Secure Distributed Data Anonymization and Integration with *m*-Privacy

Slawomir Goryczka, Li Xiong, and Benjamin C. M. Fung

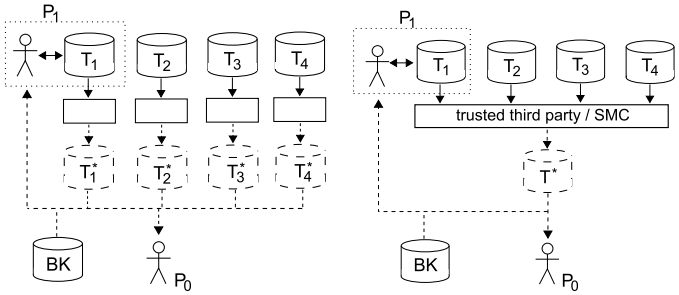
Abstract—In this paper, we study the *collaborative data publishing* problem for anonymizing horizontally partitioned data at multiple data providers. We consider a new type of "insider attack" by colluding data providers who may use their own data records (a subset of the overall data) to infer the 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 monotonicity of privacy constraints for efficiently checking *m*-privacy given a group of records. Third, 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. Finally, we propose secure multi-party computation protocols for collaborative data publishing with m-privacy. All protocols are extensively analyzed, and their security and efficiency are formally proved. Experiments on real-life datasets suggest that our approach achieves better or comparable utility and efficiency than existing and baseline algorithms, while satisfying *m*-privacy.

Index Terms—Privacy, security, integrity, and protection, distributed databases.

1 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 [2]–[4] have received considerable attention in recent years as promising approaches for sharing data while maintaining individual privacy. In a non-interactive model, a data provider (e.g., hospital) publishes a "sanitized" view of the data, simultaneously providing utility for data users (e.g., researchers), and privacy protection for the individuals represented in the data (e.g., patients). When data are gathered from multiple data providers or data owners, two main settings are used for anonymization [3], [5]. In one approach each provider anonymizes the data independently (anonymize-and-aggregate, **Fig. 1(a)**), which results in potential loss of integrated data utility. A more desirable approach is *collaborative data publishing* [3], [5]–[7], in which all providers anonymize data as if they would come from one source (aggregate-and-anonymize, **Fig. 1(b)**), using either a trusted third-party (TTP) or Secure Multi-party Computation (SMC) protocols [8], [9].



(a) Anonymize-and-aggregate. (b) Aggregate-and-anonymize.

Fig. 1. Distributed data publishing settings for four providers.

Problem Settings. We consider the collaborative data publishing setting (**Fig. 1(b)**) with horizontally distributed data across multiple data providers, each contributing a subset of records T_i. Each record has an owner, whose identity shall be protected. Each record attribute is either a sensitive attribute, which carries sensitive information about data owners, *an identifier*, which directly identifies the owner, or *a quasi-identifier* (QID), which may identify the owner if joined with a publicly known dataset. As a special case, a data provider could be the data owner itself who is contributing its own records. A data recipient may have access to some background knowledge (BK in **Fig. 1**), which represents any publicly available information about released data, e.g., Census datasets.

Our goal is to publish an anonymized view of the integrated data, T^*, which will be immune to attacks. Attacks are run by *attackers*, i.e., a single or a group (*a coalition*) of external or internal entities that wants to breach privacy of data using background knowledge, as well as anonymized data. Privacy is breached if one learns anything about data.

Existing Solutions. Collaborative data publishing can be considered as a multi-party computation problem, in which multiple providers wish to compute an anonymized view

- *A preliminary version of the manuscript has been published in [1].*
- *S. Goryczka and L. Xiong are with the Department of Mathematics and Computer Science, Emory University, Atlanta, GA, USA. E-mail: sgorycz@emory.edu, lxiong@emory.edu*
- *Benjamin C. M. Fung is with Concordia University, Montreal, QC, Canada. E-mail: fung@ciise.concordia.ca*

1. http://healthit.hhs.gov/nhin/

of their data without disclosing any private and sensitive information. We assume the data providers are semi-honest [8], [9], which is commonly assumed in distributed computations. A trusted third party (TTP) or Secure Multi-Party Computation (SMC) protocols [6] can be used to guarantee lack of *intermediate* information disclosure *during* the anonymization. However, neither TTP nor SMC protects against inferring information from the anonymized data.

The problem of inferring information from anonymized data has been widely studied in a single data provider settings [3]. A data recipient that is an attacker, e.g., P_0, attempts to infer additional information about data records using the published data, T^*, and background knowledge, BK. For example, k-anonymity [10], [11] protects against identity disclosure attacks by requiring each quasi-identifier equivalence group (QI group) to contain at least k records. l-Diversity requires each QI group to contain at least l "well-represented" sensitive values [12]. Differential privacy [2], [4] guarantees that the presence of a record cannot be inferred from a statistical data release, while assuming very little about background knowledge of attackers.

New Challenges. Collaborative data publishing introduces a new attack that has not been studied so far. Each data provider, such as P_1 in **Fig. 1**, can use both, anonymized data T^*, and its own data T_1 to infer additional information about other records. Compared to the attack by the external recipient in the second scenario, each provider has additional data knowledge of its 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 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 illustrate the m-adversary threats with an example shown in **Table 1**. 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. Note that one record, owned by Olga, is contributed by two providers P_2 and P_4, and is represented as a single record in anonymized dataset. T_a^* is one possible anonymization that guarantees k-anonymity and l-diversity ($k = 2$, $l = 2$), i.e., each QI group contains records with at least l different sensitive values. However, an attacker from the hospital P_1 may remove all records from P_1. In the first QI group there will be only one remaining record, which belongs to a patient between 20 and 30 years old. By using quasi-identifier attributes to join this record with the background knowledge BK (e.g., part of the Census database), P_1 can identify Sara as its owner (highlighted in the table) and her disease Epilepsy. In practice, the attacker would use more attributes as a QI and maximal BK to mount the linking attack [14]. 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 hospitals, and using information about her former patients.

TABLE 1
m-Adversary and *m*-privacy example.

T_1

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

T_2

Name	Age	Zip	Disease
Olga	32	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	98701	Cancer
Frank	33	12388	Asthma

T_a^*

Providers	*Name*	Age	Zip	Disease
P_1	*Alice*	[20-30]	*****	Cancer
P_1	*Emily*	[20-30]	*****	Asthma
P_3	***Sara***	**[20-30]**	*****	**Epilepsy**
P_2	*John*	[31-34]	*****	Flu
P_2, P_4	*Olga*	[31-34]	*****	Cancer
P_4	*Frank*	[31-34]	*****	Asthma
P_1	*Bob*	[35-40]	*****	Epilepsy
P_2	*Mark*	[35-40]	*****	Flu
P_3	*Cecilia*	[35-40]	*****	Flu

T_b^*

Providers	*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, P_4	*Olga*	[20-40]	987**	Cancer
P_3	*Cecilia*	[20-40]	987**	Flu
P_1	*Bob*	[20-40]	123**	Epilepsy
P_4	*Frank*	[20-40]	123**	Asthma
P_2	*John*	[20-40]	123**	Flu

Contributions. We define and address this new type of "insider attack" by an m-*adversary*, i.e., a coalition of m colluding data providers or data owners that attempt to infer data records contributed by others. Note that a 0-adversary models 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 considered when the data are anonymized.

We address the new threat introduced 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, in **Table 1** T_b^* is an anonymized table that satisfies m-privacy ($m = 1$) with respect to k-anonymity and l-diversity ($k = 2$, $l = 2$).

Second, for scenarios with a TTP, to address the challenges of checking a combinatorial number of potential m-adversaries, we present heuristic algorithms for efficient m-privacy verification 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. We also present a data *provider-aware* anonymization algorithm with adaptive strategies of checking m-privacy fulfillment, to ensure high utility and m-privacy of sanitized data with efficiency.

Compared to our preliminary version [1], our new con-

tributions extend above results. First, we adapt privacy verification and anonymization mechanisms to work for m-privacy w.r.t. to any privacy constraint, including non-monotonic ones. We list all necessary privacy checks, and prove that no fewer checks is enough to confirm m-privacy.

Second, we propose SMC protocols for secure m-privacy verification and anonymization. We prove their security, complexity and experimentally confirm their efficiency.

2 *m*-PRIVACY DEFINITION

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

Let $T = \{t_1, t_2, \ldots\}$ be a set of records with the same attributes gathered from n data providers $P = \{P_1, P_2, \ldots, P_n\}$, such that $T_i \subseteq T$ are records provided by P_i. Let A_S be a sensitive attribute with a domain D_S.

If the records contain multiple sensitive attributes then, we could treat each of them as the sole sensitive attribute, while others would be included to the quasi-identifier [12]. However, in our scenarios we use an approach, which preserves more utility without sacrificing privacy [15].

Our goal is to publish an anonymized table T^* while preventing any m-adversary from inferring A_S for any single record. An m-adversary is a coalition of data users with m data providers cooperating to breach privacy of anonymized records.

2.1 *m*-Privacy

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

We now formally define a notion of m-privacy with respect to a privacy constraint C, to protect the anonymized data against m-adversaries. The notion explicitly models the inherent data knowledge of an m-adversary, the data records they jointly contribute, and requires that each QI 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 contribute a set of records T_I. $\mathcal{A}(T)$ satisfies m-privacy with respect to a privacy constraint C if and only if, any anonymized superset of records $\mathcal{A}(T')$ from non-m-adversary providers satisfies C, i.e.,

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

Corollary 2.1: For all $m \leqslant n-1$, if $\mathcal{A}(T)$ is m-private, then it is also $(m-1)$-private. If $\mathcal{A}(T)$ is not m-private, then it is also not $(m+1)$-private.

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

***m*-Privacy with Duplicate Records.** m-Privacy can be also guaranteed when there are duplicate records (such as records from a patient transferred between hospitals). In our initial example Olga has records in two hospitals P_2 and P_4 (**Table 1**). For such cases, the duplicates are treated as a single record shared by a few providers. If any of the providers is a member of an m-adversary, the record will be considered as a part of its background knowledge.

***m*-Privacy and Syntactic Privacy Constraints.** Let C be a syntactic privacy constraint, i.e., a constraint that preserves data truthfulness at the record level, e.g., k-anonymity, l-diversity, and t-closeness [16]. T^* satisfying C will only guarantee 0-privacy w.r.t. C, i.e., C is not guaranteed to hold for every QI group after excluding records belonging to any data provider. m-Privacy is defined w.r.t. a privacy constraint C, and hence will inherit all strengths and weaknesses of C. m-Privacy w.r.t. C protects against privacy attacks issued by any m-adversary if and only if, C protects against the same attacks by an external data recipient. m-Privacy notion is orthogonal to the privacy constraint C being used, and enhances privacy it defines to distributed settings, where up to m data providers collude.

***m*-Privacy and Differential Privacy.** Differential privacy [2], [4], [17] guarantees privacy even if an attacker knows all but one record. Thus, any differentially private mechanism is $(n-1)$-private w.r.t. differential privacy, which is the maximum level of m-privacy, i.e., any $(n-1)$ colluding providers cannot breach privacy of records. However, differential privacy does not preserve data truthfulness at the record level, and hence cannot be used for some scenarios, e.g., by a pharmaceutical company that analyzes anonymized patient records to choose a small group of individual patients for clinical trials.

Opposite to differential privacy, m-privacy w.r.t. a syntactic privacy notion preserves data truthfulness at the record level. In the remaining of the paper, we will focus on checking and achieving m-privacy w.r.t. different syntactic privacy constraints.

2.2 Monotonicity of Privacy Constraints

Monotonicity of privacy constraints is defined for a single equivalence group of records, i.e., a group of records that QI attributes share the same generalized values. Let $\mathcal{A}_1$ be a mechanism that anonymizes a group of records T into a single equivalence group, $T^* = \mathcal{A}_1(T)$.

Generalization based monotonicity of privacy constraints has been already defined in the literature (Definition 2.2) [12], [16]. Its fulfillment is crucial for designing efficient generalization algorithms [11], [12], [16], [18]. In this paper we will refer to it as *generalization monotonicity*.

Definition 2.2: (GENERALIZATION MONOTONICITY OF A PRIVACY CONSTRAINT [12], [16]) A privacy constraint C is generalization monotonic if and only if, for any two

equivalence groups $\mathcal{A}_1(T)$ and $\mathcal{A}_1(T')$ that satisfy C, their union satisfies C as well,

$$\begin{array}{l} C(\mathcal{A}_1(T)) = true \\ C(\mathcal{A}_1(T')) = true \end{array} \Rightarrow C(\mathcal{A}_1(T) \cup \mathcal{A}_1(T')) = true$$

In the definition of generalization monotonicity there is an assumption that original records have been already anonymized into equivalence groups, which are used for further generalizations. In this paper, we introduce more general and record-based definition of monotonicity in order to facilitate the analysis, and design efficient algorithms for verifying m-privacy w.r.t. C.

Definition 2.3: (EQUIVALENCE GROUP MONOTONICITY OF A PRIVACY CONSTRAINT, EG MONOTONICITY) A privacy constraint C is EG monotonic if and only if, for a group of records T such that its equivalence group $\mathcal{A}_1(T)$ satisfies C, and any group of records $\widetilde{T}$, their anonymized union satisfies C,

$$C(\mathcal{A}_1(T)) = true \Rightarrow \forall \widetilde{T},\ C(\mathcal{A}_1(T \cup \widetilde{T})) = true$$

EG monotonicity is more general 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, which requires l distinct values of sensitive attribute in a QI group, are examples of EG and generalization monotonic constraints. Entropy l-diversity [12] and t-closeness [16] are examples of generalization monotonic, but not EG monotonic constraints 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., each record has different sensitive value. Entropy l-diversity is not EG monotonic, because it will not hold if we add records that change the entropy of sensitive values significantly. However, it is generalization monotonic because it will still hold if two QI groups satisfying entropy l-diversity ($l = 2$) are (generalized) into a new group.

Corollary 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 [12], C is EG monotonic as well.

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

This theorem holds also when applied for generalization monotonicity. Proofs of this theorem for both EG and generalization monotonicities defined with respect to records and not m can be found in the Appendix A.

Corollary 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 such that only $T' = T \setminus T_I$ are checked, i.e.,

$$\forall I \subset P, |I| = m, C(\mathcal{A}(T \setminus T_I)) = true$$

Indeed, if $\mathcal{A}(T \setminus T_I)$ satisfies C, then EG monotonicity of C guarantees that any anonymized superset of $T \setminus T_I$ satisfies C as well. Thus, $\mathcal{A}(T)$ fulfills definition of m-privacy w.r.t. C. In addition, if a coalition I cannot breach privacy, then any its sub-coalition with fewer records cannot do so either (Definition 2.3). Unfortunately, generalization monotonicity of C is not enough to guarantee this property.

3 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. In this section, we first analyze the problem by modeling the adversary space. Then, we present heuristic algorithms with effective pruning strategies and adaptive ordering techniques for efficiently checking m-privacy w.r.t. an EG monotonic constraint C. Implementation of introduced algorithms can be run by a trusted third party (TTP). For scenarios without such party, we introduce secure multi-party (SMC) protocols. Finally, in Appendix B.1 we present modifications of TTP heuristics and SMC protocols to verify m-privacy w.r.t. non-EG monotonic privacy constraints.

3.1 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 **Fig. 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 given by $\binom{n_G}{m}$. Each node has parents (children) representing their direct super- (sub-) coalitions. For simplicity the space is depicted as a *diamond*, where a horizontal line at a level m corresponds to all m-adversaries, the bottom node to 0-adversary (external data recipient), and the top line to $(n_G - 1)$-adversaries.

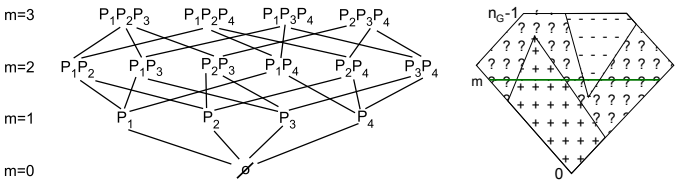


Fig. 2. *m*-Adversary space and pruning strategies upward (+), and downward (-).

In order to verify m-privacy w.r.t. a constraint C for a set of records, we need to check fulfillment of C for all records after excluding any possible subset of m-adversary records. When C is EG monotonic, we only need to check C for the records excluding all records from any m-adversary (Observation 2.3), i.e., adversaries 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. In the worst-case scenario, when $m = n_G/2$, the number of checks is equal to the central binomial coefficient $\binom{n_G}{n_G/2} = O(2^{n_G} n_G^{-1/2})$. Thus, the *direct* algorithm is not efficient enough.

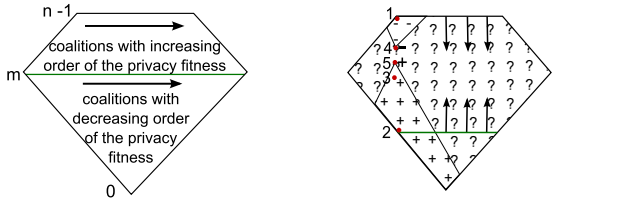
3.2 Heuristic Algorithms for EG Monotonic Constraints

In this section, we present heuristic algorithms for efficiently checking m-privacy w.r.t. an EG monotonic constraint. Then, we modify them to check m-privacy w.r.t. a non-EG monotonic constraint.

The key idea of our heuristics for EG monotonic privacy constraints 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 is possible thanks to the EG monotonicity of m-privacy (Observations 2.1, and 2.3). If a coalition is not able to breach privacy, then all its sub-coalitions will not be able to do so as well, 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 as well, 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*). **Fig. 2** 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 (**Fig. 3(a)**). This is motivated by following observations. For downward pruning, super-coalitions of m-adversaries with limited attack powers are preferred to be checked first, as they are less likely to breach privacy, and hence increase the chance of downward pruning. In contrast, sub-coalitions of m-adversaries with significant attack powers are preferred to be checked first, as they are more likely to breach privacy, and hence increase the chance of the early stop.



(a) Adaptive ordering.

(b) First steps of the *binary* algorithm with verified coalitions depicted as numbered red dots.

Fig. 3. Adaptive ordering for efficient pruning and an example run of the *binary* algorithm.

To quantify privacy fulfillment by a set of records, which is used to measure the attack power of a coalition and privacy of remaining records, we introduce a privacy fitness score w.r.t. C. It also used to facilitate the anonymization, which we will discuss in the following section.

Definition 3.1: (PRIVACY FITNESS SCORE) Privacy fitness F_C for a set of anonymized records T^* is a level of 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 a conjunction of k-anonymity and l-diversity. The privacy fitness score is defined as the minimum fitness score of privacy constraints. In our example $score_{F_C}$ is defined as follows:

$$score_{F_C}(T^*) = \min\left\{\frac{|T^*|}{k}, \frac{|\{t[A_S] : t \in T^*\}|}{l}\right\} \quad (1)$$

Notice that $score_{F_C}(T^*) \geqslant 1$, if and only if $C(T^*) = true$.

The privacy fitness score also quantifies the attack power of attackers. The higher their privacy fitness scores are, the more likely they are able to breach the privacy of the remaining 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) (**Fig. 3(a)**).

Now we present several heuristic algorithms that use different search strategies, and hence utilize different pruning directions. 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 is similar to the *top-down* algorithm. The main difference is in the sequence of coalition checks, which is in a bottom up fashion starting from 0-adversary, and moving up. The algorithm stops if a violation by any adversary is detected (early stop) or all m-adversaries are checked.

The Binary Algorithm. The *binary* algorithm (Algorithm 1), inspired by the binary search algorithm, checks coalitions between $(n_G - 1)$-adversaries and m-adversaries, and takes advantage of both pruning strategies (**Fig. 3(b)**). Thanks to EG monotonicity of the privacy constraint, we do not consider coalitions of less than m adversaries (Corollary 2.3).

The goal of each iteration in the algorithm is to search for a pair of coalitions 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.

The search works as follows. First, it starts with $(n_G - 1)$-adversaries, finds the first coalition of attackers that violates privacy, and assigns it to I_{super} (lines from 4 to 7). Then, it finds an I_{sub}, i.e., a sub-coalition of I_{super}, which does not breach privacy (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 can breach privacy, then I_{super} is updated to I (line 14). Otherwise, I_{sub} is updated to I (line 16). The algorithm continues until a direct parent-child pair I_{super} and I_{sub} is found (line 11). Then pruning

Algorithm 1: The *binary* m-privacy verification algorithm.

Data: Anonymized records T^* from providers P, an EG monotonic C, a fitness scoring function $score_F$, and the m.
Result: *true* if T^* is m-private w.r.t. C, *false* otherwise.

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

in both directions is performed (lines 17 and 18), and the algorithm starts the next iteration. The algorithm stops when m-privacy can be determined (line 3).

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 also the level of privacy fulfillment of the group, may be used to select the most suitable algorithm. The higher the fitness score, the more likely m-privacy will be satisfied, and hence the *top-down* algorithm with downward pruning will significantly reduce the number of adversary checks. We utilize such strategy in the anonymization algorithm (discussed later), and experimentally evaluate it.

3.3 Time Complexity

In this section, we derive the time complexity for the m-privacy w.r.t. C verification algorithms in terms of the number of privacy checks. Since all algorithms involve multiple checks of privacy for various records, we assume that each check of C takes a constant time. Formally, it can be modeled by an oracle, which performs a check for given records in $O(1)$ time. For a particular definition of C, time complexity of a single privacy verification should be also taken into account. Details of time complexity computations can be found in the Appendix E.

EG Monotonic *m*-Privacy. All the above verification algorithms have the same worst-case scenario, 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. For these settings, the *direct* algorithm will check exactly $\binom{n_G}{m}$ possible 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. 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 first, then smaller coalitions up to all m-adversaries, which requires $\sum_{i=n_G-1}^{m}\binom{n_G}{i} = O\left(n_G^{n_G-1-m}\right)$ checks. The *binary* algorithm will run $\binom{n_G}{m}$ iterations with $O(\log(n_G-m))$ privacy checks in every iteration. Thus, the total time complexity is equal to $O\left(n_G^m \log(n_G-m)\right)$.

The average time complexity analysis is more involved, and its results depend on the parameter m. For each algorithm the lower bound of the average time complexity is $O(n_G)$, but the upper bound varies, and is equal to $O\left((3/2)^{n_G}\right)$ for the *top-down*, $O\left(2^{n_G} n_G^{-1/2}\right)$ for the *bottom-up*, $O\left(2^{n_G} n_G^{-1}\right)$ for the *direct*, and $O\left(2^{n_G} n_G^{-1} \log_2 n_G\right)$ for the *binary*. Thus, adapting verification strategy to different settings is crucial to achieve, on average, a low runtime.

4 Secure *m*-Privacy Verification Protocols

All the above algorithms can be run by a trusted third-party (TTP). For settings without such a party, data providers need to run an SMC protocol. We assume that all providers are semi-honest, i.e., honest but curious. In this section we present secure protocols to verify m-privacy w.r.t. EG monotonic constraint C.

A secure m-privacy verification protocol for a non-EG monotonic constraint is an extension of the *bottom-up* approach. Due to space limit details of such protocol were moved to Appendix D.2.

Note that the TTP can recognize duplicated records, and treats them in the appropriate way. For SMC protocols all records are unique, and duplicates are not detected.

Preliminaries. Our SMC protocols are based on Shamir's secret sharing [19], encryption, and other secure schemas. In a secret sharing scheme, the owner of a secret message s prepares, and distributes n_G shares, such that each party gets a few shares (usually one). We use $[s]$ to denote the vector of shares, and $[s]_i$ to refer to an i^{th} share sent to P_i. An algorithm reconstructing s requires any r shares as its input. To prevent any coalition of up to m providers to reveal intermediate results, we set $r = m+1$.

Note that receivers of shares do not have to be providers nor trusted. They could be run as separate processes in a distributed environment, e.g., cloud, and still computations would stay information-theoretically secure [20]. In our implementation and complexity analyzes, we have used SEPIA framework [21].

Secure Subprotocols. To compute sums we run a secure sum protocol, which securely computes the sum of numbers held by providers. Implementation of such protocol is based on Shamir's secret sharing scheme, and has been introduced in SEPIA framework [21]. Another protocol that is provided by SEPIA is secure comparison, which securely compares two numbers. By running this protocol for a set of numbers, we find the minimum and maximum values in the set, and set its elements in order.

In our protocols we also use secure size of set union subprotocol, which is a slight modification of the secure size of set intersection protocol [22]. The modification is to count all distinct encrypted items, and not only ones that are contributed by every provider.

Correctness, security, and complexity of these protocols and their implementations have been proven in [21], [22].

Secure Leader Election Protocol. All protocols are initiated by a leader P', i.e., a chosen provider, which is found by running a secure leader election protocol (SLE). Our SLE protocol (Algorithm 9 in Appendix D.1) runs a secret sum protocol over randomly generated numbers in order to elect the leader. The implementation utilizes Shamir's secret sharing scheme, and does not disclose any information about data and its providers. The leader is considered untrusted, therefore any honest but curious party (also external) can participate in the election. Each data provider can simulate, monitor, and verify the leader actions to detect any malicious behavior.

4.1 Secure EG Monotonic *m*-Privacy Verification

Assume that a group of data records is horizontally distributed among n_G data providers. They would like to securely verify, if anonymization of their records into one QI group, is m-private w.r.t. C. Additionally, assume that verification of privacy defined by C is given (described below), and all providers have already elected a leader P'. Before verifying m-privacy the leader securely sorts data providers.

Secure Sorting and Adaptive Ordering. The main responsibility of the leader is to determine m-privacy fulfillment with as little privacy checks as possible. Our heuristic minimizes the number of privacy checks by utilizing EG monotonicity of C and adaptive ordering of m-adversary generation (Section 3.2). To define such order, P' runs any sorting algorithm, which sorts providers by fitness scores of their local records, with all comparisons run securely.

Applying the adaptive ordering heuristic uncovers the order of fitness scores of data providers. Without such ordering more privacy checks need to be performed.

Our implementations of secure sorting protocol utilizes the Shamir's secret sharing scheme with r shares required to reconstruct a secret. To ensure m-privacy we set $r = m + 1$. Thus, for n_G data providers the protocol requires running a sorting algorithm, which takes $O(n_G \log n_G)$ secure comparisons. Each secure comparison has the same complexity, i.e., requires a few secure multiplications, where each multiplication takes $O(m^2)$ time [21]. Thus, the secure sorting time complexity is equal to $O(m^2 n_G \log n_G)$. Each secure multiplication requires passing $n_G(n_G - 1)$ messages in total, although only $(m+1)^2$ of them are needed to get the result. Thus, the communication complexity is equal to $O(n_G^3 \log n_G)$.

Note that if we allow disclosing fitness score values from all providers, then all complexities can be significantly reduced to $O(n_G \log n_G)$ for time complexity, and $O(n_G)$ for communication complexity.

Secure *m*-Privacy Verification Protocol. After finding the order of data providers, the leader P' starts verifying privacy for different coalitions of attackers, which are generated in specific order. A general scheme of secure m-privacy verification is the same for all heuristic algorithms (Algorithm 2). Common steps are as follows. In the main loop P' verifies privacy of records for m-adversaries until m-privacy can be decided (line 3). Note that in order to determine m-privacy w.r.t. EG monotonic C, it is enough to check privacy for all scenarios with exactly m attackers (Corollary 2.3). In the loop, P' generates, and broadcasts a coalition of potential adversaries I, so each party can recognize its status (attacker/non-attacker) for the current privacy check. Then, the leader runs the secure privacy verification protocol for I (line 6). If privacy could be breached, and I has no more than m data providers, then the protocol stops, and returns negative answer (line 7). Otherwise, the information about privacy fulfillment is used to prune (upwards or downwards) a few potential m-adversaries (line 9). Finally, if m-privacy w.r.t. C can be decided, P' returns the result of the verification (line 10).

Algorithm 2: Secure m-privacy verification protocol w.r.t. EG monotonic constraint C for *top-down*, *bottom-up*, and *direct* algorithms; code run by the leading provider P'.

```
   Data: List of providers P, an EG monotonic C, and the m.
   Result: true if A_1(T) is m-private w.r.t. C, false otherwise.
1  sites = securely_sort_providers(P, increasing_order, score_F)
2  use_adaptive_order_generator(sites, m)
3  while is_m-privacy_decided() == false do
4      I = generate_next_coalition(P)
5      Broadcast coalition I.
       // Runs secure privacy verification protocol.
6      privacy_breached = is_privacy_breached_by(I)
7      if privacy_breached and |I| ⩽ m then
8          return false // early stop
9      prune_coalitions(I, privacy_breached)
10 return is_m-private()
```

For the *binary* algorithm, secure m-privacy verification protocol is also run by P', which executes all steps of the Algorithm 1. The only difference is privacy verification, which is implemented as an SMC protocol. Due to lack of space details of this protocol are skipped.

Proposition 4.1: Assuming security of subprotocols, all m-privacy protocols are secure except revealing results of potential attacks of generated m-adversaries.

Proof: Results of all privacy checks are publicly known, and, by applying pruning, one can determine privacy of records for a few potential m-adversaries. Thus, the security disclosure depends on data, and the sequence of generated m-adversaries I is very important to minimize it. In this proof, we analyze security for all heuristics that are presented above (Section 3.2).

All generated m-adversaries can be partitioned into two groups by the result of privacy check: 1) the m-adversary, and all its subsets, cannot breach privacy of remaining records, 2) the m-adversary, and all its supersets, can breach privacy of remaining records.

If the records are m-private w.r.t. C, then *direct* and *bottom-up* algorithms make the verification protocol fully

secure. Fulfillment of m-privacy implies that all verified coalitions have size up to m, and are in the group 1), i.e., there is no security breach. On the contrary, both *top-down* and *binary* algorithms consider coalitions of more than m providers from both groups. Coalitions from group 1) can have any size, but all coalitions from group 2) contain more than m providers. Thus, these two algorithms disclose both positive and negative results of possible attacks from coalitions of different size.

If the records are not m-private w.r.t. C, i.e., there is an m-adversary that can breach privacy, perfect security of the protocol cannot be guaranteed. Due to pruning property all heuristics reveal information about all coalitions from group 1), and about a single coalition of size up to m from group 2). In addition, *top-down* and *binary* algorithms reveal also results of privacy checks for coalitions from group 2) having more than m providers. ■

Note that for a potential attacker, finding a coalition that is able to breach privacy, is more important than finding a coalition that cannot do so. Thus, both *direct* and *bottom-up* algorithms are more secure than others. Among them *bottom-up* have more chances to identify the smallest coalition that is able to breach privacy. Thus, *direct* is our choice for maximum privacy scenarios. For other settings, our anonymization algorithm adaptively chooses the verification algorithm.

Computation Complexity. Electing the leader is a separate task, which can be run once for all privacy verifications. Its time complexity is equal to $O(mn_G)$.

In Algorithm 2, a single loop iteration executes following operations: generating next coalition of attackers ($O(\log n_G)$), broadcasting generated coalition ($O(\log n_G)$), verifying if m-privacy can be determined ($O(n_G)$), and pruning ($O(n_G)$). Among them privacy verification has the highest complexity. Assuming that its time complexity is equal to $\mathcal{V}$ (computed below), and complexity of a single verification loop is equal to $V = \mathcal{V} + O(n_G)$. The *direct* algorithm will check privacy for at most $\binom{n_G}{m}$ possible m-adversaries. Thus, the complexity of m-privacy verification is equal to $O\left(V \cdot n_G^m\right)$. 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, therefore for this case complexity is equal to $O\left(V \cdot n_G^m\right)$. The *top-down* algorithm will check all $(n_G - 1)$-adversaries first, then smaller coalitions down to all m-adversaries, which requires $\sum_{i=n_G-1}^{m}\binom{n_G}{i} = O\left(n_G^{n_G-1-m}\right)$ checks, and the overall complexity of the protocol is equal to $O\left(V \cdot n_G^{n_G-1-m}\right)$. The *binary* algorithm will run $\binom{n_G}{m}$ iterations with $O(\log(n_G - m))$ privacy checks in each of them. Thus, when used, the protocol time complexity is equal to $O\left(V \cdot n_G^m \log(n_G - m)\right)$.

Communication Complexity. During each loop iteration of the m-privacy verification protocol (Algorithm 2) the leader sends $(n_G - 1)$ messages to providers with information if they should act as attackers or not. Assume that $\mathcal{V}_C$ is a communication complexity for a privacy verification protocol (computed below), and $V_C = \mathcal{V}_C + n_G - 1$ is the total communication. Thus, the total communication complexities depend on the number of privacy checks, which is different for each algorithm, i.e., *direct*, $O\left(V_C \cdot n_G^m\right)$; *bottom-up*, $O\left(V_C \cdot n_G^m\right)$; *top-down*, $O\left(V_C \cdot n_G^{n_G-1-m}\right)$; and *binary*, $O\left(V_C \cdot n_G^m \log(n_G - m)\right)$.

4.2 Secure Privacy Constraint Verification

To allow using any privacy constraint in our m-privacy verification protocol, secure privacy verification is implemented as a separate protocol, and results of its runs are disclosed. Presenting verification protocols for any privacy constraint is out of the scope of this paper, but we present secure protocols to verify k-anonymity and l-diversity. All implementations use Shamir's secret sharing [19] as their main scheme. For a few subprotocols we use encryption (commutative, homomorphic, etc.), and other secure schemas for efficiency. Assume that there are n_G data providers, and each data provider P_i provides T_i records.

Secure k-Anonymity Verification. To securely verify k-anonymity, the leader counts all records $s = |T|$ using the secure sum protocol [22]–[24], and securely compares s with k. Our implementation of the secure sum protocol uses only Shamir's secret sharing scheme (Algorithm 3).

First, all data providers run secure sum protocol in order to compute total number of records s. To avoid disclosing s it is stored in distributed shares $[s]$ (line 1). Finally, all providers securely compare $[s]$ with k [21]. As the result, each provider gets a share of 1 if k-anonymity holds or a share of 0 otherwise (line 2).

Algorithm 3: The secure k-anonymity verification protocol.

Data: $P_1, \ldots, P_{n_G}$ providing $T_1, \ldots, T_{n_G}$ records respectively.
Result: Each P_i gets $[1]_i$ if $s \geqslant k$, $[0]_i$ otherwise.
1 $[s] = secureSum\left(|T_1|, \ldots, |T_{n_G}|\right)$
2 **return** $1 - lessThan([s], k)$

Theorem 4.1: Assuming security of subprotocols, the k-anonymity verification protocol is secure against at most m attackers.

Proof: Assuming secure communication channels, the Shamir's secret sharing scheme were proven correct and information-theoretically secure [20]. Thus, knowing up to m shares of any value does not disclose it. Correctness and security of both $secureSum$ and $lessThan$ subprotocols were proven in [21]. The protocol does not reveal anything, but the result of the comparison $s \geqslant k$. ■

Complexity Analysis. Computation complexity of the protocol is equal to the sum of complexities for both subprotocols. In [21] complexities are given as functions of secure multiplications. Each secure multiplication requires additional shares generation, and secret reconstruction, which take $O(mn_G)$ time. Assuming that number of bits used to represent a number in our protocols is constant, secure comparison protocol requires constant number of multiplications, i.e., its time complexity is $O(mn_G)$. Secure sum protocol (including shares generation) has the same complexity. Thus, the overall time complexity is $O(mn_G)$.

While running the $secureSum$ subprotocol $n_G(n_G-1)$ messages are sent. Additionally, the $lessThan$ subprotocol requires constant number of multiplications, therefore total number of messages is equal to $n_G(n_G-1)$. Thus, the total communication complexity is equal to $O(n_G^2)$.

Secure *l*-Diversity Verification. The goal of this protocol is to securely verify if the total number of sensitive values from all records, is at least l (Algorithm 4). The protocol has two phases. In the first phase, each data provider P_i finds the set of sensitive values S_i of its records. Then, it randomly generates p_i fake values, and adds them to S_i (line 1). Note that each provider generates fake values from a different domain. In the last step of this phase, the leader runs the secure size of set union subprotocol to compute $\bar{s}$, i.e., the size of the set of sensitive values of all records with a few additional fake values (line 2). The subprotocol is run in the same way as the secure size of set intersection [22], [25] with a few minor modifications. Note that the use of commutative encryption scheme in the subprotocols ensures that duplicated sensitive values are properly handled.

In the second phase, all providers securely compute the number all fake values (line 3). Then, they securely check if the number of sensitive values is not less than l, i.e, if $\bar{s} - [p] \geqslant l$. The results are stored by providers as shares of 1 if l-diversity holds or shares of 0 otherwise (line 4).

Algorithm 4: The secure l-diversity verification protocol.

Data: Each P_i has records T_i.
Result: Each P_i gets $[1]_i$ if $|\bigcup_{i=1}^{n_G} S_i| \geqslant l$, $[0]_i$ otherwise.
1 $S_i = \{t[A_s] : t \in T_i\} \cup \texttt{generate\_fake\_values}(p_i)$
2 $\bar{s} = secureSizeOfUnion\,(S_1, \ldots, S_{n_G})$
3 $[p] = secureSum\,(p_1, \ldots, p_{n_G})$
4 **return** $1 - lessThan(\bar{s} - l, [p])$

Theorem 4.2: Assuming security of subprotocols, the l-diversity verification protocol is secure against up to m attackers except an upper bound of the number of sensitive values.

Proof: Using commutative encryption scheme in implementation of the $secureSizeOfUnion$ subprotocol guarantees its correctness and security. Adding distinct fake values ensures that the local number of sensitive values will not be disclosed. Since each data provider generates different fake values, the sum of their counts is equal to the count of their union. The only information that is revealed, is $\bar{s}$, i.e., the upper bound of the number of sensitive values. However, allowing large and random number of fake values guarantees the low probability of guessing the real number of sensitive values. The second phase of the protocol utilizes Shamir's secret sharing scheme for secure sum, and comparison subprotocols, which are secure [20], [21]. Thus, the protocol is also secure. ■

Complexity Analysis. The first steps of the protocol requires $2n_G$ rounds of communication, and encryptions. Thus, if there are at most d_S sensitive values, and up to p_S fake values, the time complexity is equal to $O(n_G(d_S+p_S))$. Time complexity of the secure sum protocol implemented using secret sharing scheme is equal to $O(mn_G)$.

While computing $\bar{s}$ all providers exchange $2n_G$ messages. Both $secureSum$ and $lessThan$ protocols generate $2n_G(n_G-1)$ messages, and the overall communication complexity is equal to $O(n_G^2)$.

Secure Privacy Verification. Above protocols return the verification result as shares of $[1]$ if privacy constraint is fulfilled, and $[0]$ otherwise. Each provider holds a single share for a constraint C_i. Any $r = m+1$ providers are able to check if $C = C_1 \wedge \ldots \wedge C_w$ holds, by securely multiplying their results for all constraints, and comparing it with zero [21]. If the final reconstructed value is equal to 1, then C holds, otherwise it does not.

The fulfillment of each privacy constraint is kept secret, and only the fulfillment of their conjunction is disclosed. Given results of privacy checks for all w constraints in the conjunction, the time complexity is equal to $O(rwn_G)$, and communication complexity is equal to $O(n_G^2)$.

Overall the time complexity for our running example is equal to $O((wm+m+p_S)n_G)$, while the communication complexity is equal to $O(n_G^2)$.

5 Anonymization for *m*-Privacy

After defining the m-privacy verification algorithms and protocols, we can use them to anonymize a horizontally distributed dataset, while preserving m-privacy w.r.t. C. In this section, we present a baseline anonymization algorithm, and then our approach that utilizes a data provider-aware algorithm with adaptive verification strategies to ensure high utility and m-privacy for anonymized data. We also present an SMC protocol that implements our approach in a distributed environment, while preserving security.

For a privacy constraint C that is generalization monotonic, m-privacy w.r.t. C is also generalization monotonic (Theorem 2.1), and most existing generalization-based anonymization algorithms can be easily modified to guarantee m-privacy w.r.t. C. The adoption is straightforward, every time a set of records is tested for privacy fulfillment, we check m-privacy w.r.t. C instead. As a baseline algorithm to achieve m-privacy, we adapted the multidimensional Mondrian algorithm [18] designed for k-anonymity. The main limitation of such 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 w.r.t. C.

5.1 Anonymization Algorithm

We introduce a simple and general algorithm based on the Binary Space Partitioning (BSP) (Algorithm 5). Similar to the Mondrian algorithm, it recursively chooses an attribute to split data points in the multidimensional domain space until the data cannot be split any further without breaching 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 checking strategy for

Algorithm 5: The *provider-aware* anonymization algorithm.

Data: Records T provided by P_j $(j = 1, \ldots, n)$, QI attributes A_i $(i = 1, \ldots, q)$, the m, and a constraint C
Result: Anonymized T^* that is m-private w.r.t. C

```
1 π = get_splitting_points_for_attributes(A_i)
2 π = π ∪ get_splitting_point_for_providers(A_0)
3 π' = {a_i ∈ π, i ∈ {0,1,...,q} :
  are_both_split_subpartitions_m-private(T, a_i)}
4 if π' is ∅ then
5     T* = T* ∪ A_1(T)
6     return T*
7 A_j = choose_splitting_attribute(T, C, π')
8 (T'_r, T'_l) = split(T, A_j)
9 Run recursively for T'_l and T'_r
```

efficient verification. The pseudo code for our *provider-aware* anonymization algorithm is presented in Algorithm 5.

Provider-Aware Partitioning. The algorithm first generates all possible splitting points, π, for QI attributes and data providers (lines 1 to 2). In addition to the multidimensional QI domain space, we consider the data provider of each record as its additional attribute A_0. For instance, each record t contributed by data provider P_1 will have $t[A_0] = P_1$. Introducing this additional attribute adds also 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 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 impose a total order on the providers, e.g., sorting the providers alphabetically or based on the number of records they provide, and partition them into two groups with approximately the same size.

Adaptive Verification for EG-Monotonic *m*-Privacy. m-Privacy is then verified for all possible splitting points, and only those satisfying it are added to a candidate set π' (line 3). 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. The *top-down* algorithm, which takes advantage of the downward pruning, may be used for fast privacy 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. The *binary* algorithm may be used instead to take advantage of upward pruning. We experimentally find the threshold of privacy fitness score for selecting the best algorithm, and confirm the benefit of this strategy.

Privacy Fitness Score Based Splitting Point Selection. Given a non-empty candidate set π' (Algorithm 5), the privacy fitness score (Definition 3.1) is used to find the best split (line 7). Intuitively, if the resulting partitions have higher fitness scores, they are more likely to satisfy m-privacy, and allow 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. Then, the partition is split, and the algorithm is run recursively on each sub-partition (lines 8 and 9).

5.2 Secure Anonymization Protocol

Algorithm 5 can be executed in a distributed environment by a TTP or by all providers running an SMC protocol. In this section we present a secure protocol for semi-honest providers. As an SMC schema we use Shamir's secret sharing, but, when needed, we employ also encryption.

The key idea of the protocol is to use existing SMC protocols. The first step for all providers is to elect the leader P' by running a secure election protocol (Algorithm 9, [26]), which then runs Algorithm 6.

The most important step of the protocol is to choose an attribute used to split records based on fitness scores of record subsets. Splitting is repeated until no more valid splits can be found, i.e., any further split would return records that violate the privacy.

Secure anonymization protocol runs as follows. First, the median of each attribute A_i is found by running the secure median protocol (line 4, [27]). All records with the A_i values less than the median, and some records with the A_i values equal to the median establish the distributed set $T^{s,i}$. Remaining records define the distributed set $T^{g,i}$. Then, m-privacy w.r.t. C is verified for $T^{s,i}$ by running the secure verification protocol, i.e., either Algorithm 2 or 10 (line 8). If $\mathcal{A}_1\left(T^{s,i}\right)$ is m-private w.r.t. C, then the same verification protocol is run for $T^{g,i}$ (line 11). If $\mathcal{A}_1\left(T^{g,i}\right)$ is also m-private w.r.t. C, then this split becomes a candidate split. For each candidate split, minimum fitness score of $T^{s,i}$ and $T^{g,i}$ is computed (secure fitness score protocol is described below). Among candidate splits, the one with the maximal fitness score is chosen, and the protocol is run recursively for its subpartitions (lines 21 to 22). If no such attribute can be found for any group of records, the protocol stops.

Secure m-privacy anonymization protocol calls three different SMC subprotocols: the secure median [27], [28], the secure m-privacy verification (Section 4), and the secure fitness score (Algorithm 7). The last protocol needs to be defined for each privacy constraint C (described below). For the sake of this analysis, we assume that all these protocols are perfectly secure, i.e., all intermediate results can be inferred from the protocol outputs.

At each anonymization step following values are disclosed: medians s_i of all QID attributes, fulfillment of m-privacy w.r.t. C for records split according to every computed median, and, for m-private splits, the order of privacy fitness scores of all verified subsets of records. Medians of all QID attributes need to be revealed to allow each provider defining its local subgroups of records. Announcing results of m-privacy verification for distributed sets of records allow each provider to accept or to drop candidate splits. The best splitting attribute is the one that maximizes fitness scores of split record groups.

Theorem 5.1: The m-privacy anonymization protocol is secure except median values for each attribute, m-privacy fulfillments for records split by these medians, and the order of fitness score values for m-private QI groups.

Algorithm 6: Secure *provider-aware* anonymization protocol.

Data: A set of distributed records T, a set of QI attributes A_i $(i = 1, \ldots, q)$, m, a privacy constraint C.
Result: An anonymized view of distributed records $\mathcal{A}(T)$ that is m-private w.r.t. C.

1 $i_{max} = -1$
2 $[f_{max}] = [0]$
3 **foreach** $i \in \{0, \ldots, q\}$ **do**
4 Find the median value s_i of A_i in the set T (using secure median protocol).
5 Send s_i and A_i to other providers.
6 Locally split set T_j into $T_j^{s,i} = \{t \in T_j : t[A_i] < s_i\}$, and $T_j^{g,i} = \{t \in T_j : t[A_i] > s_i\}$.
7 Locally distribute records $\{t \in T_j : t[A_i] = s_i\}$ among $T_j^{s,i}$ and $T_j^{g,i}$ to reduce their uneven distribution.
8 Securely verify m-privacy w.r.t. C of a distributed set $T^{s,i} = \bigcup_{j=1}^{n} T_j^{s,i}$ (using Algorithm 2 or 10).
9 **if** $T^{s,i}$ *is not* m*-private w.r.t.* C **then**
10 **continue**
11 Securely verify m-privacy w.r.t. C of a distributed set $T^{g,i} = \bigcup_{j=1}^{n} T_j^{g,i}$ (using Algorithm 2 or 10).
12 **if** $T^{g,i}$ *is not* m*-private w.r.t.* C **then**
13 **continue**
14 $[f(T^{s,i})] =$`secure_fitness_score`$(T^{s,i})$
15 $[f(T^{g,i})] =$`secure_fitness_score`$(T^{g,i})$
16 $[f] = \min([f(T^{s,i})], [f(T^{g,i})])$
17 **if** $reconstruct(lessThan([f_{max}], [f])) == 1$ **then**
18 $[f_{max}] = [f]$
19 $i_{max} = i$
20 **if** $i_{max} \geqslant 0$ **then**
21 Run this protocol for $T^{s,i_{max}}$.
22 Run this protocol for $T^{g,i_{max}}$.

Proof: To prove formally that the m-privacy anonymization protocol is secure, we assume that all subprotocols are secure, and present a simulator that, using outputs of the protocol and subprotocols, computes intermediate results. Each party splits its records based on the received median values s_i. Obtained subsets are used only by secure m-privacy verification, and secure fitness score protocols. Disclosing the order of fitness scores for m-private subsets of records allows the simulator to choose the splitting attribute, which has the maximal fitness score value.

If none of possible splits is m-private, then the simulator finishes splitting the current set of records. No other intermediate and undisclosed results are computed during the protocol computation. Finally, since the secure median protocol, and the m-privacy verification protocol, as well as the secure fitness score protocol are assumed to be secure, and from the composition theorem [8] the m-privacy anonymization protocol is secure as well. ■

Complexity Analysis. Before analyzing complexity of the secure anonymization protocol, let us make a note about complexity of the secure median protocol. A secure median protocol for an attribute A_i uses the binary search to find the median. To verify if the median is found, one needs to make sure that there are $n/2$ records with values of A_i not greater and not less than the value, i.e., if both sets split by the value are $n/2$-anonymous (Algorithm 3). The time complexity of such protocol is equal to $O(n^2 \log(domain(A_i)))$. The communication complexity is also equal to $O(n^2 \log(domain(A_i)))$.

Time complexity of the m-privacy anonymization protocol depends on complexities of the secure median protocol M_T, the m-privacy verification protocol V_T, and the secure fitness protocol F_T. Assuming the worst-case scenario (maximal number of splits) for $|T|$ records and q QID attributes, the time complexity is equal to $O(|T|(q+1)(V_T + 2 \cdot V_T + 2 \cdot F_T))$. For our running example the overall time complexity is equal to $O(|T|(q+1)(n^2 + np_S))$.

Communication complexity heavily depends on used subprotocols. M_C, V_C, and F_C denote communication complexities for the secure median, the m-privacy verification, and the fitness score protocols, respectively. The communication complexity for the m-privacy anonymization protocol is equal to $O(|T|(q+1)(3 + M_C + V_C + F_C))$, which for our running example is equal to $O(|T|qn^2)$.

Secure Fitness Score Protocol. Many privacy constraints (including ones we have used in our running example) base on threshold values $\mathcal{T}$. In order to securely compare fitness scores of constraints, they need to be *scaled*, e.g., using the least common multiple (lcm) of all threshold values. After that the secure fitness score can be computed by running the following protocol (Algorithm 7). The elected leader computes the least common multiple of all thresholds from the privacy constraints (line 1). Then, values measured, and compared with thresholds in each privacy constraints can be securely computed (line 3), and *scaled* (line 4). Shares of the minimal one are scaled back, and returned (line 5).

In our running example, we require fulfillment of k-anonymity and l-diversity. Thus, for P_i, $\gamma_1 = |T|$, and γ_2 is equal to the number of distinct sensitive values of local records T. In order to compute γ_1 and γ_2, we run secure k-anonymity, and l-diversity protocols (Algorithm 3 and Algorithm 4 respectively). However, in both protocols we skip comparison of computed values with their thresholds (k and l respectively), and return shares of such values.

Algorithm 7: Secure fitness score protocol.

Data: $\mathcal{T}$ – thresholds from all w constraints, data records T.
Result: Shares of the minimal fitness score value.

1 $lcm =$`least_common_multiple`$(\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_w)$
2 **foreach** $i \in \{0, \ldots, w\}$ **do**
3 Securely compute γ_i, i.e., value measured for C_i, and T
4 $[F_i] = multiplicate([\gamma_i], lcm/\mathcal{T}_i)$
5 **return** $reconstruct\,(\min([F_1], \ldots, [F_w]))\,/lcm$

The Shamir's secret sharing scheme, with secure communication channels, is information-theoretically secure [20]. Correctness and security of the $multiplicate$ subprotocol has been discussed in details in [21]. The above protocol reveals the fitness score value. However, if this protocol is used as a subprotocol, and revealing of the minimal fitness score value is not necessary, then the protocol would return shares of the minimal value, i.e., $\min([F_1], \ldots, [F_w])$.

Complexity Analysis. Computation complexities of shares generation, as well as multiplication for n providers, are

equal to $O(n^2)$ each [21]. Secure minimum protocol requires $(\log_2 w)$ comparisons, which takes $O(n^2)$ time. Thus, the overall time complexity is equal to $O(n^2 \log_2 w) + \sum_{i=1}^{w} time\_complexity(\gamma_i)$. For our running example, the time complexity is equal to $O(n^2 + np_S)$, where p_S is the maximal number of fake values in the l-diversity protocol.

While running the above protocol, each data provider exchanges $w(n-1)$ messages for all multiplications. Secure minimum protocol is implemented using $lessThan$ comparison subprotocol, and therefore its communication complexity is equal to $O(n \log w)$ [21]. The overall communication complexity is then equal to $O(wn^2) + \sum_{i=1}^{w} communication\_complexity(\gamma_i)$, which for our running example is equal to $O(n^2)$.

6 Experiments

We run two sets of experiments for m-privacy w.r.t. C with the following goals: 1) to compare, and evaluate the different m-privacy verification algorithms, and 2) to evaluate, and compare the proposed anonymization algorithm with the baseline algorithm in terms of both utility and efficiency. All experiments have been run for scenarios with a trusted third party (TTP), and without it (SMC protocols). Due to space restrictions all experiments for a TTP setting have been moved to Appendix C. They can also be found in the earlier version of the paper [1].

6.1 Experiment Setup

We merged the training and testing sets of the Adult dataset[2]. Records with missing values have been removed. All remaining 45,222 records have been randomly distributed among n providers. As a sensitive attribute A_S we chose *Occupation* with 14 distinct values.

To implement SMC protocols, we have enhanced the SEPIA framework [21], which utilizes Shamir's secret sharing scheme [19]. Security of communication is guaranteed by the SSL using 128-bit AES encryption scheme. For the secure l-diversity protocol we have used commutative Pohlig-Hellman encryption scheme with a 64-bit key [29].

Privacy Constraints. The EG monotonic privacy constraint is defined as a conjunction of k-anonymity [11] and l-diversity [12]. Privacy fitness score is defined by **Equation 1**. All algorithm parameters, and their default values are listed in the **Table 2**.

TABLE 2
Experiment settings and default values of SMC protocols.

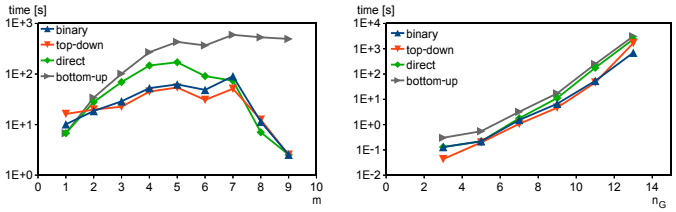
Name	Description	Verification	Anonymization
m	Power of m-privacy	3	3
n	Number of data providers	–	10
n_G	Number of data providers contributing to a group	10	–
$\|T\|$	Total number of records	–	1000
$\|T_G\|$	Number of records in a group	150	–
k	Parameter of k-anonymity	30	30
l	Parameter of l-diversity	3	3

All experiments have been performed on the local network of 64 HP Z210 with 2 quad-core CPUs, 8 GB of RAM, and running Ubuntu 2.6 each. The efficiency of protocols is measured by their computation time.

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

6.2 Secure *m*-Privacy Verification

The objective of the first set of experiments is to evaluate the efficiency of different heuristics in generating attacker coalitions for privacy verification. Note that computation times are presented in seconds, not milliseconds.



(a) (b)

Fig. 4. Computation time (logarithmic scale) vs. power of *m*-privacy and number of data providers.

Attack Power. In this experiment, we compare m-privacy verification heuristics against different attack powers, and different number of data providers. **Fig. 4(a)** shows computation time with varying m for all heuristics.

Similar to the TTP implementation, the secure protocols for the *top-down* and *binary* algorithms demonstrate the best performance. The difference between these two approaches is negligible for most values of m. The *direct* approach is not that efficient as the above algorithms except small and large values of m. The *bottom-up* approach is useful only for very small values of m.

Numbers of messages that are generated, while running protocols (not shown), are between 10^4 and 10^6 for different m, and confirm our conclusions.

Number of Contributing Data Providers. In this experiment we analyze the impact of increasing number of data providers, n_G, on different algorithms. **Fig. 4(b)** shows the runtime of different heuristics with varying n_G.

As expected, the computation time increases exponentially with the number of data providers. Differences among approaches are not significant, and as above *top-down* and *binary* algorithms are more efficient than other approaches. The *bottom-up* heuristic is the slowest among others.

6.3 Secure *m*-Privacy Anonymization

This set of experiments compares estimates of our *provider-aware* and the *baseline* approaches, and evaluates the overhead of our solution. Due to high runtime of protocols, we estimated their computation times using runs of TTP algorithms, and computation times of subprotocols.

As a comparison, we implemented an independent approach in which each provider anonymizes its data on its own. We observe that its runtime is independent of m and n, and equals to 1.2 seconds (not shown). However, the query error is significantly worse than for the collaborative setting (Appendix C.3).

Attack Power. We first evaluate both anonymization heuristics with varying attack power m. **Fig. 5(a)** shows the estimated computation time with varying m for both approaches. As expected for EG monotonic constraints, increasing m results in stopping anonymization process

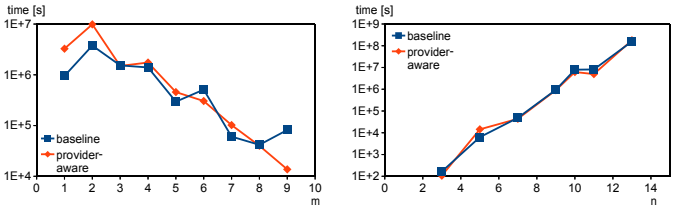


Fig. 5. Computation time vs. m and the number of providers.

significantly earlier. In addition, both approaches have comparable computation times with negligible differences.

Number of Data Providers. In this experiment, we estimate computation times for different number of data providers n, but with the same average number of records per provider ($|T|/n = 100$). **Fig. 5(b)** shows estimated time with varying the number of providers for both algorithms. As expected, the computation time is similar for both approaches, and increases exponentially with n.

7 Related Work

Privacy preserving data analysis and publishing has received considerable attention in recent years [2]–[4]. Most work has focused on a single data provider setting, and considered the data recipient as an attacker. A large body of literature [3] assumes limited background knowledge of the attacker, and defines privacy using relaxed *adversarial* notion [12] by considering specific types of attacks. Representative principles include k-anonymity [10], [11], l-diversity [12], and t-closeness [16]. A few recent works have modeled the instance level background knowledge as corruption, and studied perturbation techniques under these syntactic privacy notions [30]. In the distributed setting that we study, 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 [2], [4] is an unconditional privacy guarantee, but only for statistical data release or data computations.

There are some works focused on anonymization of distributed data. [6], [31], [32] studied distributed anonymization for vertically partitioned data using k-anonymity. Zhong et al. [33] studied classification on data collected from individual *data owners* (each record is contributed by one data owner), while maintaining k-anonymity. Jurczyk et al. [34] proposed a notion called l'-site-diversity to ensure anonymity for data providers in addition to privacy of the data subjects. Mironov et al. [35] studied SMC techniques to achieve differential privacy. Mohammed et al. [5] proposed SMC techniques for anonymizing distributed data using the notion of LKC-privacy to address high dimensional data. Gal et al. [15] proposed a new way of anonymization of multiple sensitive attributes, which could be used to implement m-privacy w.r.t. l-diversity with providers as one of sensitive attributes. However, this approach uses the same privacy requirements for all sensitive attributes, while m-privacy has no such limitation.

Nergiz et al. [36] proposed a *look ahead* approach in horizontally distributed anonymization. In their approach providers disclose some information about data in order to decide if collaborative anonymization will gain more information than individual one. We leave for the future research applying the *look ahead* approach to colluding scenarios considered with m-privacy.

Our work is the first that considers data providers as potential attackers in the collaborative data publishing setting, and explicitly models their inherent instance knowledge as well as potential collusion between them.

The m-privacy verification problem in the combinatorial m-adversary search space is reminiscent of the frequent itemset mining problem, in which the search space is the combination of all items. An example of EG monotonic constraints is *support*, which is used in mining itemsets. Each item corresponds to a single data provider, and a frequent itemset represent a group of private records. Due to the apriori property of frequent itemsets or EG monotonicity of the frequency count, both upward and downward pruning are possible. Taking advantage of the *dual-pruning* is an essential point of the algorithm presented in [37]. The main difference with our approach is the goal of constraint verifications. To find frequent itemsets, all itemsets need to be decided either by checking or pruning. Checking m-privacy of records for EG monotonic constraint requires finding out if all m-coalitions are not able to compromise privacy of remaining records (Corollary 2.3). After simple modifications (e.g., not using *early stop*) our algorithm can find frequent itemsets, and the dual-pruning algorithm can verify m-privacy, but, in both cases, inefficient.

8 Conclusions

In this paper we considered a new type of potential attackers in collaborative data publishing – a coalition of data providers, called m-adversary. Privacy threats introduced by m-adversaries are modeled by a new privacy notion, m-privacy, defined with respect to a constraint C.

We presented heuristics to verify m-privacy w.r.t. C. A few of them check m-privacy for EG monotonic C, and use adaptive ordering techniques for higher efficiency. We also presented a *provider-aware* anonymization algorithm with an adaptive verification strategy to ensure high utility and m-privacy of anonymized data. Experimental results confirmed that our heuristics perform better or comparable with existing algorithms in terms of efficiency and utility.

All algorithms have been implemented in distributed settings with a TTP and as SMC protocols. All protocols have been presented in details with their security and complexity carefully analyzed. Implementations of algorithms for the TTP setting is available on-line for further development[3]. There are many potential research directions and open questions, e.g., modeling settings with vertically or ad-hoc distributed datasets, generalizing our approach to other kinds of data such as set-valued data.

3. http://www.mathcs.emory.edu/aims/m-anonymizer/

Acknowledgments. This research is supported by a Cisco research award and AFOSR under grant FA9550-12-1-0240. 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] S. Goryczka, L. Xiong, and B. C. M. Fung, "m-Privacy for collaborative data publishing," in *Proc. of the 7th Intl. Conf. on Collaborative Computing: Networking, Applications and Worksharing*, 2011.

[2] C. Dwork, "Differential privacy: a survey of results," in *Proc. TAMC*, 2008, pp. 1–19.

[3] 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.

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

[5] N. Mohammed, B. C. M. Fung, P. C. K. Hung, and C. Lee, "Centralized and distributed anonymization for high-dimensional healthcare data," *ACM TKDD*, vol. 4, no. 4, pp. 18:1–18:33, 2010.

[6] W. Jiang and C. Clifton, "Privacy-preserving distributed k-anonymity," in *DBSec*, vol. 3654, 2005, pp. 924–924.

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

[8] O. Goldreich, *Foundations of Cryptography: Volume 2*. CUP, 2004.

[9] 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.

[10] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE TKDE*, 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] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-Diversity: Privacy beyond k-anonymity," in *ICDE*, 2006, p. 24.

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

[14] L. Sweeney, "Uniqueness of Simple Demographics in the U.S. Population," Carnegie Mellon University, Tech. Rep., 2000.

[15] T. S. Gal, Z. Chen, and A. Gangopadhyay, "A privacy protection model for patient data with multiple sensitive attributes," *IJISP*, vol. 2, no. 3, pp. 28–44, 2008.

[16] N. Li and T. Li, "t-Closeness: Privacy beyond k-anonymity and l-diversity," in *ICDE*, 2007.

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

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

[19] A. Shamir, "How to share a secret," *CACM*, vol. 22, no. 11, pp. 612–613, nov 1979.

[20] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the twentieth annual ACM symposium on Theory of computing*, ser. STOC, 1988, pp. 1–10.

[21] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "Sepia: Privacy-preserving aggregation of multi-domain network events and statistics," in *USENIX Security Symposiun*. USENIX, 2010.

[22] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *SIGKDD Explor. Newsl.*, vol. 4, pp. 28–34, December 2002.

[23] R. Sheikh, B. Kumar, and D. K. Mishra, "A distributed k-secure sum protocol for secure multi-party computations," *J. of Computing*, vol. 2, pp. 68–72, March 2010.

[24] P. Paillier and D. Pointcheval, "Efficient public-key cryptosystems provably secure against active adversaries," in *Proc. of the Intl Conf. on the Theory and Applications of Cryptology and Information Security*, ser. ASIACRYPT, 1999, pp. 165–179.

[25] J. Vaidya and C. Clifton, "Secure set intersection cardinality with application to association rule mining," *J. Comput. Secur.*, vol. 13, pp. 593–622, July 2005.

[26] A. Russell and D. Zuckerman, "Perfect information leader election in $\log^* n + O(1)$ rounds," in *Proc. of the 39th Annual Symposium on Foundations of Computer Science*, 1998, pp. 576–583.

[27] G. Aggarwal, N. Mishra, and B. Pinkas, "Secure computation of the kth-ranked element," in *Advances in Cryptology – EUROCRYPT*, 2004, pp. 40–55.

[28] M. Burkhart and X. A. Dimitropoulos, "Fast privacy-preserving top-k queries using secret sharing," in *ICCCN*, 2010, pp. 1–7.

[29] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance (Corresp.)," *IEEE Trans. Inf. Theor.*, vol. 24, no. 1, pp. 106–110, 2006.

[30] Y. Tao, X. Xiao, J. Li, and D. Zhang, "On anti-corruption privacy preserving publication," in *ICDE*, 2008, pp. 725–734.

[31] W. Jiang and C. Clifton, "A secure distributed framework for achieving k-anonymity," *The VLDB Journal Special Issue on Privacy-Preserving Data Management*, vol. 15, no. 4, pp. 316–333, 2006.

[32] N. Mohammed, B. C. M. Fung, K. Wang, and P. C. K. Hung, "Privacy-preserving data mashup," in *EDBT*, 2009, pp. 228–239.

[33] 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.

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

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

[36] M. E. Nergiz, A. E. Çiçek, T. B. Pedersen, and Y. Saygın, "A look-ahead approach to secure multiparty protocols," *IEEE TKDE*, vol. 24, pp. 1170–1185, 2012.

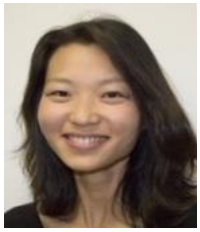
[37] C. Bucila, J. Gehrke, D. Kifer, and W. White, "Dualminer: a dual-pruning algorithm for itemsets with constraints," in *Proc. of the 8th ACM SIGKDD (KDD)*, 2002, pp. 42–51.

[38] X. Xiao and Y. Tao, "Anatomy: simple and effective privacy preservation," in *Proc. VLDB*, 2006, pp. 139–150.

[39] 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.

[40] G. Boros and V. Moll, *Irresistible Integrals: Symbolics, Analysis and Experiments in the Evaluation of Integrals*. CUP, 2004.

Slawomir Goryczka is a Ph.D. student of Mathematics and Computer Science at Emory University. He holds M.S. degrees in both Computer Science and Applied Mathematics from AGH University of Science and Technology, Poland. After graduation he worked as an Operation Research Analyst/Developer for 2 years. His research interests include data privacy and security in distributed settings, distributed data management, privacy preserving data mining, and system software.



Li Xiong is an Associate Professor of Mathematics and Computer Science at Emory University where she directs the Assured Information Management and Sharing (AIMS) research group. She holds a Ph.D. from Georgia Institute of Technology, an M.S. from Johns Hopkins University, and a B.S. from University of Science and Technology of China, all in Computer Science. Her areas of research are in data privacy and security, distributed data management, and bio and health informatics. She is a recent recipient of the Career Enhancement Fellowship by Woodrow Wilson Foundation. Her research is supported by NSF, AFOSR, and research awards from Cisco and IBM.

Benjamin C. M. Fung is an Associate Professor at Concordia University in Montreal, and a research scientist of the National Cyber-Forensics and Training Alliance Canada. He received a Ph.D. degree in Computing Science from Simon Fraser University in 2007. He has over 50 refereed publications that span across the prestigious research forums of data mining, privacy protection, cyber forensics, web services, and building engineering. His data mining work in authorship analysis has been widely reported by media worldwide.