Technical Report

TR-2011-004

m-privacy for collaborative data publishing

by

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

Mathematics and Computer Science

EMORY UNIVERSITY

m-Privacy for Collaborative Data Publishing

February 01, 2012

Slawomir Goryczka
Emory University
sgorycz@emory.edu

Li Xiong
Emory University
lxiong@emory.edu

Benjamin C. M. Fung
Concordia University
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 EG monotonicity of privacy constraints, and adaptive ordering techniques for efficiently checking m-privacy given a group of records. For remaining privacy constraints we present a verification algorithm with the minimal number of privacy checks. 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 implement all algorithms (verification and anonymization) for settings with a trusted third party and introduce secure computation protocols for scenarios without such party. All protocols are extensively analyzed, their security and efficiency is 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.

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 [7, 9, 8] 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 [9, 23]. 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* [12, 13, 9, 23], 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 [10, 20].

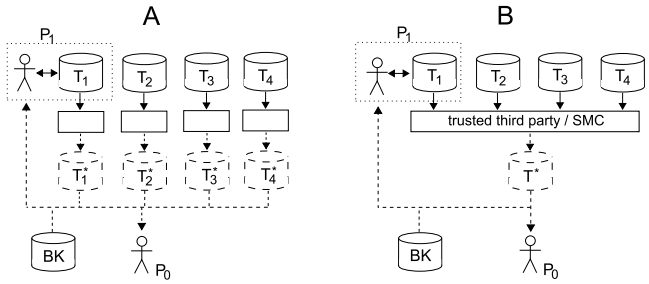

Figure 1: Distributed data publishing settings for four data providers.

Problem Settings. We consider the collaborative data publishing setting (**Figure 1B**) with horizontally distributed 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. In addition, each data owner, e.g., a patient, can share its data with many data providers, e.g., hospitals, which may not know about it. Permission to share data is not transitive, and data providers must not to share owner's data with others. A data recipient may have also access to some background knowledge (BK in **Figure 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^*, such that a data recipient, including data providers, will not be able to compromise the privacy of the individual records. Considering different types of malicious

[1]The Nationwide Health Information Network (NHIN), http://www.hhs.gov/healthit/healthnetwork/background/

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$10.00.

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 is the focus of this paper.

Attacks by External Data Recipient Using Anonymized Data. 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, such as publicly available external data. Most literature on privacy preserving data publishing in a single provider setting considers only such attacks [9]. Many of them adopt a weak or relaxed *adversarial* or *Bayes-optimal privacy* notion [21] to protect against specific types of attacks by assuming limited background knowledge. For example, fulfilling k-anonymity [27, 29] protects against identity disclosure attacks by requiring each quasi-identifier equivalence group (QI group) to contain at least k records. Another examples are l-diversity, which fulfillment requires each QI group to contain at least l "well-represented" sensitive values [21], and t-closeness, which fulfillment requires the distribution of a sensitive attribute in each QI group to be close to the global distribution [19]. In contrast, differential privacy [7, 8] publishes statistical data or computational results of data, and gives unconditional privacy guarantees independent of attackers background knowledge about attribute values.

Attacks by Data Providers Using Intermediate Results and Their Own Data. In addition to the inference attack by external recipient, collaborative data publishing introduces inference attacks by a new type of attackers – the data providers. We assume the data providers are semi-honest [10, 20], 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 [12] can be used to guarantee there is no disclosure of *intermediate* information *during* the anonymization. However, neither TTP nor SMC protects against inferring additional information about other records using the anonymized data and providers data (discussed below). The problem is orthogonal to techniques use to ensure security of computations. Relying on TTP makes computations easier to implement, but requires presence of a trusted party. If such party cannot be found, then all providers may run an SMC protocol, which implements the same algorithm. However, the SMC protocol has higher time and communication complexities. In this paper we present both implementations of data anonymization model, TTP heuristics and SMC protocols.

Attacks by Data Providers Using Anonymized Data and Their Own Data. Each data provider, such as P_1 in **Figure 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 first 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 (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 [4].

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 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 considered when the data are anonymized using a weak privacy constraint, which assumes no or very limited instance level knowledge.

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. Notice that one record 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 using existing algorithms that guarantees k-anonymity and l-diversity ($k = 2$, $l = 2$). Note that the definition of l-diversity, which we use, defines "well represented" sensitive values as distinct values, i.e., l-diversity holds if each equivalence group contains records with at least l different sensitive 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. There will be only one remaining record, which belongs to a patient between 20 and 30 years old. Combining this information with background knowledge BK, P_1 can identify Sara as the owner of the 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 using information about her former patients to infer data at other nodes.

Contributions. In this paper, we present a new privacy notion for distributed datasets. Some previous results have been already presented [11], and here we complete and extend them.

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, anonymized data satisfy m-privacy with respect to l-diversity if the records from each QI group excluding ones from any m-adversary still satisfy l-diversity. In our example (**Table 1**), T_b^* is an anonymization that satisfies m-privacy ($m = 1$) with respect to k-anonymity and l-diversity ($k = 2$, $l = 2$).

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

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

Table 1: m-Adversary and m-privacy example.

on a novel notion of privacy fitness. We also 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.

Our new contributions 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 for which pruning strategies and adaptive ordering techniques cannot be applied. We enumerate all distinct and necessary privacy checks and prove that no fewer checks can be run to confirm m-privacy w.r.t. non-monotonic constraint.

Second, we implement all presented algorithms for distributed scenarios with a trusted third party and for scenarios without such party we introduce SMC protocols. For all, verification and anonymization secure protocols, regardless monotonicity of C, we prove their security and complexity.

Finally, we enhance our implementation to deal with duplicated records, and with more than one sensitive attribute defined in the dataset. A bunch of other upgrades in our m-anonymizer framework completes the picture.

2. *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 properties of this new privacy notion.

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 is 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. 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 by 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.

In our example (**Table 1**), the privacy constraint C is defined as $C = C_1 \wedge C_2$, where C_1 is k-anonymity with $k = 2$ (Definition A.1), and C_2 is l-diversity with $l = 2$ (Definition A.2). Both anonymized tables T_a^* and T_b^* satisfy 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. 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 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,*

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

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, 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 contributed by a few data providers (such as records from a patient transfered 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 the background knowledge of the m-adversary.

m-Privacy and Weak Privacy Constraints. Given a weak privacy constraint C that does not consider instance level background knowledge, e.g., k-anonymity, l-diversity, and t-closeness (Definition A.3), a T^* satisfying C will only guarantee 0-privacy w.r.t. C, i.e., C is not guaranteed to hold for each QI group after excluding records belonging to

any single data provider. Thus, each data provider may be able to breach privacy of records provided by others. In our example from **Table 1**, T_a^* satisfies only 0-privacy w.r.t. C, 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 all 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 attacks [21] or deFinetti attack [16]. Thus, m-privacy w.r.t. C will protect against privacy attacks issued by any m-adversary if and only if C protects against the same privacy attack by any external data recipient. m-Privacy notion is orthogonal to the privacy constraint C being used, and enhances privacy guaranteed by C to scenarios where up to m data providers collude.

m-Privacy and Differential Privacy. Differential privacy [7, 8, 17] does not assume specific background knowledge, and guarantees privacy even if an attacker knows all but one record. Thus, any statistical data (or records synthesized from the statistical data) satisfying differential privacy also satisfies $(n-1)$-privacy, maximum level of m-privacy, when any $(n-1)$ providers can collude. For queries other than statistical ensuring differential privacy is more complex, and sacrifice some data utility, e.g., generating artificial data sets based on differential private results of queries [33].

While m-privacy w.r.t. any weak privacy notion does not guarantee unconditional privacy, it offers a practical trade-off 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. different weak privacy constraints.

2.2 Monotonicity of Privacy Constraints

Monotonicity of privacy constraints is defined, and considered 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 for privacy constraints has been already defined in the literature (Definition 2.2) [21, 19]. Its fulfillment is crucial for designing efficient generalization algorithms, which results satisfy a privacy constraint [29, 18, 21, 19]. In this paper we will refer to this monotonicity as *generalization monotonicity*.

Definition 2.2. (Generalization Monotonicity of a Privacy Constraint [21, 19]) *A privacy constraint C is generalization monotonic if and only if for equivalence groups of any two record sets T and T' that satisfy C, an equivalence group of 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 generalization monotonicity definition there is an assumption that original records T have been already anonymized into many equivalence groups, 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 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 any set of records T for which $\mathcal{A}_1(T)$ satisfies C, all equivalence groups of its supersets satisfy C as well,*

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

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 [21] and t-closeness [19] are examples of generalization monotonic constraints, which 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 QI group is uniform. Entropy l-diversity is not EG monotonic because it will not hold if we add records that will change the distribution of sensitive values (and entropy) significantly. However, it is generalization monotonic because it will still hold if any other QI subgroup satisfying entropy l-diversity ($l = 2$) is added (generalized) into the first one.

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 [21], 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.*

This theorem holds also when applied for generalization monotonicity. Proofs of this theorem for EG and generalization monotonicities can be found in the Appendix B. Notice 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 such that only $T' = T \setminus T_I$ are considered, i.e., $\mathcal{A}(T)$ satisfies m-privacy w.r.t. C if and only if,*

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

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.

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 privacy checks. 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 group of records w.r.t. an EG monotonic

constraint C. Implementation of introduced algorithms can be just run by a trusted third party (TTP). For scenarios without TTP we introduce a secure multi-party (SMC) protocol to verify m-privacy w.r.t. C. Finally, we present modifications of TTP heuristics and the SMC protocol that allow 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 **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 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.

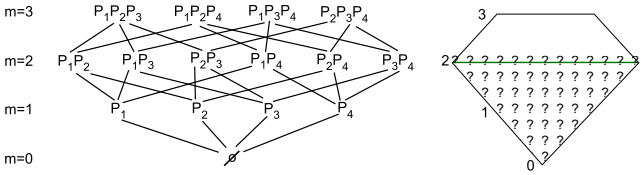

Figure 2: m-Adversary space.

In order to verify m-privacy w.r.t. a constraint C for a set of records, we need to check 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 laying 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 $\binom{n_G}{n_G/2} = O(2^{n_G} n_G^{-1/2})$. Thus, the *direct* algorithm is not efficient enough.

3.2 Heuristic Algorithms

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

The key idea of our heuristic algorithms 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 strategies are possible thanks to the EG monotonicity of m-privacy (Observations 2.1, 2.3). For these scenarios, 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.

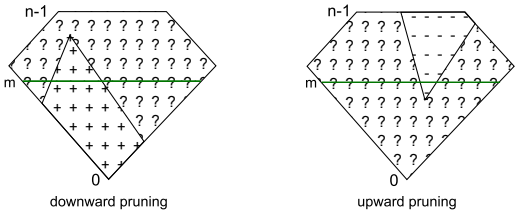

Figure 3: Pruning strategies for m-privacy check.

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 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 upward pruning (early-stop).

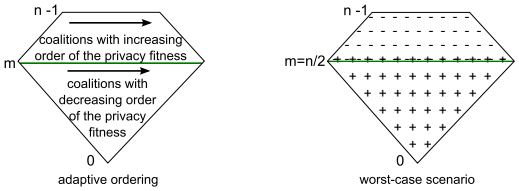

Figure 4: Adaptive ordering for efficient pruning and the worst-case scenario without any pruning possible.

To quantify privacy fulfillment by a set of records, which is used to measure the attack power of a coalition and privacy of remaining 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 anonymized 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:

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

The privacy fitness score quantifies the attack power of attackers. The higher their privacy fitness scores are, the more

likely they will be able to breach the privacy of 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 directions. All of them use the adaptive ordering of adversaries to enable fast pruning. Notice that for a record contributed by many providers, if any of them is an attacker then the record is considered as it would have been provided only by the attacker.

The straight forward algorithm to verify m-privacy w.r.t. an EG monotonic C is to check all possible m-adversaries as potential coalitions of attackers. This approach can be very inefficient, especially when m is close to $n_G/2$. However, because of its simplicity we treat it as our baseline, and call it the *direct* algorithm.

The Top-Down Algorithm. The *top-down* algorithm (Algorithm 1) 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.

Algorithm 1: The *top-down* m-privacy w.r.t. C verification algorithm.

Data: A set of anonymized records T^* provided by $P_1, \ldots, P_n$, an EG monotonic C, a privacy fitness scoring function $score_F$, and the m value.
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  foreach i = n − 1, n − 2, ..., m do
4      while is_m-privacy_verified(T*, i) = false do
5          I = next_coalition_of_size(i)
6          if privacy_is_breached_by(I) = false then
7              prune_all_sub-coalitions_of(I)
8          if is_m-privacy_verified(T*, m) = true then
9              return true
10 return false
```

The Bottom-Up Algorithm. The *bottom-up* algorithm (Algorithm 2) 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 (Algorithm 3), inspired by the binary search algorithm, checks coalitions between $(n_G - 1)$-adversaries and m-adversaries, and takes advantage of both upward and downward pruning (**Figure 5**). 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 2: The *bottom-up* m-privacy w.r.t. C verification algorithm.

Data: A set of anonymized records T^* provided by $P_1, \ldots, P_n$, an EG monotonic C, a privacy fitness scoring function $score_F$, and the m value.
Result: *true* if T^* is m-private w.r.t. C, *false* otherwise.

```
1  sites = sort_sites(P, decreasing_order, score_F)
2  use_adaptive_order_generator(sites, m)
3  foreach i = 0, 1, ..., m do
4      while is_m-privacy_verified(T*, i) = false do
5          I = next_coalition_of_size(i)
6          if privacy_is_breached_by(I) = true then
7              return false // early stop
8  return true
```

Algorithm 3: The *binary* m-privacy w.r.t. C verification algorithm.

Data: A set of anonymized records T^* provided by $P_1, \ldots, P_{n_G}$, an EG monotonic C, a privacy fitness scoring function $score_F$, and the m value.
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) = false do
4      I_super = next_coalition_of_size(n_G − 1)
5      if privacy_is_breached_by(I_super) = false then
6          prune_all_sub-coalitions(I_super)
7          continue
8      I_sub = next_sub-coalition_of(I_super, m)
9      if privacy_is_breached_by(I_sub) = true 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) = true 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, finds the first coalition of attackers that violates privacy, and assigns it to I_{super} (lines from 4 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 pairs I_{super} and I_{sub} are found (line 11). Then both upward and downward pruning are performed (lines 17 and 18), and the algorithm moves to the next iteration. The algorithm stops when m-privacy can be determined (line 3).

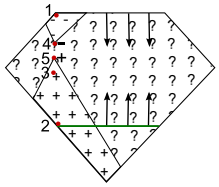

Figure 5: The *binary* m-privacy w.r.t. C verification algorithm.

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 the group, may be used to select the most suitable verification 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 an adaptive strategy in the anonymization algorithm (discussed later), and experimentally evaluate all presented algorithms.

Non-EG Monotonic m-Privacy. If a privacy constraint C applied to the definition of m-privacy (Definition 2.1) is not EG monotonic, then pruning strategies are not useful. The only way to verify m-privacy w.r.t. C for this setting is to check all possible sets of records that can be used by any m-adversary in attacks (attacking records). The domain of privacy checks needs to be expanded to cover all possible scenarios of attacks for different sets of attacking records.

A general verification algorithm for m-privacy w.r.t. any C (Algorithm 4) works as follow. For each cardinality of m-adversary, starting from 0, it generates all possible coalitions of adversaries (lines 1 to 2). Then, for each coalition it generates all possible subsets of the coalition records such that each provider participate to this set with at least one record (line 4). Finally, it verifies for such subsets if using them to attack remaining records breaches privacy (line 5). If the attack is successful then no further checks are necessary, and the algorithm returns negative answer (*early stop*, line 6). After verifying that all possible subsets of records provided by any m-adversary are not enough to breach privacy the algorithm returns positive answer.

Notice that for a given i-adversary the algorithm does not generate all possible subsets of its records (line 4). Subsets, with not all providers participating their records, are skipped because they have been already verified. Each malicious provider that is not using any of its records for the privacy attack can be treated as a non-attacker. But that setting has been already checked while considering smaller coalitions.

Algorithm 4: The verification algorithm of m-privacy w.r.t. any constraint C.

Data: A set of anonymized records T^* provided by $P_1, \ldots, P_n$, a privacy constraint C, and the m value.
Result: *true* if T^* is m-private w.r.t. C, *false* otherwise.

```
1 foreach i = 0, 1, ..., m do
2     foreach I ∈ ordered_coalitions_of_size(i) do
3         T_I = ⋃_{P_j ∈ I} records_of(P_j)
4         foreach S ∈ 2^{T_I} : providers_of(S) = I do
5             if privacy_does_not_hold_for(T* \ S)
              then
6                 return false // early stop
7 return true
```

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 the 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.

EG Monotonic m-Privacy. 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. For this 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, 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 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 and within each $O(\log(n_G-m))$ privacy checks. 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 is strongly correlated with the parameter m. 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 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*[2], and $O\left(2^{n_G}\frac{\log_2 n_G}{n_G}\right)$ for *binary*. Thus, adapting m-

[2]In the shorter version of this paper [11] the average time

privacy verification strategy to settings is crucial to achieve, on average, a low runtime. The detailed analysis can be found in the Appendix D.

Non-EG Monotonic m-Privacy. In order to verify m-privacy w.r.t. non-EG monotonic constraint C for a group of anonymized records provided by P parties maximal number of privacy checks follows the formula,

$$\sum_{i=0}^{m} \sum_{\substack{I \subset P \\ |I|=i}} \prod_{R \in I} \left(2^{|records\_of(R)|} - 1\right) \tag{2}$$

Detailed derivation of the formula can be found in the Appendix D.

3.4 Secure Privacy Verification Protocols

Secure verification of m-privacy w.r.t. C requires for a set of records a secure protocol of checking its privacy for a given C. Thus, privacy verification needs to be implemented independently. Presenting protocols for all privacy notions is not in the scope of this paper, but for the sake of our introduction example we present secure verification protocols for k-anonymity and l-diversity. For all protocols we assume semi-honest model, i.e., adversaries are curious but follow the protocol.

All parties choose a leading provider, which initiates and runs the m-privacy verification protocol. The leading data provider is chosen randomly by running a secure leader election algorithm [26].

Secure k-Anonymity Verification. To securely verify k-anonymity of a distributed set of records the leading provider can compute number of records using a secure sum protocol [5, 28, 25] and securely compare it with k. The leader run the secure comparison of summation and a number protocol (SCoSAN, Algorithm 10). Many of those implementations are also secure against actively malicious attackers.

Secure l-Diversity Verification. To securely verify if a set of distributed records is l-diverse the leading provider runs, e.g., a secure set union protocol, and compares size of the result with l [5]. By running this protocol the set of sensitive values is disclosed. To avoid this privacy leak one may use the secure size of set union protocol, which returns number of distinct sensitive attribute values of records. The protocol is run in the same way as the secure size of set intersection [5, 31] except the last step. In the last step, instead counting values that are present in *all* of the encrypted itemsets, we count values that are present in *any* encrypted itemset. The result is compared against l, and based on that the privacy is determined.

Complexity and security of secure size of set union protocol are exactly the same as for the secure size of set intersection protocol [31]. The protocol requires running $O(n_G \cdot l_{\max})$ encryptions, where $l_{\max}$ is maximal number of distinct sensitive values hold by records delivered by one provider. In the worst-case scenario $l_{\max}$ is equal to the cardinality of the sensitive attribute domain.

3.5 Secure m-Privacy Verification Protocols

All verification algorithms (*top-down*, *bottom-up*, *direct*, and *binary*) can be easily implemented, and run by a trusted third-party (TTP). For settings without any TTP data providers need to run an SMC protocol that implements one of verification algorithms. We assume that all providers are semi-honest (honest but curious).

Structure of Protocols. In order to apply adaptive ordering data providers should be sorted and ordered according to fitness scores of their records. To sort providers each of them prepares a pair of numbers with a randomly generated identifier and the fitness score of its records. New and individual identifiers of providers protect their identities, such that other parties are not able to learn their fitness scores. Then, to order all providers, the leading provider runs a secure sorting protocol [35]. If any two identifiers are the same, current results are dropped, and the protocol is rerun again. Finally, each provider finds its ID, and based on that determines its index among others. Ordered providers can be connected in a ring, where all providers pass messages they receive, and only addressee interprets them.

Secure verification of m-privacy w.r.t. C requires flexibility of running secure protocols for different C. Secure C fulfillment verification is implemented as a separate SMC protocol. Thus, all results of privacy verifications are disclosed. An SMC protocol that would not disclose results of verifications needs to be design and implemented for already defined C. Changing C requires redesigning the whole protocol from scratch.

Secure EG Monotonic m-Privacy Verification. All verification algorithms have many common components, but they verify privacy of given sets of records in different orders (Algorithm 5). We assume that verification of privacy defined by C, is implemented in a separated SMC protocol (Section 3.4).

After generating a coalition, the leader sends IDs of its members to all parties, and each party recognizes its status (attacker/non-attacker) for the current privacy check. Then, the leader runs the secure m-privacy verification protocol to check if records used by the coalition of attackers I can breach privacy (line 4). The result of this verification is announced to all parties. Then, the leader checks if m-privacy w.r.t. C can be decided (lines 5, and 1), and if so the protocol stops and returns the results of m-privacy verification (line 8). If m-privacy w.r.t. C cannot be determined, the leading provider generates next coalition of attackers, and repeat above steps.

Size of generated coalitions is determined by the choice of the verification algorithm, e.g., *top-down* verifies large coalitions first, and then decrements their sizes, *bottom-up* generates coalitions with the ascending size starting from 0. All coalitions that have been already checked or results of their attacks can be determined based on pruning, are skipped by generators. Details of security and overhead analyses can be found in the Appendix E.

Secure m-Privacy Verification for the *binary* Algorithm. Similar like for other m-privacy verification algorithms the *binary* algorithm can be easily implemented as an SMC protocol. The protocol is run by the leading provider. Results of privacy checks are announced to other providers. Thus, each of them is able to recognize when m-privacy is determined or the protocol should be run further. All steps of secure *binary* protocol are the same as run by a TTP (Algorithm 3). The only exception are privacy verifications, which are run as separate SMC protocols.

complexity is estimated using different model, which results in less precise estimation.

Algorithm 5: Secure m-privacy w.r.t. EG monotonic constraint C verification protocol for secure *top-down*, *bottom-up*, and *direct* algorithms; code run by the leading provider P_1.

Data: Local records T_1, an EG monotonic privacy constraint C, and the m value.
Result: *true* if $\mathcal{A}_1\left(\bigcup_{i=1}^{n} T_i\right)$ is m-private w.r.t. C, *false* otherwise.

```
1 while is_m-privacy_determined() = false do
2     I = generate_next_coalition()
3     Broadcast I.
      // Run secure privacy verification.
4     privacy_breached = is_privacy_breached_by(I)
5     if privacy_breached and |I| ⩽ m then
6         return false // early stop
7     prune(I, privacy_breached)
8 return is_m-private()
```

Secure non-EG Monotonic m-Privacy Verification. If a privacy constraint C is not EG monotonic, then the TTP algorithm (Algorithm 4) cannot be adapted straightforward as an SMC protocol. A single coalition of potential attackers may use any subset of records provided by its members. Thus, there are many privacy verifications that need to be run in order to check if the coalition is able to breach privacy.

The pseudo code of the protocol is presented in the Algorithm 6. First, each provider broadcasts number of records it provided (line 1). This step seems to be redundant, but even without publishing the number of records it can be inferred by counting runs of secure verification protocol. By broadcasting it upfront parties can synchronize their generators of record subsets used in potential attacks.

Starting from the smallest possible set of attackers (0-adversary) the leading provider generates and broadcasts coalitions of attackers I (lines 5 to 6). Then, it computes t, the number of privacy checks for all possible sets of attacking records (line 8). A set of records that could be used in a single attack is a union $\bigcup_{P_i \in I} T_i^I$, i.e., attacking records selected by each party. The set of records selected by a party P_i is changed only if all possible combinations of records used by attackers with lower indices in the ring have been already checked (line 10). Until then the current set is used. Then, the leader runs a secure privacy verification protocol for a set of attacking records used by I (line 11). Finally, if no coalition of up to m providers could breach privacy, then the given set of records is m-private w.r.t. C (line 14). Details of security and overhead analyses can be found in the Appendix E.

4. ANONYMIZATION FOR m-PRIVACY

After defining the m-privacy verification algorithm, we can use it to anonymize a horizontally distributed dataset while preserving m-privacy w.r.t. C. In this section, we 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. We also present an SMC protocol that implements our approach in a distributed environment while preserving security.

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 easily modified to guarantee m-privacy w.r.t. C. The adoption is straight forward, 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 [18] designed originally 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 w.r.t. C.

Algorithm 6: Secure verification protocol for m-privacy w.r.t. non-EG monotonic constraint C run by the leading provider P_1.

Data: Local records T_1, a non-EG monotonic privacy constraint C, and the m value.
Result: *true* if $\mathcal{A}_1\left(\bigcup_{i=1}^{n} T_i\right)$ is m-private w.r.t. C, *false* otherwise.

```
1  All providers broadcast numbers of their records |T_i|.
2  foreach k = 0, 1, ..., m do
3      I = {}// Coalition of attackers.
4      while I ≠ ∅ do
5          I = generate_next_coalition_of_size(k)
6          Broadcast I.
7          T_1^I = ∅ // Attacking records.
           // Number of all privacy checks for I.
8          t = ∏_{P_s ∈ I} (2^{|T_s|} − 1)
9          foreach j = 0, 1, ..., t − 1 do
10             Send request to all P_i ∈ I, which should
               update their attacking records.
               // Securely verify privacy.
11             privacy_breached =
               is_privacy_breached_using(⋃_{P_s ∈ I} T_s^I)
12             if privacy_breached then
13                 return false // early stop
14 return true
```

4.1 Anonymization Algorithm

We introduce a simple and general algorithm based on the Binary Space Partitioning (BSP) (Algorithm 7). Similar to the Mondrian algorithm, which is also a BSP 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 verification strategy for efficient verification. The pseudo code for our *provider-aware* anonymization algorithm is presented in Algorithm 7. We describe the algorithm details below.

Provider-Aware Partitioning. The algorithm first generates all possible splitting points, π, for QI attributes and data providers (lines 1 to 2 of the Algorithm 7). In addition to the multidimensional QI domain space, we consider the

Algorithm 7: 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 is m-private w.r.t. C

1 π = `get_splitting_points_for_attributes`(A_i)
2 $\pi = \pi \cup$ `get_splitting_point_for_providers`(A_0)
3 $\pi' = \{a_i \in \pi, i \in \{0, 1, \ldots, q\}$: `are_both_split_subpartitions_m-private`$(T, a_i)\}$
4 **if** π' *is* $\emptyset$ **then**
5 $\quad T^* = T^* \cup \mathcal{A}_1(T)$
6 $\quad$ **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

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 1**) 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 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 Verification for EG-Monotonic m-Privacy. m-Privacy is then verified for all possible splitting points, and only those satisfying m-privacy are added to a candidate set π' (line 3 of the Algorithm 7). 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 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 7), we use the privacy fitness score (Definition 3.1) to find the best splitting QI attribute and point (line 7). 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 (lines 8 and 9).

4.2 Secure Anonymization Protocol

The Algorithm 7 can be implemented and executed in a distributed environment by a trusted third-party (TTP) or by running a secure multi-party (SMC) protocol. In this section we present implementation of such protocol. The protocol is designed to run by semi-honest providers.

The key idea of the implementation is to embed existing SMC protocols. All parties running the protocol choose a leading provider P_1, which runs the Algorithm 8. Remaining providers run the Algorithm 9. To find the leading provider all providers run the leader election algorithm [26].

The most important step of the leading provider algorithm is to choose an attribute, which median is used to split records. The splitting attribute is chosen based on the fitness scores of record subsets. This process is repeated until no more possible splits can be found, i.e., any further split would return records that violate the privacy constraint. Median of each attribute is found by running the secure median protocol (line 4, [1]). All records with the current attribute values less than the median and some records with the attribute values equal to the median are in the set $T^{s,i}$. Remaining records are in the set $T^{g,i}$. Then m-privacy w.r.t. C is verified for $T^{s,i}$ by running the secure protocol (line 8). If $T^{s,i}$ is m-private w.r.t. C, then the same verification protocol is run for $T^{g,i}$ (line 11). If $T^{g,i}$ is also m-private w.r.t. C, then fitness scores of $T^{s,i}$ and $T^{g,i}$ are computed and compared with scores from the best split found so far. If the current split is better (greater fitness scores of $T^{s,i}$ and $T^{g,i}$), then it is taken as the new best split (lines 14 to 17). After checking all attributes and identifying the best one, the leading site runs the protocol independently for $T^{s,i_{max}}$ and $T^{g,i_{max}}$ (lines 19 to 21). If no such attribute can be found, and all other groups of records are already split, the protocol stops.

Other providers run the Algorithm 9 splitting their own records according to median values that have been sent to them from the leading provider. They also participate in other SMC protocols run by the leader.

Details of security and overhead analyses can be found in the Appendix E.

Secure Fitness Score Protocol. To compute the fitness score for a group of distributed records the leading provider runs an SMC protocol. For each privacy constraint C a new protocol need to be designed. However, computing fitness scores is usually a simple arithmetic task that can be easily and securely implemented. As an example we present SMC protocols for k-anonymity and l-diversity, which are used in our example (**Table 1**).

To compute the fitness score of all records the leading provider runs a secure sum protocol [5, 28], and gets the number of all records. The number is divided by k, and used to compute overall fitness score. To find the fitness score for l-diversity the leading provider runs secure size of set union protocol, which returns the number of distinct sensitive attribute values. The protocol is very similar to the secure size of set intersection [5] except the last step. In the last step, instead counting values that are present in all of the encrypted itemsets, the protocol counts values that are present in *any* encrypted itemset. Then, the result is

Algorithm 8: Secure *provider-aware* protocol run by the leading provider P_1.

Data: A set of records T_1, a set of QI attributes A_i $(i = 1, \ldots, q)$, m, a privacy constraint C.
Result: An anonymized view of distributed records $\mathcal{A}\left(\bigcup_{j=1}^{n} T_j\right)$ that is m-private w.r.t. C.

```
i_max = −1
f_max = 0
foreach i ∈ {0, ..., q} do
    Find the median value s_i of A_i in the set T (using secure median protocol).
    Send s_i and A_i to other providers.
    Split set T_1 into T_1^{s,i} = {t ∈ T_1 : t[A_i] < s_i}, and T_1^{g,i} = {t ∈ T_1 : t[A_i] > s_i}.
    Distribute median records among T_1^{s,i} and T_1^{g,i} to reduce uneven distribution of records.
    Verify m-privacy w.r.t. C of T^{s,i} = ⋃_{j=1}^{n} T_j^{s,i} (using secure m-privacy verification protocol Algorithm 5).
    if T^{s,i} is not m-private w.r.t. C then
        continue
    Verify m-privacy w.r.t. C of T^{g,i} = ⋃_{j=1}^{n} T_j^{g,i} (using secure m-privacy verification protocol Algorithm 5).
    if T^{g,i} is not m-private w.r.t. C then
        continue
    Compute fitness scores for T^{s,i} and T^{g,i} (using secure fitness score protocol).
    if f_max < min(f(T^{s,i}), f(T^{g,i})) then
        f_max = min(f(T^{s,i}), f(T^{g,i}))
        i_max = i
Send i_max to other providers.
if i_max ⩾ 0 then
    Run this protocol for T^{s,i_max}.
    Run this protocol for T^{g,i_max}.
```

Algorithm 9: Secure *provider-aware* protocol run by a non-leading provider $P_j (j > 1)$.

Data: A set of records T_j, a set of QI attributes A_i $(i = 1, \ldots, q)$, m, a privacy constraint C

```
Read the median value s_i and the attribute A_i from the leader P_1.
Split set T_j into T_j^{s,i} = {t ∈ T_j : t[A_i] < s_i}, and T_j^{g,i} = {t ∈ T_j : t[A_i] > s_i}.
Distribute median records among T_j^{s,i} and T_j^{g,i} to reduce uneven distribution of records.
Read i_max from the leader P_1.
if i_max ⩾ 0 then
    Run this protocol for T^{s,i_max}.
    Run this protocol for T^{g,i_max}.
```

divided by l, and used in the formula that computes the final score (**Equation 1**).

5. EXPERIMENTAL RESULTS FOR EG MONOTONIC PRIVACY CONSTRAINTS

We present two sets of experimental results for m-privacy w.r.t. EG monotonic constraint C with the following goals: 1) to compare and evaluate the different m-privacy verification algorithms given a group 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.

5.1 Experiment Setup

We used combined training and test sets of the Adult dataset[3]. Records with missing attribute values have been removed. All remaining 45,222 records have been used in 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 them, and report only those for the exponential distribution.

The privacy constraint is defined using k-anonymity (Definition A.1) and l-diversity (Definition A.2). We note again that m-privacy is orthogonal to the privacy constraints being used, and these are chosen to demonstrate the feasibility and efficiency of our approach. Both m-privacy verification and anonymization use privacy fitness scores (**Equation 1**), 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 (Appendix C) and values of α for the most efficient runs have been chosen as default values. All experiments, algorithm parameters, and their default values are listed in **Table 2**. All experiments are performed on Sun Microsystems SunFire V880 with 8 CPUs, 16 GB of RAM, and running Solaris 5.10.

Name	Description	Verification	Anonymization
α	Weight parameter	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	–	45222
$\|T_G\|$	Number of records in a group	{150, 750}	–
k	Parameter of k-anonymity	50	30
l	Parameter of l-diversity	4	4

Table 2: Experiment parameters and default values.

5.2 *m*-Privacy Verification

The objective of the first set of experiments is to evaluate the efficiency of different algorithms for m-privacy ver-

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

ification given a group 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. **Figure 6** shows the runtime with varying m for different heuristics for both groups of records.

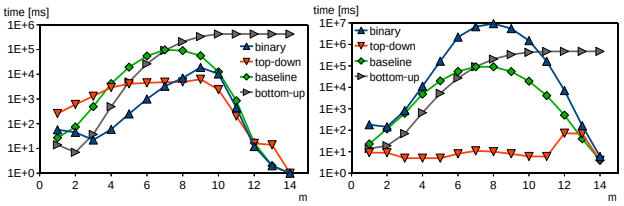

Figure 6: Runtime (logarithmic scale) vs. m.

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. In most cases the *binary* algorithm achieves the 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 an m-adversary will be able to breach privacy, and the downward pruning can be applied often. This intuition is confirmed by the result, which shows that the *top-down* algorithm is significantly better than other heuristics. Since the remaining algorithms do not rely only 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 analyze 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.

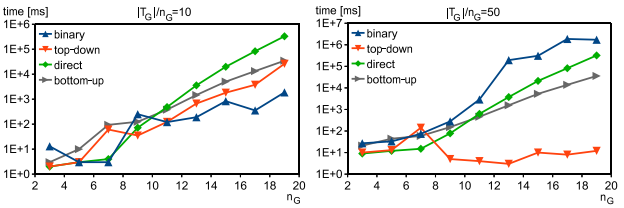

Figure 7: Runtime (logarithmic scale) vs. number of data providers.

We observe that increasing the number of contributing data providers has different impact on different algorithms in both 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 group of records has a low privacy fitness score, and is very vulnerable to attacks from m-adversaries. Increasing number of providers will make the domain of possible m-adversaries exponentially bigger.

Similar trend is found for the large group with higher number of records per provider for *binary*, *direct*, and *bottom-up* algorithms. However, for the *top-down* algorithm runtime grows linearly with the number of providers. This is due to its effective use of downward pruning.

The Average Number of Records Per Provider. In this experiment we systematically evaluated the impact of the 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.

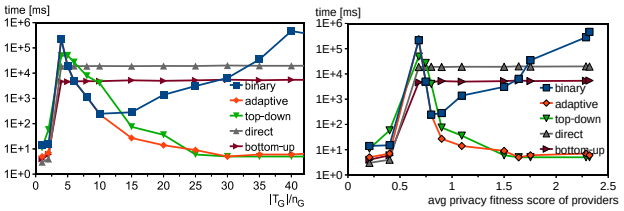

Figure 8: Runtime vs. $|T_G|/n_G$ and the average fitness score of providers.

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

Adaptive Strategy. Based on the above results, we use 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 use the *binary* algorithm, while for other cases the *top-down* is our choice.

5.3 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 [32, 6]. 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.

SELECT t FROM T^* WHERE p_1 AND ... AND p_{qd};

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

Attack Power. We first evaluate and compare 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 longer. However, in each iteration of the *provider-aware* algorithm, the additional splitting point along data providers, if chosen, reduces the number of providers for each 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 $m \approx n/2$.

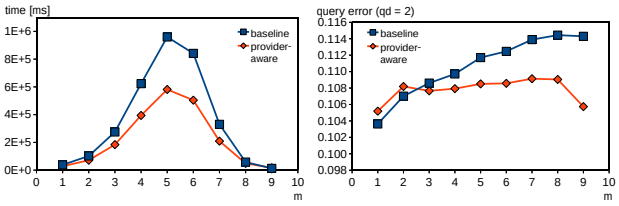

Figure 9: Runtime and the query error for different powers of m-privacy.

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 observed 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 providers in each group. In addition, the query error is at the same rate for both algorithms.

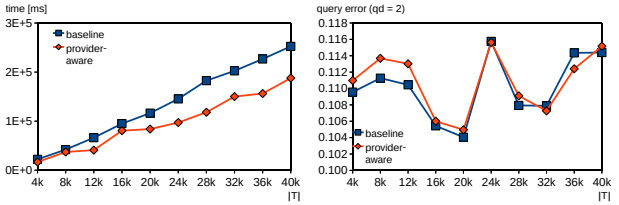

Figure 10: Runtime and query error vs. $|T|$.

Adaptive m-Privacy Verification. In this experiment we evaluate 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.

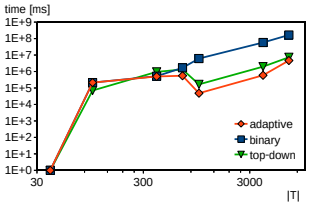

Figure 11: Runtime of adaptive and non-adaptive m-privacy verifications vs. $|T|$ (logarithmic scale).

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).

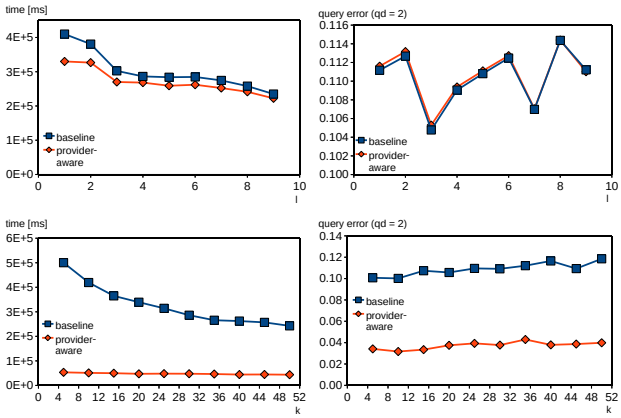

Figure 12: Runtime and query errors vs. k in k-anonymity and l in l-diversity used in m-privacy

As expected, an increasing value of k or l will require more records or more distinct values of sensitive attribute in each QI group, and thus results in higher query error. However, execution times are shorter comparing to *weaker* privacy constraints because less partitioning is needed.

6. EXPERIMENTAL RESULTS FOR NON-EG MONOTONIC PRIVACY CONSTRAINTS

In this section we present results of similar experiments as above but for the m-privacy w.r.t. a non-EG monotonic constraint C. Our goal is to compare and evaluate the verification algorithm (Algorithm 4) as well as the anonymization algorithm (Algorithm 7) for a non-EG monotonic privacy constraint C. Direct comparison of experimental results between EG and non-EG monotonic scenarios is very difficult due to significant differences in their complexities.

6.1 Experiment Setup

As a dataset we used the Adult dataset described previously (Section 5.1). However, due to high time complexity we sampled the dataset. Records have been distributed among n data providers $P_1, \ldots, P_n$ randomly, and with uniform probability.

The privacy constraint C is defined as a conjunction of k-anonymity (Definition A.1) and t-closeness (Definition A.3). Since t-closeness is not EG monotonic, then the m-privacy w.r.t. C is not monotonic as well (Theorem 2.1). Because of that both adaptive ordering of data providers and pruning strategies cannot be used. Thus, the weight parameter α has no impact to the performance of the verification algorithm. Due to high complexity of m-privacy verification and anonymization algorithms we run experiments using small sample of data records. Default values of parameters for all experiments are listed in **Table 3**.

Name	Description	Verification	Anonymization
α	Weight parameter	0.3	0.8
m	Power of m-privacy	3	1
n	Total number of data providers	–	5
n_G	Number of providers contributing to a group	5	–
$\|T\|$	Total number of records	–	30
$\|T_G\|$	Number of records in a group	25	–
k	Parameter of k-anonymity	4	4
t	Parameter of t-closeness	0.5	0.5

Table 3: Parameters and their default values for experiments with non-EG monotonic constraint C.

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

6.2 *m*-Privacy Verification

The objective of this set of experiments is to evaluate the efficiency and complexity of the m-privacy verification algorithm given a group of records T_G with respect to the previously defined non-EG monotonic constraint C.

Attack Power. In this experiment we compare the impact of the coalition size to the runtime for different sets of records. We used four different sets of records generated randomly and independently from each other. **Figure 13** shows the runtime with varying m for different sets of records.

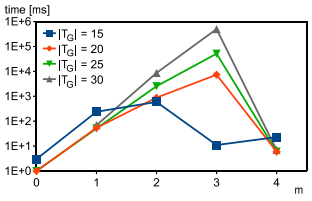

Figure 13: Runtime (logarithmic scale) vs. m for non-EG monotonic C.

The first group counts 15 records and is 2-private. As expected the runtime is increasing with m for $m \leqslant 2$, while for $m > 2$ time is much shorter. Significant speedup of verification for the latter case is caused by relatively quick finding a set of attacking records that can breach privacy. Remaining checks are then redundant, and the algorithm stops.

Other groups have 20, 25, and 30 records, and all of them are 3-private. Similar as for the first group the runtime increases exponentially with m for $m \leqslant 3$. The magnitude of the exponential growth is proportional to number of records in the set. All three sets of records verifies 4-privacy (with negative answers) in a very short time, because sets of attacking records that are used for successful attacks are identified early.

Number of Contributing Data Providers. In this experiment we analyze the impact of increasing number of data providers while preserving the average number of records per provider. **Figure 14** shows the runtime with varying n_G for different average numbers of records per provider.

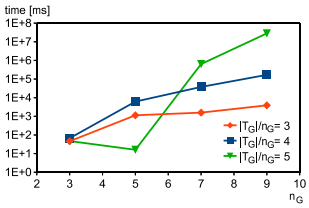

Figure 14: Runtime (logarithmic scale) vs. n_G for non-EG monotonic C.

Because $m = 3$, then for all scenarios with 3 providers records are not m-private, which is detected early. This is our baseline for experiments with more providers. Scenarios with the average number of records per provider equal to 3 and 4 are 3-private for 5 and more providers. As we have expected increasing number of providers (and records) will increase the runtime exponentially. For the last scenario, i.e., when the average number of records per provider is equal to 5, records are not 3-private for less than 6 providers. Luckily, for these cases sets of attacking records that would breach privacy have been found early. In the worst-case scenario the last set of attacking records that is checked, would breach privacy, and the runtime would be the same as if m-privacy holds. For 3-private sets of records increasing n_G causes exponential growth in the runtime. This results confirm our intuition and were expected.

6.3 Anonymization for *m*-Privacy

In this set of experiments we analyze performance of the m-privacy anonymization algorithm w.r.t. a non-EG monotonic constraint C for different parameter values. The constraint is defined as a conjunction of k-anonymity (Definition A.1) and t-closeness (Definition A.3). Both anonymization algorithms (*provider-aware* and *baseline*) are the same as for the EG monotonic privacy. Similar as for experiments with EG monotonic constraints we evaluated the utility of the anonymized data using the query error metric. To quantify the error we used 2,500 randomly generated queries with two predicates ($qd = 2$). Time complexity of m-privacy verification does not allow us to test large datasets. Thus, the absolute query error values may be high (above 0.1), but it is enough to evaluate the *provider-aware* algorithm.

Attack Power. We first evaluate our algorithm varying

size of possible malicious coalitions. **Figure 15** shows the runtime and query errors for different levels of privacy restrictiveness (parameter m).

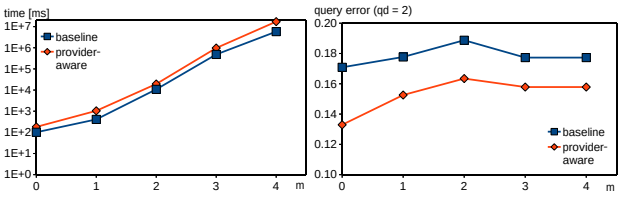

Figure 15: Runtime (logarithmic scale) and the query error for different powers of m-privacy.

As expected runtimes for both algorithms increase exponentially while increasing m. The *provider-aware* algorithm needs more time to run, which is expected due to considering additional splits of records. However, the query error is significantly lower for data anonymized by our algorithm.

Increasing restrictiveness of the m-privacy should result in anonymized data with higher query error values. This intuition is confirmed for lower values of m, but for $m > 2$ query error slightly decreases, and maintains the same level. This counter-intuitive behavior is a side effect of the dataset size. Due to complexity experiments with large number of records are not possible, and for small set of records it is very likely to get an anonymized 3-private dataset, which is also 4-private. The same reason stays behind higher error rate for 2-private anonymized dataset.

Number of Data Records. The goal of this experiment is to evaluate our algorithm for different number of records. Each set of records contains all records from the previous (smaller) set and a few more chosen randomly records. **Figure 16** presents runtime and query error for different numbers of anonymized records.

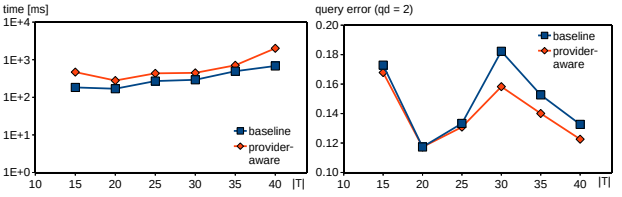

Figure 16: Runtime (logarithmic scale) and the query error for different number of anonymized records T.

As expected runtimes increase exponentially with number of records. Dynamics of these growths, which are represented by slopes of lines in the chart, is correlated with m. The greater the m is, the more dynamically runtime increases for both algorithms.

Query error is less dependent from the size of anonymized dataset. At the first glance this seems to be counter-intuitive, but it can be easily explained. Adding records increases query error, but only up to some point, after which the anonymization algorithm generates more QI groups, and the precision of query answers increases. Thus, the query error is more a periodic-like function of the dataset size.

For all scenarios the *provider-aware* algorithm performs at least as good as the *baseline* algorithm, and for larger datasets its results have lower query errors.

Privacy Constraints. In these experiments we evaluate our algorithm for different restrictivenesses of privacy constraints, i.e., k-anonymity and t-closeness. Notice that the restrictiveness of k-anonymity is proportional to values of k (i.e., increasing k makes k-anonymity more restrictive), while t-closeness is disproportional to values of t (i.e., increasing t makes t-closeness less restrictive). **Figure 17** shows runtime and query error while varying k and t values.

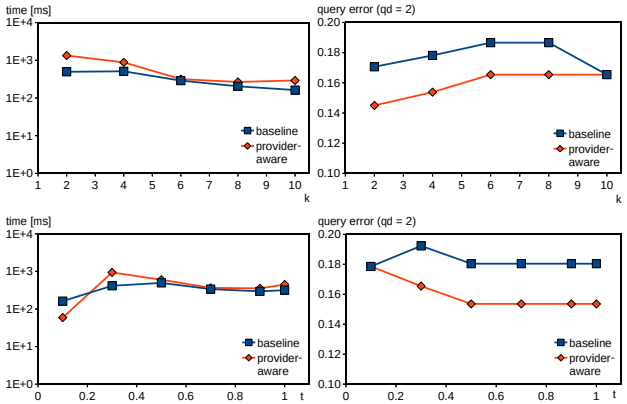

Figure 17: Runtime (logarithmic scale) and query errors vs. k in k-anonymity and t in t-closeness used as a privacy constraint C.

We expect that increasing k will reduce runtime due to less splits of records that can be performed. At the same time the query error will increase because of bigger equivalence groups in the anonymized data. Our expectations are generally confirmed by experiments. Only for $k \geqslant 6$ query errors are constant or decrease (*baseline* for $k = 10$). The reason of that is, again, small size of the dataset. After some number of splits, further splits are not possible due to privacy constraints, and obtained QI groups meet more restrictive privacy constraints than required. Thus, query errors as well as runtimes for those scenarios would be the same. Notice that modifying k impacts privacy fitness scores and choices of attributes used to split records, hence for $k = 10$ the *baseline* algorithm runs in a different way than for $k = 9$, and returns results with lower query errors.

Experiments also confirmed that our algorithm performs better than the *baseline* algorithm in almost all scenarios. The price for getting higher utility is just a slight increase of the runtime.

7. RELATED WORK

Privacy preserving data analysis and publishing has received considerable attention in recent years [7, 9, 8]. Most work has focused on a single data provider setting and considered the data recipient as an attacker. A large body of literature [9] assumes limited background knowledge of the attacker, and defines privacy using relaxed *adversarial* notion [21] by considering specific types of attacks. Representative principles include k-anonymity [27, 29], l-diversity [21], and t-closeness [19]. Few recent works have modeled the instance level background knowledge as corruption, and studied perturbation techniques under these weak 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 [7, 8] 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.

There are some work focused on anonymization of distributed data. [12, 14, 24] studied distributed anonymization for vertically partitioned data using k-anonymity. Zhong et al. [36] studied classification on data collected from individual *data owners* (each record is contributed by one data owner) while maintaining k-anonymity. Jurczyk et al. [15] proposed a notion called l'-site-diversity to ensure anonymity for data providers in addition to privacy of the data subjects. Mironov et al. [22] studied SMC techniques to achieve differential privacy. Mohammed et al. [23] 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 when a weak privacy notion is used.

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 frequency count, both upward and downward pruning are possible. Taking advantage of the *dual-pruning* is an essential point of the algorithm presented in [3]. 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 a group of records for EG monotonic privacy requires finding out if all m-coalitions are not able to compromise privacy of remaining records (Observation 2.3). After simple modifications (e.g., not using *early-stop*) our algorithm can be used to find frequent itemsets and the dual-pruning algorithm can be used to verify m-privacy, but in both cases they will not be very efficient.

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. To prevent privacy disclosure, we introduced a new notion, m-privacy, to ensure that any m-adversary or any of its subsets is not able to compromise the privacy. The notion has been extensively analyzed for different groups of privacy constraints in order to take advantage of them. Initial results describing m-privacy w.r.t. EG monotonic constraints for a TTP setting have been already published [11].

We presented heuristic algorithms exploiting the EG monotonicity of privacy constraints and adaptive ordering techniques for efficiently checking m-privacy. For non-EG monotonic constraints we showed an algorithm, with minimal required number of privacy checks in order to verify m-privacy. We also presented a *provider-aware* anonymization algorithm with adaptive m-privacy checking strategies to ensure high utility and m-privacy of anonymized data. Our experiment results showed that our approach achieves better or comparable utility than existing algorithms while achieving m-privacy efficiently regardless monotonicity of privacy constraint.

For all verification and anonymization algorithms we have presented implementations for two distributed data scenarios, with and without trusted third party. In the first scenario implemented code can be run directly by a trusted third party (TTP), while in the latter case, we presented secure multi-party scenarios (SMC). All protocols have been presented in details and their security and complexity carefully analyzed. Implementations of algorithms for the TTP setting is available on-line for further development and deployments[4].

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

9. REFERENCES

[1] G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the kth-ranked element. In *In Avdances in Cryptology - Proc. of Eurocyrpt '04*, pages 40–55, 2004.

[2] G. Boros and V. Moll. *Irresistible Integrals: Symbolics, Analysis and Experiments in the Evaluation of Integrals*. Cambridge University Press, 2004.

[3] 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 Intl. Conf. on Knowledge Discovery and Data Mining*, KDD '02, pages 42–51, 2002.

[4] 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.

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

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

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

[8] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54:86–95, January 2011.

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

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

[11] S. Goryczka, L. Xiong, and B. C. M. Fung. m-privacy for collaborative data publishing. In *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, CollaborateCom '11, pages –, 2011.

[12] W. Jiang and C. Clifton. Privacy-preserving distributed k-anonymity. In *Data and Applications Security XIX*, volume 3654 of *Lecture Notes in Computer Science*, pages 924–924. Springer Berlin / Heidelberg, 2005.

[13] W. Jiang and C. Clifton. A secure distributed framework for achieving k-anonymity. *VLDB J.*, 15(4):316–333, 2006.

[14] W. Jiang and C. Clifton. A secure distributed framework for achieving k-anonymity. *The VLDB Journal*, 15(4):316–333, 2006.

[15] P. Jurczyk and L. Xiong. Distributed anonymization: Achieving privacy for both data subjects and data providers. In *DBSec*, pages 191–207, 2009.

[4] http://www.mathcs.emory.edu/aims/m-anonymizer/

[16] D. Kifer. Attacks on privacy and definetti's theorem. In *Proc. of the 35th SIGMOD Intl. Conf. on Management of Data*, SIGMOD '09, pages 127–138. ACM, 2009.
[17] D. Kifer and A. Machanavajjhala. No free lunch in data privacy. In *Proc. of the 2011 Intl. Conf. on Management of Data*, pages 193–204, 2011.
[18] K. Lefevre, D. J. Dewitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, 2006.
[19] 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.
[20] Y. Lindell and B. Pinkas. Secure multiparty computation for privacy-preserving data mining. *The Journal of Privacy and Confidentiality*, 1(1):59–98, 2009.
[21] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, page 24, 2006.
[22] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan. Computational differential privacy. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 126–142, 2009.
[23] 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)*, 4(4):18:1–18:33, October 2010.
[24] 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*, EDBT, pages 228–239, 2009.
[25] P. Paillier and D. Pointcheval. Efficient public-key cryptosystems provably secure against active adversaries. In *Advances in Cryptology - ASIACRYPT'99*, volume 1716 of *Lecture Notes in Computer Science*, pages 165–179. Springer Berlin / Heidelberg, 1999.
[26] A. Russell and D. Zuckerman. Perfect information leader election in $\log^* n + O(1)$ rounds. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, FOCS '98, pages 576–, 1998.
[27] P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. Knowl. Data Eng.*, 13(6):1010–1027, 2001.
[28] R. Sheikh, B. Kumar, and D. K. Mishra. A distributed k-secure sum protocol for secure multi-party computations. *Journal of Computing*, 2:68–72, March 2010.
[29] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, 2002.
[30] 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*, pages 725–734. IEEE Computer Society, 2008.
[31] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *J. Comput. Secur.*, 13:593–622, July 2005.
[32] X. Xiao and Y. Tao. Anatomy: simple and effective privacy preservation. In *Proc. of the 32nd Intl. Conf. on Very Large Data Bases*, VLDB '06, pages 139–150, 2006.
[33] Y. Xiao, L. Xiong, and C. Yuan. Differentially private data release through multidimensional partitioning. In *Proceedings of the 7th VLDB conference on Secure data management*, SDM'10, pages 150–168, 2010.
[34] B. Yang, H. Nakagawa, I. Sato, and J. Sakuma. Collusion-resistant privacy-preserving data mining. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 483–492, 2010.
[35] B. Zhang. Generic constant-round oblivious sorting algorithm for mpc. In *Proceedings of the 5th international conference on Provable security*, ProvSec'11, pages 240–256, 2011.
[36] S. Zhong, Z. Yang, and R. N. Wright. Privacy-enhancing k-anonymization of customer data. In *PODS '05: Proc. of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 139–147, 2005.

APPENDIX

A. PRIVACY CONSTRAINT DEFINITIONS

A privacy constraint may be defined by a conjunction of other privacy constraints. For example, k-anonymity, l-diversity, and t-closeness are used to define privacy for our experiments. For all definitions assume that T^* is a set of anonymized records.

DEFINITION A.1. (k-ANONYMITY [29]) *A set of records T^* satisfies k-anonymity if and only if every group of records with the same values of all quasi-identifier attributes contains at least k records.*

DEFINITION A.2. (l-DIVERSITY [21]) *A set of records T^* satisfies l-diversity if for every group q^* of records with the same values of all quasi-identifier attributes number of distinct values of sensitive attribute is not less than l.*

DEFINITION A.3. (t-CLOSENESS [19]) *A set of records T^* satisfies t-closeness if for every group q^* of records with the same values of all quasi-identifier attributes the distance between the distribution of a sensitive attribute in this group, and the distribution of the attribute in the whole table is no more than a threshold t.*

B. MONOTONICITY OF *m*-PRIVACY

In this section we prove Theorem 2.1 for both EG monotonicity and generalization monotonicity.

PROOF FOR EG MONOTONICITY. Assume T is a set of records provided by $P = \{P_1, \ldots, P_n\}$ providers, and $\mathcal{A}$ is an anonymization mechanism that returns records, which are m-private w.r.t. C ($0 \leqslant m \leqslant n-1$). Thus, $\mathcal{A}(T)$ is m-private w.r.t. C, and $C(\mathcal{A}(T)) = true$.

Suppose m-privacy w.r.t. C is EG monotonic, and let $\widetilde{T}$ be a superset of T. Then based on the definition of EG monotonicity (Definition 2.3), and fulfillment of m-privacy w.r.t. C by $\mathcal{A}(T)$, $\mathcal{A}(\widetilde{T})$ is m-private w.r.t. C as well. In particular, $\mathcal{A}(\widetilde{T})$ is 0-private w.r.t. C (Observation 2.1), i.e., $\mathcal{A}(\widetilde{T})$ fulfills C, and thus C is EG monotonic.

Conversely, suppose C is an EG monotonic privacy constraint applied to the definition of m-privacy, and let $\widetilde{T}$ be a superset of T. Assume $I \subset P$ is a coalition of m attackers providing T_I ($T_I \subset T \subset \widetilde{T}$) records, $|I| = m$, and $\mathcal{A}(T)$ is m-private w.r.t. C, then $C(\mathcal{A}(T \setminus T_I)) = true$. Furthermore, $T \subset \widetilde{T}$ implies that $\mathcal{A}(T) \subset \mathcal{A}(\widetilde{T})$, and $\mathcal{A}(T \setminus T_I) \subset \mathcal{A}(\widetilde{T} \setminus T_I)$, which together with EG monotonicity of C, show that $C(\mathcal{A}(\widetilde{T} \setminus T_I)) = true$. Thus, $\mathcal{A}(\widetilde{T})$ is m-private w.r.t. C, and we conclude that m-privacy is EG monotonic. □

PROOF FOR GENERALIZATION MONOTONICITY. Assume n providers $P = \{P_1, \ldots, P_n\}$ contributed two sets of records T_1 and T_2 ($T_1 \cap T_2 = \emptyset$), and $\mathcal{A}$ is an anonymization mechanism that returns anonymized records, which are m-private w.r.t. C ($0 \leqslant m \leqslant n-1$). Thus, $\mathcal{A}(T_1)$, and $\mathcal{A}(T_2)$ are m-private w.r.t. C.

Suppose m-privacy w.r.t. C is generalization monotonic. Then based on the definition of generalization monotonicity (Definition 2.2), $\mathcal{A}(T_1 \cup T_2)$ is m-private w.r.t. C. In particular, $\mathcal{A}(T_1) \cup \mathcal{A}(T_2)$ is 0-private w.r.t. C (Observation 2.1), i.e., $C(\mathcal{A}(T_1) \cup \mathcal{A}(T_2)) = true$. Because $C(\mathcal{A}(T_1)) = true$ and $C(\mathcal{A}(T_2)) = true$, then C is generalization monotonic.

Conversely, suppose C is a generalization monotonic privacy constraint applied to the definition of m-privacy. Assume $I \subset P$ is a coalition of m attackers providing T_I ($T_I \subset T_1 \cup T_2$) records. $|I| = m$, and $\mathcal{A}(T_1)$, and $\mathcal{A}(T_2)$

are m-private w.r.t. C, then $C(\mathcal{A}(T_1 \setminus T_I)) = true$, and $C(\mathcal{A}(T_2 \setminus T_I)) = true$. Furthermore, generalization monotonicity of C implies that $C(\mathcal{A}(T_1 \setminus T_I) \cup \mathcal{A}(T_2 \setminus T_I)) = true$. Thus, $\mathcal{A}(T_1) \cup \mathcal{A}(T_2)$ is m-private w.r.t. C, and we conclude that m-privacy is generalization monotonic. □

C. ADDITIONAL EXPERIMENTS

In this section we present a set of additional experiments that show the impact of the parameter α in the definition of the fitness score (**Equation 1**).

Privacy Fitness Score Parameter for Verification Algorithms. Privacy fitness score is used to adaptively order the adversaries for m-privacy verification, and is defined in the **Equation 1** with parameter α. **Figure 18** shows the runtime with varying α for different verification heuristics for EG monotonic constraint C.

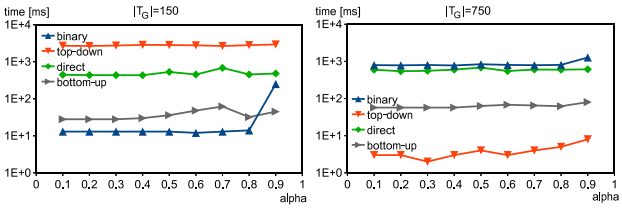

Figure 18: Runtime vs. α for m-privacy verification heuristics w.r.t. EG monotonic C.

Different values of α impact the adaptive ordering of adversaries that may generate a different sequence of m-adversaries for verification. This sequence is crucial for groups with low privacy fitness scores, which are likely to violate m-privacy. Our intuition is confirmed by the results, where values of α affect the overall runtime only for the small group, where downward pruning is not used so often. The optimal value of α lies around 0.3. For the large group, α does not have a significant impact on the runtime. As a result, we use $\alpha = 0.3$ for all verification experiments.

Privacy Fitness Score Parameter for Anonymization Algorithms. Privacy fitness score is also used to find the best splitting point in each iteration of the anonymization algorithm, and is also defined in **Equation 1**. **Figure 19** shows the runtime and query error with varying α for both anonymization algorithms for EG monotonic constraint C.

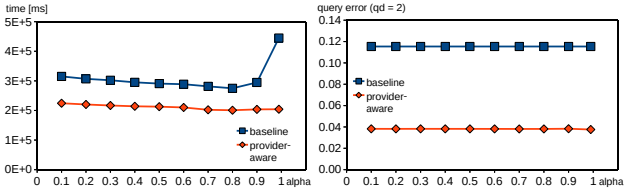

Figure 19: Query error and runtime vs. α of the *provider-aware* algorithm for m-privacy w.r.t. EG monotonic C.

Based on our definition (**Equation 1**), the greater the α is, the more weight is set to the diversity component. For our settings, we observe that values of α close to 0.8 give the best results in terms of utility and efficiency. Thus, for the anonymization algorithm, we set $\alpha = 0.8$. This result is empirical, but since runtimes do not vary significantly for different values of α in our algorithm, setting this parameter based only on experiments is reasonable.

Privacy Fitness Score Parameter for Anonymization with non-EG Privacy Constraint C. The same experiment as above has been repeated for a non-EG monotonic constraint C (k-anonymity and l-diversity). Our goal is to check the impact of parameter α to runtimes and query errors of anonymization algorithms. **Figure 20** presents results of our experiments, where the same set of records has been anonymized many times with different values of α.

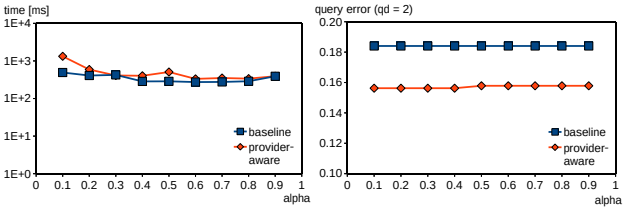

Figure 20: Query error and runtime vs. α for a non-EG monotonic privacy constraint C.

Values of the parameter α has no significant impact on m-privacy anonymization time and quality of its results. However, experiments confirm that *provider-aware* algorithm requires more time to run, but returns anonymized data with noticeable lower query errors.

D. THE AVERAGE TIME COMPLEXITY ANALYSIS OF *m*-PRIVACY VERIFICATION ALGORITHMS

Computing average time complexity for majority algorithms is very difficult. Number of factors that impact computation time is significant. One of the most important is time complexity of a single privacy check, which is different for each privacy constraint. We assume that it takes a single unit of time to verify privacy of any set of anonymized records. Formally, this assumption is modeled by an oracle that verifies privacy in constant time.

Since finding the exact average time complexity depends on many factors, we estimate the lower bound of the average complexity time E. For a set of anonymized records T^* provided by n_G parties let all adversary coalitions that breaches m-privacy w.r.t. C be supersets of a single x-adversary X, which breaches privacy as well. If such coalition does not exist, then we take $x = n_G$. Values of x, m, and n_G are parameters in our computations. The number of all possible x-adversaries for $x < m$ is equal to $\sum_{x=0}^{m-1} \binom{n_G}{x}$, and for $x \geqslant m$ is equal to $\sum_{x=m}^{n_G-1} \binom{n_G}{x}$. The number of all privacy checks for an x-adversary ($x < m$) is denoted by $H_m(x)$ and for $x \geqslant m$ by $H_M(x)$,

$$E(m, n_G) = \frac{\sum_{x=0}^{m-1} \binom{n_G}{x} H_m(x) + \sum_{x=m}^{n_G-1} \binom{n_G}{x} H_M(x)}{2^{n_G}} \quad (3)$$

The *top-down* Algorithm. For ($x \geqslant m$) anonymized records T^* are m-private w.r.t. C, and the *top-down* verifies all ($n_G - 1$)-coalitions applying downwards pruning when possible, and also checks all coalitions that violate privacy. After all those checks m-privacy w.r.t. C is verified. Number

of privacy checks for a single value of x follows the formula:

$$\begin{aligned} H_m(x) &= \sum_{i=1}^{n_G-x-2} \binom{n_G-x}{i} + n_G - 1 \\ &= \sum_{i=0}^{n_G-x} \binom{n_G-x}{i} + x - 3 \\ &= 2^{n_G-x} + x - 3 \end{aligned} \quad (4)$$

When $x < m$ anonymized records T^* are not m-private w.r.t. C. Similar like for the above case the *top-down* algorithm verifies all $(n-1)$-coalitions, and those with more than m providers that breaches privacy. Then it checks a single m-adversary that breaches privacy (m-adversaries that do not breach privacy have been already pruned) and stops. Thus, for a single x-adversary the number of privacy checks can be computed using the same formula as above after subtracting coalitions of size smaller or equal to m plus one check for the m-coalition.

$$H_M(x) = 2^{n_G-x} + x - 2 - \sum_{i=1}^{m-x} \binom{n_G-x}{i} \quad (5)$$

Thus, the lower bound of the average number of privacy checks follows,

$$\begin{aligned} E(m,n_G) &= \left[\sum_{x=0}^{m-1} \binom{n_G}{x} H_m(x) \right. \\ &\quad \left. + \sum_{x=m}^{n_G-1} \binom{n_G}{x} H_M(x)\right] /2^{n_G} \\ &= \left[2^{n_G} \cdot \sum_{x=0}^{n_G-1} \binom{n_G}{x} \cdot 2^{-x} \right. \\ &\quad + \sum_{x=0}^{n_G-1} \binom{n_G}{x} x - 2\sum_{x=0}^{n_G-1} \binom{n_G}{x} - \sum_{x=0}^{m-1} \binom{n_G}{x} \\ &\quad \left. - \sum_{x=0}^{m-1} \binom{n_G}{x} \sum_{i=1}^{m-x} \binom{n_G-x}{i}\right] /2^{n_G} \end{aligned} \quad (6)$$

Then applying the binomial theorem $\sum_{x=0}^{n_G} \binom{n_G}{x} r^x = (1+r)^{n_G}$ for $r = 1/2$, and the following formula, $\sum_{k=0}^{n_G} k\binom{n_G}{k} = 2^{n_G-1} n_G$ [2] we obtain,

$$\begin{aligned} E(m,n_G) &= \left[2^{n_G} \cdot \left((3/2)^{n_G} - 2^{-n_G}\right) \right. \\ &\quad + 2^{n_G-1} n_G - n_G - 2(2^{n_G}-1) \\ &\quad \left. - \sum_{x=0}^{m-1} \binom{n_G}{x} \left(1 + \sum_{i=1}^{m-x} \binom{n_G-x}{i}\right)\right] /2^{n_G} \\ &= (3/2)^{n_G} + \frac{n_G}{2} - 2 - \frac{n_G-1}{2^{n_G}} \\ &\quad - \frac{1}{2^{n_G}} \sum_{x=0}^{m-1} \binom{n_G}{x} \left(1 + \sum_{i=1}^{m-x} \binom{n_G-x}{i}\right) \end{aligned} \quad (7)$$

Values of $E(m,n_G)$ vary for different m, for example,

$$\begin{aligned} E(1,n_G) &= (3/2)^{n_G} + n_G/2 + n_G 2^{1-n_G} - 2 \\ &= O((3/2)^{n_G}) \end{aligned} \quad (8)$$

$$\begin{aligned} E(n_G-1,n_G) &= (3/2)^{n_G} + n_G/2 - 2 - \frac{n_G-1}{2^{n_G}} \\ &\quad -2^{-n_G} \sum_{x=0}^{n_G-1} \binom{n_G}{x} \left(2^{n_G-x} - 1\right) \\ &= (3/2)^{n_G} + n_G/2 - 2 - \frac{n_G-1}{2^{n_G}} \\ &\quad - \sum_{x=0}^{n_G-1} \binom{n_G}{x} \cdot 2^{-x} \\ &\quad + 2^{-n_G} \sum_{x=0}^{n_G-1} \binom{n_G}{x} \\ &= (3/2)^{n_G} + n_G/2 - 2 - \frac{n_G-1}{2^{n_G}} \\ &\quad -(3/2)^{n_G} + 2^{-n_G} + 1 - 2^{-n_G} \\ &= n_G/2 - 1 - \frac{n_G-1}{2^{n_G}} \\ &= O(n_G) \end{aligned} \quad (9)$$

The lower bound of the average time complexity varies from linear to exponential for different values of m. Thus, for the *top-down* algorithm m is a significant parameter of the expected average computation time.

The *bottom-up* Algorithm. Similar like for the *top-down* algorithm we compute the lower bound of the average complexity time for the *bottom-up* algorithm. The same assumptions hold. When $x \geqslant m$, then all coalitions of up to m providers need to be considered, and $H_M(x) = \sum_{i=0}^{m} \binom{n_G}{i}$. For $x < m$, the algorithm verifies all coalitions with up to $(x-1)$ providers, and some coalitions with x adversaries. Since X can be any x-adversary with equal probability, then on average half of x-adversaries will be checked before finding the one that breaches privacy. Thus, number of privacy checks is equal to $H_m(x) = \sum_{i=0}^{x-1} \binom{n_G}{i} + \binom{n_G}{x}/2$, and the lower bound of the average number of privacy checks follows,

$$\begin{aligned} E(m,n_G) &= 2^{-n_G} \sum_{x=m+1}^{n_G-1} \binom{n_G}{x} \sum_{i=0}^{m} \binom{n_G}{i} \\ &\quad + 2^{-n_G} \sum_{x=1}^{m} \binom{n_G}{x} \left[\sum_{i=0}^{x-1} \binom{n_G}{i} + \binom{n_G}{x}/2\right] \end{aligned} \quad (10)$$

$E(m,n_G)$ varies a lot for different m, for example,

$$\begin{aligned} E(1,n_G) &= n_G + 1 - \frac{(n_G+1)^2 + 2}{2^{n_G+1}} \\ &= O(n_G) \end{aligned} \quad (11)$$

$$
\begin{aligned}
E(n_G-1,n_G) &> 2^{-n_G}\cdot\sum_{x=1}^{n_G-1}\binom{n_G}{x}\binom{n_G}{x}/2\\
&= 2^{-n_G-1}\cdot\left[\sum_{x=0}^{n_G}\binom{n_G}{x}^2-2\right]\\
&= 2^{-n_G-1}\cdot\binom{2n_G}{n_G}-2^{-n_G}\\
&\geqslant 2^{-n_G-1}\frac{4^{n_G}}{\sqrt{4n_G}}-2^{-n_G}\\
&= \frac{2^{n_G}}{4\sqrt{n_G}}-2^{-n_G}\\
&= O\left(2^{n_G}n_G^{-1/2}\right) \qquad (12)
\end{aligned}
$$

The *direct* Algorithm. Similar like for above algorithms we compute the lower bound of the average complexity time for the *direct* algorithm. The same assumptions as above hold. In addition, we assume that all m-adversaries are checked in a random order. When $x \geqslant m$, then all m-adversaries need to be considered, hence $H_M(x)=\binom{n_G}{m}$.

For $x<m$, the algorithm verifies m-adversaries until finding one that breaches privacy. Among $\binom{n_G}{m}$ m-adversaries there are $\binom{n_G-x}{m-x}$ that can breach privacy. Thus, assuming independence of privacy verifications, probabilities of not breaching privacy p and breaching privacy q follow formulas,

$$
\begin{aligned}
p &= 1-\frac{\binom{n_G-x}{m-x}}{\binom{n_G}{m}} \qquad (13)\\
q &= \frac{\binom{n_G-x}{m-x}}{\binom{n_G}{m}} \qquad (14)
\end{aligned}
$$

Given x the average number of privacy checks $H_m(x)$ follows the formula,

$$
\begin{aligned}
H_m(x) &= q+2pq+3p^2q+\ldots+\binom{n_G}{m}p^{\binom{n_G}{m}-1}q\\
&= q\sum_{k=1}^{\binom{n_G}{m}}kp^{k-1}\\
&= q\frac{(p-1)\binom{n_G}{m}p^{\binom{n_G}{m}}-p^{\binom{n_G}{m}}+1}{(p-1)^2} \qquad (15)
\end{aligned}
$$

The overall average number of privacy checks follows the **Equation 3**, and can be simplified,

$$
\begin{aligned}
E(m,n_G) &= \frac{\sum_{x=0}^{m-1}\binom{n_G}{x}H_m(x)+\sum_{x=m}^{n_G-1}\binom{n_G}{x}H_M(x)}{2^{n_G}}\\
&= \sum_{x=0}^{m-1}\binom{n_G}{x}\frac{1-\left(\binom{n_G-x}{m-x}+1\right)p^{\binom{n_G}{m}}}{2^{n_G}q}\\
&\quad+\binom{n_G}{m}\left(1-\frac{\sum_{x=0}^{m-1}\binom{n_G}{x}+1}{2^{n_G}}\right) \qquad (16)
\end{aligned}
$$

For $m=1$ values of p, q, and $E(1,n_G)$ follow formulas,

$$
\begin{aligned}
p &= 1-\frac{\binom{n_G-x}{1-x}}{n_G} \qquad (17)\\
q &= \frac{\binom{n_G-x}{1-x}}{n_G} \qquad (18)\\
E(1,n_G) &= \frac{1-(n_G+1)\cdot 0^{n_G}}{2^{n_G}}+n_G(1-2^{1-n_G})\\
&= n_G+\frac{1-2n_G}{2^{n_G}}\\
&= O(n_G) \qquad (19)
\end{aligned}
$$

For $m=n_G/2$ values of p, q, and $E(n_G/2,n_G)$ follow formulas,

$$
\begin{aligned}
p &= 1-\frac{\binom{n_G-x}{n_G/2-x}}{\binom{n_G}{n_G/2}} \qquad (20)\\
q &= \frac{\binom{n_G-x}{n_G/2-x}}{\binom{n_G}{n_G/2}} \qquad (21)\\
E(n_G/2,n_G) &= \sum_{x=0}^{n_G/2-1}\binom{n_G}{x}\frac{1-\left(\binom{n_G-x}{n_G/2-x}+1\right)p^{\binom{n_G}{n_G/2}}}{2^{n_G}q}\\
&\quad+\binom{n_G}{n_G/2}\left(1/2+\frac{\binom{n_G}{n_G/2}-1}{2^{n_G}}\right) \qquad (22)
\end{aligned}
$$

Because $x<n_G$, and $0<\left(\binom{n_G-x}{n_G/2-x}+1\right)p^{\binom{n_G}{n_G/2}}<1$, and $\binom{n_G}{n_G/2}\geqslant\frac{2^{n_G}}{\sqrt{2n_G}}$ (one of the Stirling's approximation results) we can bound $E(n_G/2,n_G)$ as follow,

$$
E(n_G/2,n_G) = O\left(2^{n_G}n_G^{-1}\right) \qquad (23)
$$

The *binary* Algorithm. For the *binary* algorithm we use different settings in order to compute possible lower bound of the average time complexity. Instead a single x-adversary X, we assume that every x-combination of providers can breach privacy of remaining records.

For $x \leqslant m$ all m-adversaries are able to break the privacy (due to upward pruning). Therefore, for each size of the coalition up to m providers only one privacy check will be performed. For $x>m$ the algorithm will stop its run after finding an x-adversary. In order to do so, it will run $\log_2(n_G-m-1)$ privacy checks. Thus, the expected number of checks follows formula,

$$
\begin{aligned}
E(m,n_G) &= \frac{m+1}{n_G-1}\\
&\quad+\frac{\log_2(n_G-m-1)}{n_G-1}\sum_{x=m+1}^{n_G-1}\binom{n_G}{x} \qquad (24)
\end{aligned}
$$

$$
\begin{aligned}
E(1,n_G) &= \frac{2}{n_G-1}+\frac{\log_2(n_G-2)}{n_G-1}\sum_{x=2}^{n_G-1}\binom{n_G}{x}\\
&= \frac{2}{n_G-1}+\frac{\log_2(n_G-2)}{n_G-1}(2^{n_G}-n_G-2)\\
&= O\left(2^{n_G}\cdot\frac{\log_2 n_G}{n_G}\right) \qquad (25)
\end{aligned}
$$

$$E(n_G/2, n_G) = \frac{n_G/2+1}{n_G-1} + \frac{\left(2^{n_G-1} - \binom{n_G}{n_G/2} - 1\right)\log_2\left(\frac{n_G}{2}-1\right)}{n_G-1} = O\left(2^{n_G} \cdot \frac{\log_2 n_G}{n_G}\right) \quad (26)$$

$$E(n_G-2, n_G) = 3 + \frac{2}{n_G-1} = O(1) \quad (27)$$

Non-EG Monotonic m-Privacy. The maximal number of privacy checks required to verify m-privacy w.r.t. non-EG monotonic constraint C for a group of anonymized records provided by P parties follows the **Equation 2**.

An m-adversary I can use any of its records in its attacks. Thus, to ensure m-privacy, all possible subsets of those records for each possible combination of attackers need to be considered. Number of all possible subsets of adversary records is exponential to total number of anonymized records, and equal to,

$$\sum_{i=0}^{m} \sum_{\substack{I \in P \\ |I|=i}} \prod_{R \in I} 2^{|records\_of(R)|} \quad (28)$$

However, some privacy checks are repeated and therefore redundant. If one of the malicious providers from an m-adversary does not use any of its records in an attack, then this provider could be treated as a non-attacker, which is equivalent to another scenario with an $(m-1)$-adversary that could be verified earlier. Thus, to avoid unnecessary privacy checks all scenarios with an adversary that does not participate any records in the attack, are skipped.

Privacy for adversarial coalitions are checked starting from the 0-adversary, then increasing number of attackers gradually, similar as in the *bottom-up* algorithm. Each scenario has a different coalition of adversaries that actively participate in attacks using different subsets of their records. Thus, each scenario is unique and required for verification.

By skipping redundant privacy checks for an i-adversary I we verify only scenarios, where a data provider R may use in the attack any but the empty subset of its records. For I, the number of possible sets of records used to attack is equal to $\prod_{R \in I}\left(2^{|records\_of(R)|}-1\right)$. Summing up over all possible coalitions of all possible sizes proves that the maximal number of privacy checks follows the **Equation 2**.

E. SECURE MULTIPARTY PROTOCOLS

SMC protocols implement distributed computations with security. A multiparty computation is secure if an adversary observes protocol execution, and learns nothing more than can be inferred from the output of the protocol. Formally, the view of each participant while running the protocol, can be simulated in a polynomial time given its input and output. In order to prove security of the protocol it is enough to show a simulator that simulates the view of computations given only input and output data.

There are two models of security in SMC, passive and active. In the latter model providers may not follow the protocol and act malicious, e.g., abort computations, send fake data. In the passive model (semi-honest) providers follow the protocol, but may use any intermediate results and communications observed during its execution to infer additional information. In both models participants may collude, i.e., share their information to increase their overall attack power. Details of the security definitions and underlying models can be found in [10].

Definition of the m-privacy notion assumes that there are no more than m out of n colluding providers ($1 \leqslant m \leqslant n-1$). We consider only the semi-honest model, where all parties follow a protocol.

E.1 Secure Comparison of Summation with a Number Protocol (SCoSAN)

The idea behind this protocol is that the result of two numbers comparison can be obtained by comparing their difference with zero. We assume that each party keeps a secret number x_i and k is publicly known. Thus, instead computing the sum of distributed numbers x_i and comparing it against k our protocol determines the sign of $(\sum_i x_i - k)(\sum_i c_i)$, where c_i are random and positive numbers.

The protocol works as follow (Algorithm 10). Each party from uniform distribution draws a random positive number c_i (line 1), and securely computes s by running a secure product of summations protocol (line 2) [34]. Since the sum $\sum_{i=1}^{n} c_i$ is greater than zero, multiplying it by the sum $\sum_{i=1}^{n}(x_i - k/n)$ preserves the sign of the latter sum, and hides the magnitude of the difference between $\sum_{i=1}^{n} x_i$ and k. Finally, the protocol returns $\mathrm{sgn}(s)$, which is equal to 1 if $s > 0$, 0 if $s = 0$, and -1 if $s < 0$ (line 3).

Algorithm 10: The Secure Comparison of Summation and a Number Protocol (SCoSAN) for $n > 2$; code run by a party P_i.

Data: A secret value x_i, and the publicly know k.
Result: 1 if $\sum_{i=1}^{n} x_i$ is greater than k, -1 if opposite, and 0 if they are equal.

1 c_i = `get_positive_random_number()`
2 Compute $s = \sum_{i=1}^{n}(x_i - k/n) \cdot \sum_{i=1}^{n} c_i$ (using secure product of summations protocol)
3 **return** $\mathrm{sgn}(s)$

Time complexity of the secure product of summations protocol is equal to $O(n)$ operations [34]. Thus, the time complexity for the SCoSAN protocol is also equal to $O(n)$ operations.

Security of the SCoSAN protocol. All parties generate positive numbers c_i, and run a secure product of summations protocol (SPoS) for pairs $(x_i - k/n, c_i)$. The SPoS protocol realizes full-privacy [34], i.e., all information revealed while running the protocol can be deduced from input and output data. The result of the SPoS protocol is a multiplication of two sums, where one of them is positive and randomly chosen. Thus, s does not reveal any information about $(\sum_{i=1}^{n} x_i - k)$ except its sign, which is enough to compare $\sum_{i=1}^{n} x_i$ and k. When parties collude, they are not able to breach privacy as well. Even when all but one parties cooperate, they are not able infer anything new about the remaing party, because they do not know its c_i.

THEOREM E.1. *The SCoSAN protocol is secure for the semi-honest model.*

PROOF. The value of s is the only intermediate result seen by all parties. s is a product of two sums, where $\sum_{i=1}^{n} c_i$ is a sum of positive random numbers. Thus, s disclose no information about $(\sum_{i=1}^{n} x_i - k)$ other than the sign of it, and s can be easily simulated by multiplying the output of the protocol by a random and positive number. Since SPoS is secure [34], then from the composition theorem [10] the SCoSAN protocol is secure. □

E.2 Secure *m*-Privacy Verification Protocols

In this section we describe level of security and efficiency achieved by secure verification protocols. We assume that all data providers are ordered based on fitness scores of their records, and all protocols are structured as described in the Section 3.5.

Part of the m-privacy definition is a privacy constraint C, which defines privacy of records. To keep the secure m-privacy verification protocol flexible for any constraint C, the protocol is implemented separately. Because of this design all verification protocols release results of privacy checks, which allow to determine privacy of some set of records, and monitor the verification algorithm.

Verification Procedure for an EG Monotonic C. The leading provider verifies if m-privacy w.r.t. C can be determined. If so, then it returns the status of m-privacy verification, and finishes the protocol. Otherwise it generates next coalition of potential attackers I. Each verification algorithm (*top-down*, *bottom-up*, *direct*) has different sequence of generated coalitions, but other than that they run the same operations. The I is broadcasted, and each data provider is able to identify if it is considered as an attacker or not. Then, the leader runs an SMC protocol to check if records from providers, which are not in I, are private. If privacy is breached, and the coalition contains no more than m providers, then due to the upward pruning the protocol returns negative answer, and is stopped. In other cases, the leading provider apply upward or downward pruning for I to limit number of further privacy checks.

An SMC protocol of the *binary* algorithm has the same code as for the TTP setting (Algorithm 3) with a few modifications. The code is implemented and run by the leading provider. All privacy checks are implemented as separate SMC protocols, and their results are announced. In addition, all coalitions of potential attackers that are used to verify privacy are broadcasted as well.

Security of the Protocol. Generated coalitions and results of privacy checks are revealed and not protected. Notice that coalitions are sets of identifiers, and each identifier is recognized only by its provider.

THEOREM E.2. *SMC protocols to verify m-privacy w.r.t. EG monotonic C are secure except results of potential attacks of generated and published coalitions.*

PROOF. To prove security of verification protocols for EG monotonic C we simulate all intermediate results of protocols by a simulator that knows the protocol output and broadcasted information.

Knowing previous I, the result of privacy verification, and the verification algorithm, the simulator is able to generate next coalition by finding its size, and finding first coalition that is not determined either by verification or pruning.

To simulate pruning, the simulator needs I, and the result of privacy verification for I treated as a coalition of attackers. Then simulator decides if pruning should be upward or downward, and what coalitions are pruned.

By keeping track of pruning the simulator can simulate checking if m-privacy is determined, i.e., verifying if all m-adversaries are decided.

The same conclusions apply for the *binary* protocol and super- and sub- coalitions generated during its run. Notice that super- and sub- coalitions are easily simulated when knowing previously generated coalitions.

Finally, since the privacy verification is a separate SMC protocol, which we assumed is secure, and from the composition theorem [10] m-privacy verification protocols are secure. □

Verification Procedure for a non-EG Monotonic C. The SMC protocol that implements an m-privacy verification algorithm for a non-EG monotonic constraint C bases on the *bottom-up* algorithm for EG monotonic C. Differences include broadcasting number of records provided by each party, and generating sets of records (attacking records) used in potential attacks. Although, sets of attacking records are not disclosed, results of their potential attack, i.e., privacy of remaining records, are revealed. Lack of monotonicity for C does not allow to apply any pruning techniques.

THEOREM E.3. *SMC protocols to verify m-privacy w.r.t. non-EG monotonic C are secure except results of potential attacks of generated and published coalitions that use subsets of their records.*

PROOF. We show that all intermediate results of the protocol execution can be simulated from the output of from the following disclosed information: IDs of providers in attacking coalitions, number of records each provider contributed, and results of privacy checks for different attacking records.

Knowing previously generated coalitions, a simulator can generate following coalition that has not been checked so far. Also, having information about number of records of each provider, and knowing their order allows the simulator to compute number of privacy checks that need to be performed.

The simulator can also simulate when each provider should generate next set of its local attacking records, in the following way. The leading provider gets a counter mapped to each provider. Counters are sorted in the same way as providers and zeroed. Every time a new set of attacking records needs to be generated, the leader increased the first counter by one. If its value is greater than number of records in the provider mapped to this counter, the provider zeroed this counter, sends a message to the provider to generate next set of attacking records, and increment one mapped to the next provider. If this counter also gets value greater than number of records for its provider, then the procedure is repeated, etc. Thus, the simulator can infer number of attacking records from each attacker as well as the index of current set of attacking records.

Finally, since the privacy verification is a separate SMC protocol, which we assume is secure, and from the composition theorem [10] the m-privacy verification protocol is secure. □

Time and Communication Complexities. Complexity of m-privacy verification protocols depend on complexity of a privacy verification protocol. Assuming that the privacy

verification protocol needs VER time to run, complexities of m-privacy verification protocols are equal to their respective complexities from their TTP algorithms (Appendix D) multiplied by VER.

Assume that VER_{COM} is a communication complexity for a privacy verification protocol and $\deg(VER_{COM})$ is the degree of polynomial that is equal or at least approximate (e.g., using the Taylor approximation of a function) the value $O(VER_{COM})$. All verification protocols shares all attacking coalitions I and results of privacy verifications. For an EG monotonic C, the number of possible attacking coalitions is equal to the number of privacy checks, but is different for each protocol: $\sum_{i=0}^{m}\binom{n_G}{i}$ for the *bottom-up* protocol, $\sum_{i=n-1}^{m}\binom{n_G}{i}$ for the *top-down* protocol, and $\binom{n_G}{m}$ for the *direct* protocol. Thus, the communication complexity is equal to $O\left(n_G^{\deg(VER_{COM})}2^{n_G-\deg(VER_{COM})}\right)$ for both the secure *top-down* and the secure *bottom-up* protocol, and $O((n_G+VER_{COM})2^{n_G-1})$ for the secure *direct* protocol.

For a non-EG monotonic C, time complexity of the secure protocol is the same as the complexity of their TTP algorithms (Appendix D) multiplied by VER. Communication complexity for this protocol is more complex to compute. It contains components from distribution of attacker coalitions I, requests for updates of attacking records, privacy verifications and their results, and broadcasting number of records. Overall the communication complexity is equal to $O((n_G+VER_{COM})2^{2n_G})$.

E.3 Secure m-Privacy Anonymization

Secure m-privacy anonymization protocol requires three different SMC protocols to be run: the secure median [1], the secure m-privacy verification (Section 3.5), and the secure fitness score. The last protocol needs to be defined for each privacy constraint C. The protocol defined for our example is described in the Section 4.2. We assume that all these protocols are secure, and results of their runs are disclosed.

All intermediate results of the protocol can be inferred from outputs of SMC protocols. Information that is disclosed at each anonymization step can be described as medians of all attributes s_i, fulfillment of m-privacy w.r.t. C for records split according to each median, and for m-private splits privacy fitness scores of verified subsets of records. Medians of all attributes need to be revealed to allow each provider to define its local possible subgroups of records. Verifying m-privacy for a set of records, and announcing its results allow each provider to accept or drop candidate splits. Finding the best splitting attribute is done based on fitness scores, which is different for each privacy constraint C. Allowing different privacy constraints C in the definition of m-privacy does not allow to embed the fitness score computation into the secure anonymization protocol. Thus, computing fitness score is implemented as a separate SMC protocol.

Knowing fitness scores allows to determine the best splitting attribute, definitions of subgroups of records that have been generated, and decisions if each subgroup should be further split. To prove formally that the m-privacy anonymization protocol is secure we present a simulator that using outputs of the protocol and subprotocols, simulates intermediate results.

THEOREM E.4. *The m-privacy anonymization protocol is secure except median values of each attribute, m-privacy fulfillments of subsets of records split by these medians, and for m-private subsets their fitness score values.*

PROOF. Each party locally splits its records based on the received median values s_i. Information about obtained subsets is used only by secure m-privacy verification and secure fitness score computations, and it is not disclosed in any other way. Disclosing fitness scores of obtained subsets of records allows a simulator to simulate choosing the splitting attribute, by selecting one, which maximizes the fitness score.

If all possible splits are not m-private, then the simulator decides to finish splitting the current set of records. 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 [10] the m-privacy anonymization protocol is secure as well. □

Time and Communication Complexities. Time complexity of m-privacy anonymization protocol depends on the complexity of secure median protocol MED, the complexity of m-privacy verification protocol VER, and complexity of secure fitness protocol FIT. Assuming the worst-case scenario (maximal number of splits) for $|T|$ records and q QID attributes, the time complexity is equal to $O(|T|(q+1)(MED+2\cdot VER+2\cdot FIT))$.

Communication complexity also depends on run protocols. MED_{COM}, VER_{COM}, and FIT_{COM} denote communication complexities for secure median, m-privacy verification, and fitness score protocols, respectively. Then, the communication complexity for the m-privacy anonymization protocol is equal to $O(|T|(q+1)(3+MED_{COM}+VER_{COM}+FIT_{COM}))$.