first subgroup.

Observation 2.2: If all constraints in a conjunction $C = C_1 \wedge C_2 \wedge \ldots \wedge C_w$ are EG monotonic, then the constraint C is EG monotonic.

Similar observation holds for generalization monotonicity. In our example, C is defined as a conjunction of k-anonymity and l-diversity. Since both of them are EG monotonic [9], C is EG monotonic.

Theorem 2.1: m-Privacy with respect to a constraint C is EG monotonic if and only if C is EG monotonic.

Due to limited space, the proof of this theorem has been moved to our technical report [17]. This theorem and its proof holds also when applied for generalization monotonicity. Note that monotonicity in this theorem is defined with respect to records and not m.

Observation 2.3: If a constraint C is EG monotonic, then the definition of m-privacy w.r.t. C (Definition 2.1) may be simplified. $T^* = \mathcal{A}(T)$ satisfies m-privacy w.r.t. C, if and only if,

$$\forall I \subset P, |I| = m, C \text{ is monotonic}, C\left(\mathcal{A}\left(T \backslash T_I\right)\right) = true$$

Indeed, for an EG monotonic C, if a coalition I cannot breach privacy, then any sub-coalition with fewer records cannot do so either (Definition 2.3). Unfortunately, generalization monotonicity of C is not sufficient for the simplification presented in this observation.

III. Verification of m-Privacy

Checking whether a set of records satisfies m-privacy creates a potential computational challenge due to the combinatorial number of m-adversaries that need to be checked. In this section, we first analyze the problem by modeling the checking space. Then we present heuristic algorithms with effective pruning strategies and adaptive ordering techniques for efficiently checking m-privacy for a set of records w.r.t. an EG monotonic privacy constraint C.

A. Adversary Space Enumeration

Given a set of n_G data providers, the entire space of m-adversaries (m varying from 0 to $n_G - 1$) can be represented using a lattice shown in **Figure 2**. Each node at layer m represents an m-adversary of a particular combination of m providers. The number of all possible m-adversaries is equal to $\binom{n_G}{m}$. Each node has parents (children) representing their direct super- (sub-) coalitions. For simplicity the space is also represented as a *diamond*, where a horizontal line corresponds to all m-adversaries with the same m value, the bottom node corresponds to 0-adversary (external data recipient), and the top line to $(n_G - 1)$-adversaries.

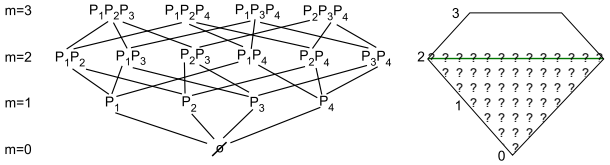

Fig. 2. m-Adversary space.

In order to verify m-privacy w.r.t. a constraint C for a set of records, we need to check C for the records excluding any subset of records owned by any m-adversary. When C is EG monotonic, we only need to check C for the records excluding all records from any m-adversary (Observation 2.3). For example, in **Figure 2**, given $m = 2$, all coalitions that need to be checked are represented by question marks. If C is EG monotonic, then it is sufficient to check only the question marks on the horizontal line. Given an EG monotonic constraint, a *direct* algorithm can sequentially generate all possible $\binom{n_G}{m}$ m-adversaries and then check privacy of the corresponding remaining records. The complexity is then determined by $\binom{n_G}{m}$. In the worst-case scenario, when $m = n_G/2$, the number of checks is equal to the central binomial coefficient $\binom{n_G}{n_G/2}$. In the remainder of this section, we will focus on the EG monotonic case and present heuristic algorithms for efficiently checking m-privacy.

B. Heuristic Algorithms

The key idea of our heuristic algorithms is to efficiently search through the adversary space with effective pruning such that not all m-adversaries need to be checked. This is achieved by two different pruning strategies, an adversary ordering technique, and a set of search strategies that enable fast pruning.

Pruning Strategies. The pruning strategies are possible thanks to the EG monotonicity of m-privacy (Observations 2.1, 2.3). If a coalition is not able to breach privacy, then all its sub-coalitions will not be able to do so and hence do not need to be checked (downward pruning). On the other hand, if a coalition is able to breach privacy, then all its super-coalitions will be able to do so and hence do not need to be checked (upward pruning). In fact, if a sub-coalition of an m-adversary is able to breach privacy, then the upward pruning allows the algorithm to terminate immediately as the m-adversary will be able to breach privacy (*early stop*). **Figure 3** illustrates the two pruning strategies where $+$ represents a case when a coalition does not breach privacy and $-$ otherwise.

Adaptive Ordering of Adversaries. In order to facilitate the above pruning in both directions, we adaptively order the coalitions based on their attack powers (**Figure 4**). This is motivated by the following observations. For downward pruning, super-coalitions of m-adversaries with limited attack powers are preferred to check first as they are less likely to

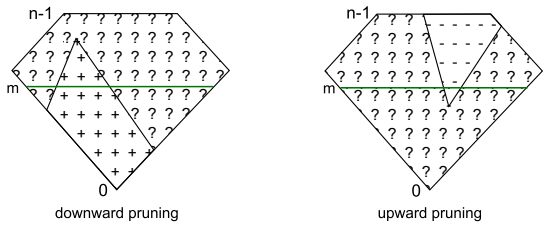

Fig. 3. Pruning strategies for m-privacy check.

breach privacy and hence increase the chance of downward pruning. In contrast, sub-coalitions of m-adversaries with significant attack powers are preferred to check first as they are more likely to breach privacy and hence increase the chance of upward pruning (early-stop).

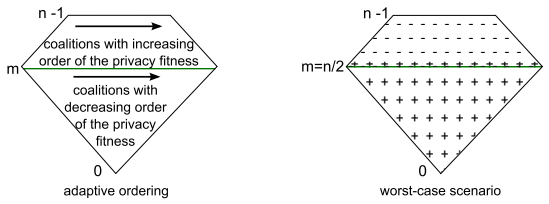

Fig. 4. Adaptive ordering for efficient pruning and the worst-case scenario without any pruning possible.

To quantify privacy fulfillment by a set of records, which is used to measure the attack power of a coalition and privacy of remaning records (used to facilitate the anonymization, which we will discuss in next section), we introduce the privacy fitness score w.r.t. C for a set of records.

Definition 3.1: (PRIVACY FITNESS SCORE) Privacy fitness F_C for a set of records T^* is a level of the fulfillment of the privacy constraint C. A privacy fitness score is a function f of privacy fitness with values greater or equal to 1 only if $C(T^*) = true$,

$$score_{F_C(T^*)} = f\left(F_{C_1}(T^*), F_{C_2}(T^*), \ldots, F_{C_w}(T^*)\right)$$

In our setting, C is defined as k-anonymity $\wedge$ l-diversity. The privacy fitness score can be defined as a weighted average of the two fitness scores with $\alpha \in (0, 1)$. When $C(T^*) = false$, $score_{F_C(T^*)} = max(1-\epsilon, F_C(T^*))$, where ϵ is small. In our example $score_{F_C}$ is defined as follow:

$$score_{F_{C_1 \wedge C_2}}(T^*) = (1-\alpha)\cdot\frac{|T^*|}{k} + \alpha\cdot\frac{|\{t[A_S] : t \in T^*\}|}{l} \quad (1)$$

The attack power of a coalition can be then measured by the privacy fitness score of the records jointly contributed by its members, as the higher the privacy fitness score, the more likely they will be able to breach the privacy for the remaining records in a group after removing their own records. In order to maximize the benefit of both pruning strategies, the super-coalitions of m-adversaries are generated in the order of ascending fitness scores (ascending attack powers), and the sub-coalitions of m-adversaries are generated in the order of descending fitness scores (descending attack powers) (**Figure 4**).

Now we present several heuristic algorithms that use different search strategies, and hence utilize different pruning. All of them use the adaptive ordering of adversaries to enable fast pruning.

The Top-Down Algorithm. The *top-down* algorithm checks the coalitions in a top-down fashion using downward pruning, starting from $(n_G - 1)$-adversaries and moving down until a violation by an m-adversary is detected or all m-adversaries are pruned or checked.

The Bottom-Up Algorithm. The *bottom-up* algorithm checks coalitions in a bottom up fashion using upward pruning, starting from 0-adversary and moving up until a violation by any adversary is detected (*early-stop*) or all m-adversaries are checked.

The Binary Algorithm. The *binary* algorithm, inspired by the binary search algorithm, checks coalitions between $(n_G - 1)$-adversaries and m-adversaries and takes advantage of both upward and downward prunings (**Figure 5**, Algorithm 1). The goal of each iteration is to search for a pair I_{sub} and I_{super}, such that I_{sub} is a direct sub-coalition of I_{super} and I_{super} breaches privacy while I_{sub} does not. Then I_{sub} and all its sub-coalitions are pruned (downward pruning), I_{super} and all its super-coalitions are pruned (upward pruning) as well.

Algorithm 1: The *binary* verification algorithm

Data: A set of records T provided by $P_1, \ldots, P_{n_G}$, a monotonic privacy constraint C, a privacy fitness scoring function $score_F$ and the m value
Result: *true* if T^* is m-private, *false* otherwise

```
1  begin
2      sites = sort_sites(P, increasing_order, score_F)
3      use_adaptive_order_generator(sites, m)
4      while is_m-privacy_verified(T*, m) = false do
5          I_super = next_coalition_of_size(n − 1)
6          if privacy_is_breached_by(I_super) then
7              continue
8          I_sub = next_sub-coalition_of(I_super, m)
9          if privacy_is_breached_by(I_sub) then
10             return false //early stop
11         while is_coalition_between(I_sub, I_super) do
12             I = next_coalition_between(I_sub, I_super)
13             if privacy_is_breached_by(I) then
14                 I_super = I
15             else
16                 I_sub = I
17         prune_all_sub-coalitions(I_sub)
18         prune_all_super-coalitions(I_super)
19     return true
```

The search works as follows. First, it starts with $(n_G - 1)$-adversaries and finds the first one that violates privacy and assigns it to I_{super} (lines from 5 to 7). Then, it finds an m-adversary that is a sub-coalition of I_{super} and assigns it to I_{sub} (line 8). At each step, a new coalition $I : I_{sub} \subset I \subset I_{super}$ (such that $|I| = \frac{|I_{super}|+|I_{sub}|}{2}$; line 12) is checked (line 13). If I violates privacy, then I_{super} is updated to I (line 14). Otherwise, I_{sub} is updated to I (line 16). It continues until the direct parent-child pair I_{super} and I_{sub} are found (line 11). Then both upward and downward prunings are performed (lines 17 and 18) and the algorithm moves to the next iteration. The algorithm stops with the same criteria as the top down algorithm (line 4).

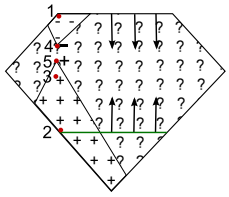

Fig. 5. The *binary* verification algorithm.

Adaptive Selection of Algorithms. Each of the above algorithms focuses on different search strategy, and hence utilizes different pruning. Which algorithm to use is largely dependent on the characteristics of a given group of providers. Intuitively, the privacy fitness score (**Equation 1**), which quantifies the level of privacy fulfillment of records, may be used to select the most suitable verification algorithm. The higher the fitness score of *attacked* records, the more likely m-privacy will be satisfied, and hence a top-down algorithm with downward pruning will significantly reduce the number of adversary checks. We utilize such an adaptive strategy in the anonymization algorithm (discussed in the next section) and will experimentally compare and evaluate different algorithms in the experiment section.

C. Time Complexity

In this section, we derive the time complexity for the m-privacy verification algorithms. Since the algorithms involve multiple checks of privacy constraint C used to define m-privacy for various combinations of records, we assume that each check of C takes a constant time. Formally, it can be modeled by an oracle, which performs the check for given records in $O(1)$ time.

All the above verification algorithms have the same worst-case scenario (**Figure 4**), in which all super-coalitions of m-adversaries violate privacy, while all sub-coalitions of m-adversaries do not. Hence neither adaptive ordering nor pruning strategies are useful. The *direct* algorithm will check exactly $\binom{n_G}{m}$ m-adversaries before confirming m-privacy, where n_G is the number of data providers contributing to the group. This is the minimal number of privacy verifications for this scenario and any other algorithm will execute at least that many privacy checks. The *bottom-up* algorithm will check 0-adversary (external data recipient) up to all m-adversaries, which requires $\sum_{i=0}^{m}\binom{n_G}{i} = O\left(n_G^m\right)$ checks. The *top-down* algorithm will check all (n_G-1)-adversaries to all m-adversaries, which requires $\sum_{i=m}^{n_G-1}\binom{n_G}{i} = O\left(n_G^{n_G-1-m}\right)$ checks. The *binary* algorithm will run $\binom{n_G}{m}$ iterations and within each $O(\log(n_G-m))$ privacy checks. Thus, the total time complexity is $O\left(n_G^m \log(n_G-m)\right)$.

The average time complexity analysis is more involved. The average time complexity is strongly correlated with value of m for all algorithms. For each of them the lower bound of the average time complexity is $O(n_G)$. The upper bound of the average time complexity is different for each algorithm, that is $O\left((3/2)^{n_G}\right)$ for *top-down*, $O\left(2^{n_G} n_G^{-1/2}\right)$ for both *bottom-up* and *direct*, and $O\left(2^{n_G}\frac{\log_2 n_G}{n_G}\right)$ for *binary*. Thus, adapting m-privacy verification strategy to domain settings is crucial to achieve, on average, a low runtime. The analysis details are omitted in this paper due to space restrictions. Please refer to our technical report [17] for how we derived the bounds.

IV. Anonymization for m-Privacy

After defining the m-privacy verification algorithm, we can now use it in anonymization of a horizontally distributed dataset to achieve m-privacy. In this section, we will present a baseline algorithm, and then our approach that utilizes a data provider-aware algorithm with adaptive m-privacy checking strategies to ensure high utility and m-privacy for anonymized data.

Since we have shown that m-privacy with respect to a generalization monotonic constraint is generalization monotonic (Theorem 2.1), most existing generalization-based anonymization algorithms can be modified to achieve m-privacy – every time a set of records is tested for a privacy constraint C, we check m-privacy w.r.t. C instead. As a **baseline algorithm** to achieve m-privacy, we adapted the multidimensional Mondrian algorithm [16] designed for k-anonymity. A main limitation of such a simple adaptation is that groups of records are formed *oblivious* of the data providers, which may result in over-generalization in order to satisfy m-privacy.

We introduce a simple and general algorithm based on the Binary Space Partitioning (BSP) (Algorithm 2). Similar to the Mondrian algorithm, which is also an example of BSP algorithms, it recursively chooses an attribute to split data points in the multidimensional domain space until the data cannot be split any further while satisfying m-privacy w.r.t. C. However, the algorithm has three novel features: 1) it takes into account the data provider as an additional dimension for splitting; 2) it uses the privacy fitness score as a general scoring metric for selecting the split point; 3) it adapts its m-privacy verification strategy for efficient verification. The pseudo code for our *provider-aware* anonymization algorithm is presented in Algorithm 2. We describe the algorithm details with respect to the novel features below.

Algorithm 2: The *provider-aware* algorithm.

Data: A set of records $T = \bigcup_{j=1}^{n} T_j$ provided by $\{P_1, P_2, \ldots, P_n\}$, a set of QI attributes A_i $(i = 1, \ldots, q)$, m, a privacy constraint C
Result: Anonymized T^* that satisfies m-privacy w.r.t. C

```
1  begin
2      π = get_splitting_points_for_attributes(A_i)
3      π = π∪ get_splitting_point_for_providers(A_0)
4      π' = {a_i ∈ π, i ∈ {0,1,...,q} :
       are_both_split_subpartitions_m-private(T, a_i)}
5      if π' is ∅ then
6          T* = T*∪ generalize_all_QIs(T)
7          return T*
8      A_j = choose_splitting_attribute(T, C, π')
9      (T'_r, T'_l) = split(T, A_j)
10     Run recursively for T'_l and T'_r
```

Provider-Aware Partitioning. The algorithm first generates all possible splitting points, π, for QI attributes and data providers (line 2 and 3 of Algorithm 2). In addition to the multidimensional QI domain space, we consider the data provider or data source of each record as an additional attribute

of each record, denoted as A_0. For instance, each data record t contributed by data provider P_1 in our example (**Table I**) will have $t[A_0] = P_1$. Introducing this additional attribute in our multi-dimensional space adds a new dimension for partitioning. Using A_0 to split data points decreases number of providers in each partition and hence increases the chances that more sub-partitions will be m-private and feasible for further splits. This leads to more splits resulting a more precise view of the data and have a direct impact on the anonymized data utility. To find the potential split point along this dimension, we can impose a total order on the providers, e.g. sorting the providers alphabetically or based on the number of records they provide, and find the splitting point that partitions the records into two approximately equal groups.

Adaptive m-privacy verification. m-Privacy is then verified for all possible splitting points and only those satisfying m-privacy are added to a candidate set π' (line 4). In order to minimize the time, our algorithm adaptively selects an m-privacy verification strategy using the fitness score of the partitions. Intuitively, in the early stage of the anonymization algorithm, the partitions are large and likely m-private. A *top-down* algorithm, which takes advantage of the downward pruning, may be used for fast verification. However, as the algorithm continues, the partitions become smaller, the downward pruning is less likely and the *top-down* algorithm will be less efficient. A *binary* algorithm or others may be used instead to allow upward pruning. We experimentally determine the threshold of privacy fitness score for selecting the best verification algorithm and verify the benefit of this strategy.

Privacy Fitness Score Based Splitting Point Selection. Given a non-empty candidate set π' (Algorithm 2), we use the privacy fitness score (Definition 3.1) defined in the previous section and choose the best splitting point (line 8). Intuitively, if the resulting partitions have higher fitness scores, they are more likely to satisfy m-privacy with respect to the privacy constraint and allow for more further splitting. We note that the fitness score does not have to be exactly the same function used for adaptive ordering in m-privacy check. For example, if we use **Equation 1**, the weight parameter used to balance fitness values of privacy constraints, should have, most likely, different value. The algorithm then splits the partition and runs recursively on each sub-partition (line 9 and 10).

V. Experimental Results

We present two sets of experiment results with the following goals: 1) to compare and evaluate the different m-privacy verification algorithms given a set of records, and 2) to evaluate and compare the proposed anonymization algorithm for a given dataset with the baseline algorithm in terms of both utility and efficiency.

A. Experiment Setup

We used combined training and test sets of the Adult dataset[2]. Records with missing attribute values have been removed. All remaining 45,222 records have been used in all experiments. The *Occupation* has been chosen as a sensitive attribute A_S. This attribute has 14 distinct values. Data are distributed among n data providers $P_1, P_2, \ldots, P_n$ such that their distribution follows a uniform or exponential distribution. We observe similar results for both of them and only report those for the exponential distribution in the paper.

The privacy constraint C is defined by k-anonymity [11] and l-diversity [9]. C is EG monotonic (Definition 2.3). We note again m-privacy is orthogonal to the privacy constraint being used in its definition. Both m-privacy verification and anonymization use privacy fitness scores, but with different values of the weight parameter α. Values of α can be defined in a way that reflects restrictiveness of privacy constraints. The impact of the weight parameter to overall performance was experimentally investigated and values of α for the most efficient runs have been chosen as defaults. All experiment and algorithm parameters, and their default values are listed in **Table II**.

Name	Description	Verification	Anonymization
α	Weight paramter	0.3	0.8
m	Power of m-privacy	5	3
n	Total number of data providers	–	10
n_G	Number of data providers contributing to a group	15	–
$\|T\|$	Total number of records	–	45,222
$\|T_G\|$	Number of records in a group	{150, 750}	–
k	Parameter of k-anonymity	50	30
l	Parameter of l-diversity	4	4

TABLE II
EXPERIMENT PARAMETERS AND DEFAULT VALUES.

All experiments have been performed on Sun Microsystems SunFire V880 with 8 CPUs, 16 GB of RAM, and running Solaris 5.10.

B. m-Privacy Verification

The objective of the first set of experiments is to evaluate the efficiency of different algorithms for m-privacy verification given a set of records T_G with respect to the previously defined privacy constraint C.

Attack Power. In this experiment, we compared the different m-privacy verification heuristics against different attack powers. We used two different groups of records with relatively small and large average number of records per data provider, respectively. **Figure 6** shows the runtime with varying m for different heuristics for the two groups.

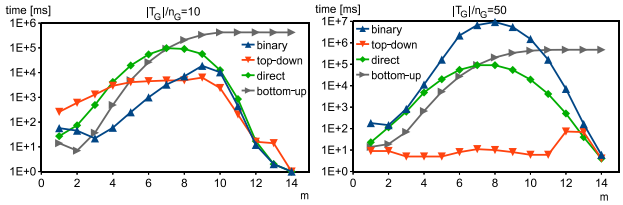

Fig. 6. Runtime (logarithmic scale) vs. m.

The first group counts 150 records and has a small average fitness score per provider (equal to 0.867), which reflects a high probability of privacy breach by a large m-adversary. For almost all values of m the *binary* algorithm achieves the

[2]The Adult dataset has been prepared using the Census database from 1994, http://archive.ics.uci.edu/ml/datasets/Adult

best performance due to its efficient upward and downward pruning. However, the *top-down* algorithm is comparable with *binary* for $m > n_G/2$.

The second group counts 750 records and has a larger average fitness score per provider (equal to 2.307). Therefore intuitively, it is very unlikely that a coalition of adversaries will be able to breach privacy and the downward pruning can be applied often. This intuition is confirmed by results, which show that the *top-down* algorithm is significantly better than other heuristics. Since the remaining algorithms do not rely so much on the downward pruning, they have to perform an exponential number of checks. We can also observe a clear impact of m when $m \approx n_G/2$ incurs the highest cost.

Number of Contributing Data Providers. In this experiment, we analyzed the impact of contributing data providers (n_G) on the different algorithms for the small and large group respectively. **Figure 7** shows the runtime of different heuristics with varying number of contributing data providers n_G.

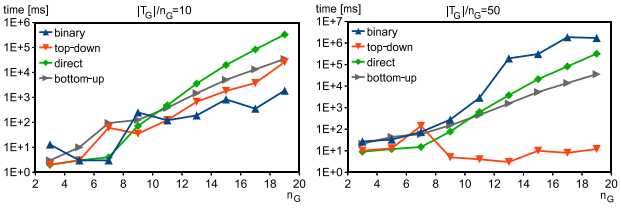

Fig. 7. Runtime (logarithmic scale) vs. number of data providers.

We observe that increasing the number of contributing data providers has different impacts on different algorithms in the two group settings. In the first group, where the average number of records per provider is small, the execution time for each algorithm grows exponentially. In this case the set of records has a low privacy fitness score and is very vulnerable to attacks from m-adversaries. Adding more providers will exponentially increase the domain of possible m-adversaries.

Similar trend is found for the large group with higher number of records per provider and for *binary*, *direct*, and *bottom-up* algorithms. However, for the *top-down* algorithm runtime stays low despite the number of providers. This is due to its effective use of downward pruning. In our experiment the *top-down* algorithm runtime was very short and a no trend is recognized.

The Average Number of Records Per Provider. In this experiment, we systematically evaluated the impact of average number of records per provider ($|T_G|/n_G$) on the efficiency of the algorithms. **Figure 8** shows runtime with varying $|T_G|/n_G$ (n_G is constant while $|T_G|$ is being changed) for different heuristics. We observe that for groups with small average number of records per provider, both *direct* and *bottom-up* algorithms are very efficient as the group is likely to violate m-privacy. For groups with larger average number of records per provider, i.e. when $|T_G|/n_G \geqslant 15$, the *top-down* algorithm outperforms others.

Figure 8 also presents the runtime with varying average fitness score of contributing providers. It yields an almost identical trend as the result for average number of records per

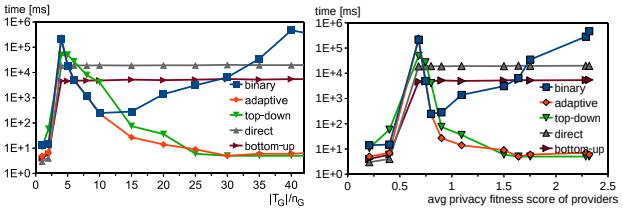

Fig. 8. Runtime vs. $|T_G|/n_G$ and average fitness score of providers.

provider. In fact, they are linearly correlated (squared correlation coefficient $R^2 = 0.97$, $score_F = 0.04 \cdot |T_G|/n_G + 0.33$) due to the definition of our privacy fitness score.

When a set of records is large, i.e. values of $|T_G|$ and $|T_G|/n_G$ are high, then its privacy fitness will be high as well. Greater values of the fitness score for a set of records indicates that its adversaries are less likely to breach privacy and the downward pruning is more likely to happen for a big set of adversaries. Applying pruning as early as possible significantly reduces computation time (**Figure 8**).

Adaptive Strategy. Based on the above results, we used the following parameters for the adaptive m-privacy checking strategy used in our anonymization experiments. If the average fitness score of contributing providers in a group is less than 0.85 ($|T_G|/n_G < 15$), we used the *binary* algorithm, while for other cases the *top-down* was our choice.

C. Anonymization for m-Privacy

This set of experiments compares our *provider-aware* algorithm with the *baseline* algorithm and evaluates the benefit of provider-aware partitioning as well as the adaptive m-privacy verification on utility of the data as well as efficiency. To evaluate the utility of the anonymized data, we used the query error metric similar to prior work (e.g. [18], [19]). 2,500 queries have been randomly generated and each query had qd predicates p_i, defining a range of a randomly chosen quasi-identifier, where $qd \in \left[2, \frac{q}{2}\right]$ and q is the number of quasi-identifier attributes.

$$\text{SELECT } t \text{ FROM } T^* \text{ WHERE } p_1 \text{ AND } \ldots \text{AND } p_{qd};$$

Query error is defined as the difference in the results coming from anonymized and original data.

Attack Power. We first evaluated and compared the two algorithms with varying attack power m. **Figure 9** shows the runtime with varying m for the two algorithms respectively. We observe that the *provider-aware* algorithm significantly outperforms the *baseline* algorithm. This fact may look counter intuitive at the first glance – our algorithm considers one more candidate splitting point at each iteration, thus the execution time should be higher. However, in each iteration of the *provider-aware* algorithm, the additional splitting point along data providers, if chosen, reduces the number of providers represented in a subgroup and hence reduces m-privacy verification time significantly (as observed in **Figure 7**). In contrast, the *baseline* algorithm preserves the average number of providers in each subgroup, which incurs a high cost for m-privacy verification. As expected, both algorithms show a peak cost when $m \approx n/2$.

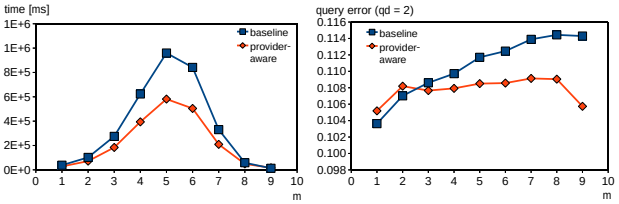

Fig. 9. Runtime (logarithmic scale) and the query error for different powers of m-privacy.

Figure 9 shows also the query error of the two algorithms with varying m. Intuitively, a higher attack power m should increase the query error as the data need to be generalized further to satisfy m-privacy. Our intuition is confirmed by the result of the *baseline* algorithm, but is disproved for the *provider-aware* algorithm. The constant values of the query error looks counter intuitive, but can be explained. The *baseline* algorithm, oblivious of the provider information, results in more generalized anonymized groups with increasing m. In contrast, the *provider-aware* algorithm, taking into account the data providers, will result in groups with smaller number of contributing providers (on average 1 for $k = 15$), hence can maintain a more precise view of the data and significantly outperforms the *baseline* algorithm. Thus, the query error may increase with m eventually, but it will not be as significant growth as for the *baseline* algorithm.

Number of Data Records. This set of experiments evaluates the impact of total number of records in the dataset. **Figure 10** shows the runtime and query error with varying number of records for both anonymization algorithms. As expected, the runtime for both algorithms grows with the number of records. However, the *baseline* algorithm has a higher growth rate than the *provider-aware* algorithm. This difference is due to the significantly reduced m-privacy verification time in our algorithm, which splits the data providers and thus reduces the number of providers represented in a group. In addition, the query error is at the same rate for both algorithms.

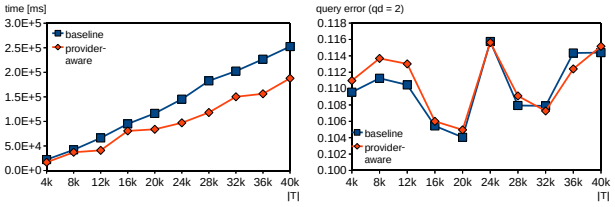

Fig. 10. Runtime and the query error vs. $|T|$.

Adaptive m-Privacy Verification. In this experiment, we evaluated the benefit of the adaptive selection of m-privacy verification algorithms. **Figure 11** compares the runtime of adaptive anonymization algorithm with two other m-privacy checking strategies with varying $|T|$ and constant n_G. For small values of $|T|$, the algorithm using adaptive verification strategy follows the *binary* and then the *top-down* algorithms, as we expected. However, for values of $|T| > 300$, our algorithm outperforms the non-adaptive strategies. The reason is that anonymization of a large number of records requires verification of m-privacy for many subgroups of different sizes. Adapting to such variety of groups is crucial for achieving high efficiency.

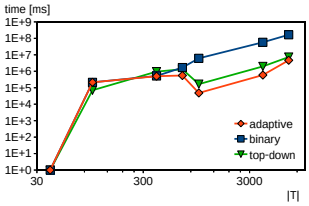

Fig. 11. Runtime of adaptive and non-adaptive m-privacy verifications vs. $|T|$ (log-log scale).

Impact of Privacy Constraints. We also performed a set of experiments evaluating the impact of the privacy constraints on the utility of data using anonymization algorithms for m-privacy. In our experiments, the constraint is defined as a conjunction of k-anonymity and l-diversity. **Figure 12** shows runtime and query errors with varying privacy constraint restrictiveness (varying k and l). Query error values are relative and dependent from selectiveness of queries. Query error values are different for different queries, but our algorithm will always have the same or lower error comparing to the baseline.

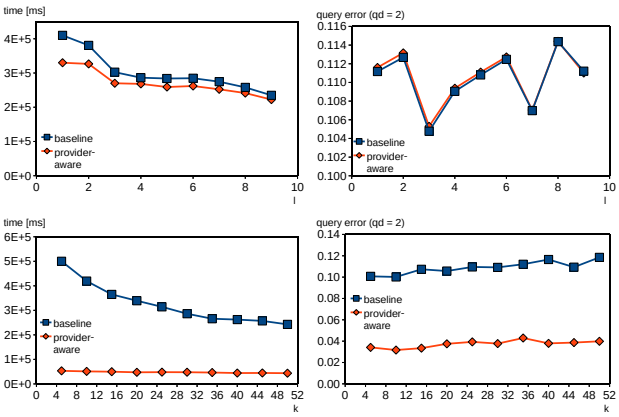

Fig. 12. Runtime and the query error vs. k in k-anonymity and l in l-diversity used in m-privacy.

As expected, increasing k causes more records in each equivalence group and higher query error. Varying l changes quality of anonymization results in an non-monotonic way that depends on the distribution of sensitive values in the dataset. However, execution times are shorter for decreasing k or l values because less partitions are created.

VI. Related Work

Privacy preserving data analysis and publishing has received considerable attention in recent years [1], [2], [3]. Most work has focused on a single data provider setting and considered the data recipient as an attacker. A large body of literature [2] assumes limited background knowledge of the attacker and defines privacy using relaxed *adversarial* notion [9] by considering specific types of attacks. Representative principles include k-anonymity [10], [11], l-diversity [9], and t-closeness [12]. Few recent works have modeled the instance level background knowledge as corruption and studied perturbation techniques

under these weak privacy notions [20]. In the distributed setting we studied, since each data holder knows its own records, the *corruption* of records is an inherent element in our attack model and is further complicated by the collusive power of the data providers. On the other hand, differential privacy [1], [3] is an unconditional privacy guarantee for statistical data release or data computations. While providing a desirable unconditional privacy guarantee, non-interactive data release with differential privacy remains an open problem. Many different anonymization algorithms have been introduced so far including Datafly [21], Incognito [22], and Mondrian [16]. In our research we considered the Mondrian algorithm as a baseline because its efficiency and extensibility.

There are some work focused on anonymization of distributed data. [5], [6], [23] studied distributed anonymization for vertically partitioned data using k-anonymity. Zhong et al. [24] studied classification on data collected from individual *data owners* (each record is contributed by one data owner) while maintaining k-anonymity. Jurczyk et al. [25] proposed a notion called l'-site-diversity to ensure anonymity for data providers in addition to privacy of the data subjects. Mironov et al. [26] studied SMC techniques to achieve differential privacy. Mohammed et al. [4] proposed SMC techniques for anonymizing distributed data using the notion of LKC-privacy to address high dimensional data. Our work is the first that considers data providers as potential attackers in the collaborative data publishing setting and explicitly models the inherent instance knowledge of the data providers as well as potential collusion between them for any weak privacy.

VII. Conclusions

In this paper, we considered a new type of potential attackers in collaborative data publishing – a coalition of data providers, called m-adversary. To prevent privacy disclosure by any m-adversary we showed that guaranteeing m-privacy is enough. We presented heuristic algorithms exploiting equivalence group monotonicity of privacy constraints and adaptive ordering techniques for efficiently checking m-privacy. We introduced also a *provider-aware* anonymization algorithm with adaptive m-privacy checking strategies to ensure high utility and m-privacy of anonymized data. Our experiments confirmed that our approach achieves better or comparable utility than existing algorithms while ensuring m-privacy efficiently.

There are many remaining research questions. Defining a proper privacy fitness score for different privacy constraints is one of them. It also remains a question to address and model the data knowledge of data providers when data are distributed in a vertical or ad-hoc fashion. It would be also interesting to verify if our methods can be adapted to other kinds of data such as set-valued data.

Acknowledgments

This work is supported by a Cisco Research Award. The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

[1] C. Dwork, "Differential privacy: a survey of results," in *Proc. of the 5th Intl. Conf. on Theory and Applications of Models of Computation*, 2008, pp. 1–19.

[2] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Comput. Surv.*, vol. 42, pp. 14:1–14:53, June 2010.

[3] C. Dwork, "A firm foundation for private data analysis," *Commun. ACM*, vol. 54, pp. 86–95, January 2011.

[4] N. Mohammed, B. C. M. Fung, P. C. K. Hung, and C. Lee, "Centralized and distributed anonymization for high-dimensional healthcare data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 4, pp. 18:1–18:33, October 2010.

[5] W. Jiang and C. Clifton, "Privacy-preserving distributed k-anonymity," in *Data and Applications Security XIX*, ser. Lecture Notes in Computer Science, 2005, vol. 3654, pp. 924–924.

[6] W. Jiang and C. Clifton, "A secure distributed framework for achieving k-anonymity," *VLDB J.*, vol. 15, no. 4, pp. 316–333, 2006.

[7] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004.

[8] Y. Lindell and B. Pinkas, "Secure multiparty computation for privacy-preserving data mining," *The Journal of Privacy and Confidentiality*, vol. 1, no. 1, pp. 59–98, 2009.

[9] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," in *ICDE*, 2006, p. 24.

[10] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE T. Knowl. Data En.*, vol. 13, no. 6, pp. 1010–1027, 2001.

[11] L. Sweeney, "k-anonymity: a model for protecting privacy," *Int. J. Uncertain. Fuzz.*, vol. 10, no. 5, pp. 557–570, 2002.

[12] N. Li and T. Li, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *In Proc. of IEEE 23rd Intl. Conf. on Data Engineering (ICDE)*, 2007.

[13] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik, "Identifying attack models for secure recommendation," in *In Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*, 2005.

[14] D. Kifer, "Attacks on privacy and definetti's theorem," in *Proc. of the 35th SIGMOD Intl. Conf. on Management of Data*, 2009, pp. 127–138.

[15] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *Proc. of the 2011 Intl. Conf. on Management of Data*, 2011, pp. 193–204.

[16] K. Lefevre, D. J. Dewitt, and R. Ramakrishnan, "Mondrian multidimensional k-anonymity," in *ICDE*, 2006.

[17] S. Goryczka, L. Xiong, and B. C. M. Fung, "m-privacy for collaborative data publishing," Emory University, Tech. Rep., 2011.

[18] X. Xiao and Y. Tao, "Anatomy: simple and effective privacy preservation," in *Proc. of the 32nd Intl. Conf. on Very Large Data Bases*, 2006, pp. 139–150.

[19] G. Cormode, D. Srivastava, N. Li, and T. Li, "Minimizing minimality and maximizing utility: analyzing method-based attacks on anonymized data," *Proc. VLDB Endow.*, vol. 3, Sept. 2010.

[20] Y. Tao, X. Xiao, J. Li, and D. Zhang, "On anti-corruption privacy preserving publication," in *Proc. of the 2008 IEEE 24th Intl. Conf. on Data Engineering*, 2008, pp. 725–734.

[21] L. Sweeney, "Datafly: A system for providing anonymity in medical data," in *Proc. of the IFIP TC11 WG11.3 Eleventh Intl. Conf. on Database Security XI: Status and Prospects*, 1998, pp. 356–381.

[22] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan, "Incognito: efficient full-domain k-anonymity," in *Proc. of the 2005 ACM SIGMOD Intl. Conf. on Management of Data*, 2005, pp. 49–60.

[23] N. Mohammed, B. C. M. Fung, K. Wang, and P. C. K. Hung, "Privacy-preserving data mashup," in *Proc. of the 12th Intl. Conf. on Extending Database Technology*, 2009, pp. 228–239.

[24] S. Zhong, Z. Yang, and R. N. Wright, "Privacy-enhancing k-anonymization of customer data," in *Proc. of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2005, pp. 139–147.

[25] P. Jurczyk and L. Xiong, "Distributed anonymization: Achieving privacy for both data subjects and data providers," in *DBSec*, 2009, pp. 191–207.

[26] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan, "Computational differential privacy," in *Advances in Cryptology – CRYPTO 2009*, ser. Lecture Notes in Computer Science, vol. 5677, 2009, pp. 126–142.