Distribution Agreement

In presenting this thesis or dissertation as a partial fulfillment of the requirements for an advanced degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis or dissertation in whole or in part in all forms of media, now or hereafter known, including display on the world wide web. I understand that I may select some access restrictions as part of the online submission of this thesis or dissertation. I retain all ownership rights to the copyright of the thesis or dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

Signature:

_____ _____

Slawomir A. Goryczka Date

Secure and Privacy-Preserving Distributed Data Release

By

Slawomir A. Goryczka
Doctor of Philosophy

Computer Science and Informatics

Li Xiong, Ph.D.
Advisor

Shun Yan Cheung, Ph.D.
Committee Member

Benjamin C. M. Fung, Ph.D.
Committee Member

Vaidy Sunderam, Ph.D.
Committee Member

Accepted:

Lisa A. Tedesco, Ph.D.
Dean of the Graduate School

Date

Secure and Privacy-Preserving Distributed Data Release

By

Slawomir A. Goryczka
M.S., Computer Science, Emory University, Atlanta, 2013
M.S./B.S., Mathematics, AGH University of Science and Technology, Kraków, 2007
M.S./B.S., Computer Science, AGH University of Science and Technology, Kraków, 2006

Advisor: Li Xiong, Ph.D.

An abstract of
A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2014

Abstract

Secure and Privacy-Preserving Distributed Data Release
By Slawomir A. Goryczka

The rapidly increasing prevalence of distributed data-driven applications highlights security and privacy issues in storing and processing sensitive data. Although manipulating raw data may violate privacy of their owners, techniques for processing and using privacy-preserving data descriptions can help. It remains a challenge, however, to ensure that adapted and new solutions are efficient, secure, and preserve privacy of data owners without disclosing confidentiality of data providers.

This dissertation proposes a new notion of m-privacy that addresses situations in which data providers may act as adversaries. To verify if such adversaries are capable of breaching privacy, we introduce novel strategies and an adaptive algorithm to select and use the most efficient approach. In addition, we design an algorithm to anonymize data to be m-private, i.e., any m colluding parties cannot compromise privacy. All verification and anonymization algorithms are implemented to be run in distributed environments by a trusted third party.

For settings without a trusted third party, we introduce new secure multiparty computation protocols that implement m-privacy verification and anonymization algorithms. For each protocol, we prove its security, analyze its communication complexity, and evaluate its overall performance for various settings.

This dissertation also describes a new two-phase algorithm to release differentially private histograms for records with customized privacy levels. We adapt a v-optimal partitioning algorithm to make it usable with differential privacy, and experimentally evaluate its performance.

Finally, for settings without a trusted third party, this dissertation presents a new distributed differential privacy mechanism that achieves collusion resistance with small overhead. We also define an enhanced fault tolerant and secure scheme for multiparty aggregation operations, and we employ it to implement our differential privacy mechanism in distributed environments. Both the privacy mechanism and the fault tolerant scheme are extensively analyzed and experimentally evaluated.

Secure and Privacy-Preserving Distributed Data Release

By

Slawomir A. Goryczka
M.S., Computer Science, Emory University, Atlanta, 2013
M.S./B.S., Mathematics, AGH University of Science and Technology, Kraków, 2007
M.S./B.S., Computer Science, AGH University of Science and Technology, Kraków, 2006

Advisor: Li Xiong, Ph.D.

A dissertation submitted to the Faculty of the
James T. Laney School of Graduate Studies of Emory University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Informatics
2014

Acknowledgments

First, I would like to thank my advisor, Prof. Li Xiong. Her insight and guidance were invaluable in exploring areas of research described in this dissertation. I would like to thank her for enormous hours we spent together discussing and addressing different privacy and security challenges. Also, I thank her for her patience, support, and all the words of encouragement that helped me to overcome doubts and tough moments of despair over last years. I would also like to thank my committee members, Prof. Vaidy Sunderam, Prof. Shun Yan Cheung and Prof. Benjamin C. M. Fung, for their valuable suggestions and comments. Discussions with them greatly helped me shape this dissertation. Additionally, I would like to thank Vaidy for his support, especially over last months. I would like to thank Prof. Ken Mandelberg, Prof. James Lu, Prof. Michelangelo Grigni, Prof. Eugene Agichtein, and Prof. Dominic Thomas. The knowledge and expertise they shared with me is a foundation of my studies and my teaching experience. I am grateful to all faculty, staff, and colleagues, who I met during my studies at Emory University. They build an exceptional and unique community of researchers that encourages everyone to ask questions, challenge propositions, verify ideas, and prove new theorems. I would like to thank to all my students for the honor of being their teacher. I also thank my colleague, Paweł Jurczyk, for inspiring me to return to academia, and for revealing new and exciting challenges in privacy and security. Finally, I owe my thanks to my family and all my friends, who helped and supported me every time I needed it over last years.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

The rapidly increasing prevalence of distributed data-driven applications has highlighted security and privacy issues in storing and processing sensitive data. Although manipulating raw data in distributed settings may violate privacy of data owners, we can still employ different techniques to prepare, maintain, and use privacy-preserving descriptions of data, e.g., anonymized databases, database statistics, data mining models. Such demand to perform more privacy-preserving computations in distributed settings has increased significantly. To address these emerging requirements, new privacy mechanisms need to be developed, and existing techniques need to be adapted to distributed environments.

1.1.1 Application Scenarios

Ensuring security and privacy in the process of preparing and using descriptions of distributed and sensitive data is a challenge, largely due to the absence of mutual trust among distributed, autonomous data owners and providers. All descriptions of data are subject to at least two privacy constraints: 1) privacy of data subjects or owners, and 2) confidentiality of data providers. For example, consider a system that integrates publicly available flight schedules with sensitive information about booked tickets. Data users, e.g., researchers that study patterns of disease spreading along communication routes, cannot process such data until a consent from every passenger is given. However, if such data

could be made available without divulging personal details, e.g., through obfuscation or aggregation, many benefits would accrue.

A few other scenarios in which release of sensitive data, modified to protect privacy would be useful, are described below.

Syndromic Surveillance. The terrorist attacks in 2001 and following years, and various disease outbreaks, such as the 2009 outbreak of H1N1 Flu [99], and the outbreak in Germany of Escherichia coli [101] have prompted much attention in syndromic surveillance systems [15, 89, 97]. In a simple scenario, a public health agency collects data from individual visitors that report their Influenza cases (*self surveillance*). The collected data, e.g., the daily number of Influenza cases, is monitored and analyzed to detect seasonal epidemic outbreaks. Both participation of a person and details of medical diagnosis are example of highly sensitive data, and privacy of their owners should be protected. Ensuring that data are collected and processed in a secure and privacy-aware manner is a challenge, but could be very valuable for users and researchers, if made available with high utility and high privacy guarantees.

Intelligence Data Collection. In numerous situations, intelligence gathering is performed in crowd settings both non-deliberately by the general public and by principals, who are anonymously embedded in the crowd. A canonical example is an uprising in a major city under hostile governmental control. The general public may use smart devices to report on various field data (*third party surveillance* [52]) as shown in Figure 1.1.

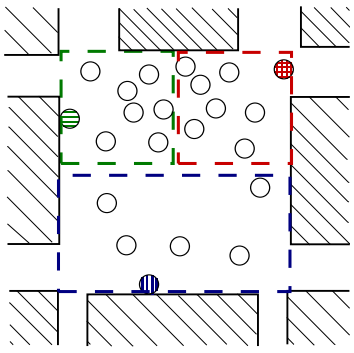


Figure 1.1: An example of a third party surveillance scenario: a town square with people (circles) divided into 3 zones monitored by data providers (colored and shaded circles).

In Figure 1.1, the shaded circles that represent data providers whose identity and location

should be hidden, and the open circles represent targets or data subjects about whom generic or abstract data should be reported, but personal data should not be divulged. There may also be agents among the crowd, reporting similar data using popular media (e.g. Twitter) to avoid identification. In either scenario, the number of participants reporting data may change over time, but that shall neither compromise their security, nor reveal their identity.

Collaborative Medical Data Aggregation. Consider a scenario in which a group of hospitals would like to collaborate in order to evaluate medical treatments of a rare illness. They cannot share their patient records among themselves without getting consent from all patients, but they can anonymize their records and use them instead. There may be also patients with records in multiple hospitals, and joining their data from different sources without their consent should not be possible. In addition, employees of one hospital may have a history of working for another hospital, hence they may have knowledge about some patients from their former place of work. That knowledge could be used by them, while analyzing anonymized data, which could lead to a privacy breach. To address this threat each hospital can anonymize their data records independently and then aggregates them with other hospitals. Alternatively, they can anonymize their data records in collaboration, i.e., hospitals would securely aggregate and then anonymize their data. In either case all privacy and security risks have to be carefully analyzed and addressed.

Generalized Settings. All above examples have similar goals and the same actors: data providers (hospitals, agents, patients) and data owners or subjects (patients, individuals). Notice that for some scenarios data owners are also data providers. In all examples data providers would like to anonymize data and/or compute some data statistics without exposing to third parties any sensitive data. Anonymized data are created from original data by applying suppression, generalization, or perturbation, such that no data recipient is able to learn anything about any data owner, except what can be derived from the final result [38]. Thus, we generalize above examples into one with the goal of preparing a data description (e.g., anonymized data, data statistics), which preserves both privacy of data owners, or subjects and confidentiality of data providers.

One approach to fulfill the goal is to find *a trusted third party* (TTP) that will collect data

from all providers and then perform necessary computations, or transformations to ensure that sensitive data is not disclosed (Figure 1.2). Employing a TTP to do computations addresses some, but not all challenges. For example, a few data providers can be attackers that collude in order to increase their chances of success. In addition, each data provider may have different privacy requirements, which need to be fulfilled. On top of that, a TTP may be unreliable or even unavailable, e.g., finding a TTP for a group of hospitals to share their data without consent from every patients may be impossible, unless such access is granted by law.

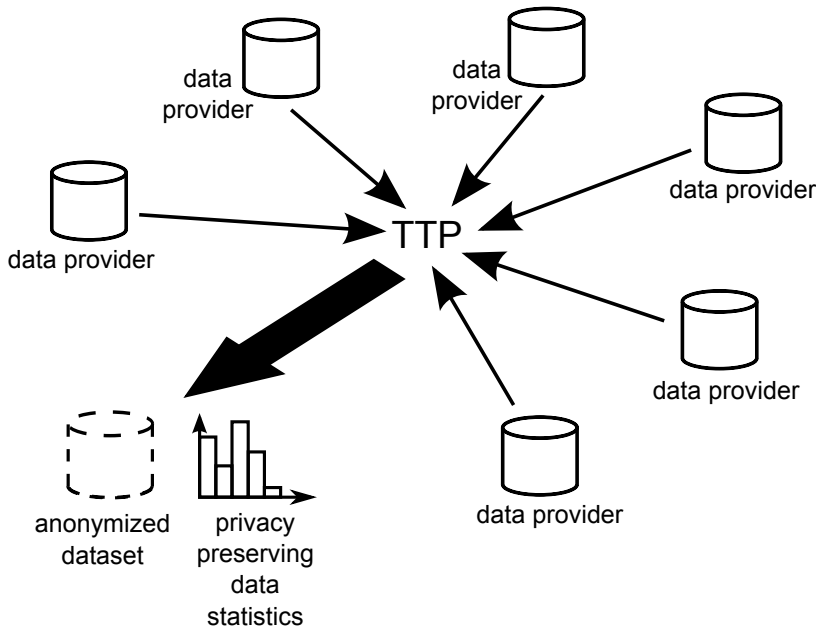


Figure 1.2: Collecting, preparing, and releasing privacy preserving data descriptions by a trusted third party (TTP).

When a TTP is not available all computations need to be executed by untrusted third parties or data providers without disclosing any sensitive data. Therefore, an important challenge is to protect the privacy of data owners and subjects, when a data aggregator is untrusted or not present. If the TTP is not present, providers may collaborate to perform a *secure multiparty computations* (*SMC*) protocol that returns the same outcome as it would be returned by the TTP. Each SMC protocol is described as a sequence of computations and exchanging messages among data providers.

Figure 1.3 depicts a general scenario with a group of data providers participating in

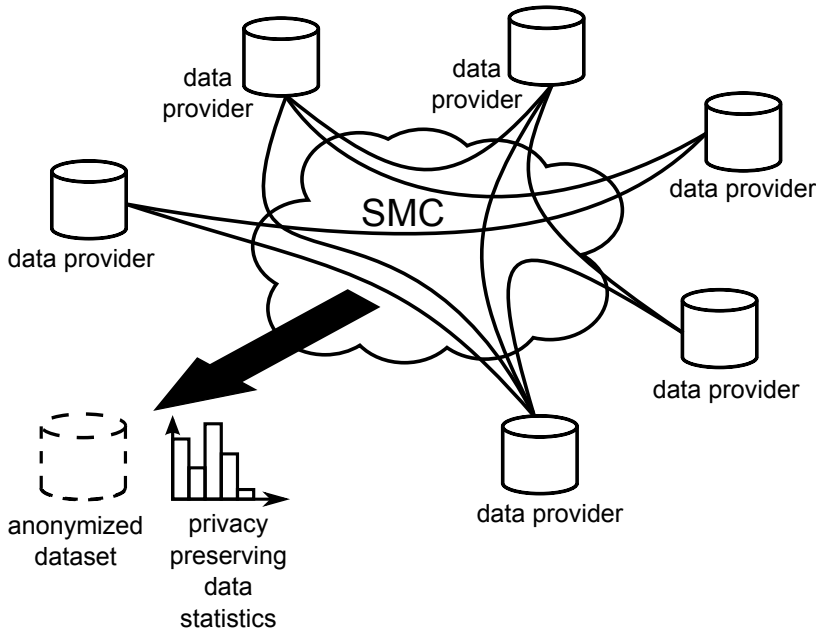


Figure 1.3: Preparing and releasing privacy preserving data descriptions by collaborating data providers running secure multiparty computation (SMC) protocols with insecure communication and computation resources.

an SMC protocol that returns anonymized dataset, or privacy preserving data statistics. In such settings, each data provider is connected to others using insecure communication channels. In addition, we assume that each provider is semi-honest (honest-but-curious) [39], i.e., it follows every step of an SMC protocol, but may use all intermediate results and its own data to breach privacy of remaining data providers and individuals. As a final result of our setting the SMC protocol returns a privacy-preserving description of data.

1.1.2 Challenges

Privacy Challenges. To model privacy threats for data providers and data owners we need to define a potential attacker and its background knowledge. A potential attacker can be external or internal. In either case predicting possible background knowledge of an attacker is a challenge. For example, some third parties may corrupt a small fraction of anonymized records, which may lead to a privacy breach [87]. In the most restrictive setting, we assume that all but one data record are corrupted. The fewer data records are breached, the more utility of original data can be preserved in its description. Finding a reasonable tradeoff between level of privacy restrictiveness and utility of data is a main privacy challenge for all

settings.

If a TTP is present, it needs to also evaluate risks of colluding data providers, i.e., providers that share their data, which may increase their chances in breaching privacy. Preserving privacy in a TTP scheme in the presence of colluding providers is a challenge that got very little attention so far.

Traditional *syntactic* approaches to data anonymization, such as removing identifying attributes, generalizing, or perturbing individual attribute values, preserve truthfulness of data, i.e., if needed an anonymized record can be linked with its original record. Many syntactic privacy notions utilize these approaches to counter various attacks, e.g., [57, 60, 85]. An important challenge for such notions in the distributed setting is to model the background knowledge of attackers. Notice that data providers may be among potential adversaries as well. In addition, if adversaries collude, they may share their data records, which increases their attacking power. To preserve privacy of remaining data owners attacked by a coalition of adversaries, new privacy mechanisms are needed.

Recently introduced *semantic* approaches protect participation of data owners in any computed statistics. They do not preserve truthfulness of data, but they also make no assumptions about background knowledge of attackers, i.e., they assume the worst-case scenario, in which such attacker knows all but one record. Semantic mechanisms employ different techniques to ensure data privacy, among which are perturbation and random sampling, e.g., differential privacy [28, 62]. A challenge for semantic privacy notions is to ensure that by removing a single record from a dataset all evidences of its presence are removed as well.

Applying semantic privacy notions to distributed settings with unreliable and colluding data providers introduces additional challenges. To address some of them, all data providers should participate in data anonymization in the same way, as we do not know which of them is reliable and not colluding. For example, each data provider needs to participate in perturbation process, in order to ensure that necessary level of perturbation is achieved even when a few participants have failed. Reliability of distributed systems is usually achieved by replicating necessary actions. Requiring participation of fewer providers to ensure enough perturbation, results in introducing additional and redundant noise. Finding a distributed

protocol that generates enough noise/perturbation with a small magnitude of redundant noise is an important challenge, which needs to be addressed to ensure scalability of the protocol. On top of that, different data providers (or data owners) may require different levels of privacy, e.g., in differential privacy level of required privacy is expressed by its parameter, which is often called a "privacy budget". For such settings new approaches are needed to efficiently release an integrated view of the data, while guaranteeing such customized level of privacy requirements.

Security Challenges. Both distributed settings, with and without a TTP, give rise to many data privacy and security challenges [52, 68, 77, 84]. If a TTP is present, then ensuring secure communication between each data provider and the TTP is the most important challenge.

For scenarios without a TTP all data providers need to agree on security schema and the protocol they will run. They may decide to choose a specific secret sharing scheme, encryption scheme, or perturbation scheme. Each of these schemes has its own challenges, but few of them are common for all schemes, e.g., colluding data providers, or unreliable communication channels. An additional group of issues is related to performance of protocols and security schemes, e.g., communication complexity of many protocols in Shamir's secret sharing scheme may be very high.

In addition, confidentiality of data providers should be also guaranteed, as any information about them and about data they provide may harm their security. Therefore, information about participating data providers is also sensitive and should be protected, which introduce additional challenges.

1.2 Contributions

Our contributions cover two dimensions of presented challenges: 1) different distributed data settings, i.e., with and without a TTP, 2) two groups of privacy notions: syntactic and semantic. In particular, we propose new solutions for anonymization of distributed datasets using syntactic notions. We introduce an m-privacy notion and corresponding algorithms to address the challenges when data providers may act as adversaries. We also introduce solutions to compute statistics of distributed data with respect to semantic

privacy notions. We propose new algorithms to address the challenge when different data providers or data owners have varied (customized) requirement of differential privacy. Our proposed algorithms are implemented for centralized scenarios, i.e., they can be run by a TTP (Table 1.1). In addition, we define new secure multiparty computation (SMC) protocols that implement these algorithms in a distributed environment, i.e., they can be run collaboratively by data providers and untrusted third parties without involving any TTP. All our algorithms and protocols are collusion resistance, i.e., they securely generate privacy-preserving data description in settings with a group of colluding data providers. The key contributions for proposed settings are presented in Table 1.1.

Table 1.1: Contributions.

Privacy notions	**TTP**	**SMC**
Syntactic	m-privacy: **[102]**, **[41]**	distributed m-privacy protocols: **[102]**, **[42]**
Semantic	customized differential privacy: [93]	distributed customized differential privacy protocols: **[43]**, [94], [75]

1.2.1 Syntactic Privacy Notions in Distributed Environments

We assume that data records are horizontally distributed among data providers and each record has an owner or a subject, whose identity should be protected. Each record attribute is either *an identifier*, which directly identifies the owner, or *a quasi-identifier* (QID), which may identify the owner if joined with a publicly known dataset, or *a sensitive attribute*, which values should not be possible to link with their data owners or subjects. A data recipient also has access to anonymized data as well as to some background knowledge, which represents any publicly available information about released data, e.g., Census datasets. For example, Table 1.2 presents data contributed by hospitals P_1, P_2, P_3, and P_4 that 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, and **Disease** is a sensitive attribute. Notice that one record, owned by Olga, is contributed by two providers P_2 and P_4, and is represented as a single record in anonymized dataset.

Syntactic privacy notions guarantee that an identifier can be linked to a sensitive value

Table 1.2: m-Adversary and m-privacy example.

T_1

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

T_2

Name	**Age**	**Zip**	**Disease**
Olga	32	98701	Cancer
Mark	37	12389	Flu
John	31	12399	Flu

T_3

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

T_4

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

T_a^*

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

T_b^*

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

only with limited probability. During anonymization identifiers are suppressed and QIDs are modified to achieve required privacy notion C, e.g., k-anonymity [80, 85], l-diversity [60], and t-closeness [57]. A table achieves C, if every of its QI group achieves C as well, where a QI group is a group of records with the same QID values. Attacks are run by *attackers*, i.e., a single or a group (*a coalition*) of external and internal entities that would like to breach privacy of data using their background knowledge, as well as anonymized data. Privacy

is breached if any attacker learns anything about data that cannot be derived from the background knowledge, anonymized dataset, and corrupted data records.

In our running example, T_a^* is one possible anonymization that guarantees k-anonymity and l-diversity ($k = 2$, $l = 2$), i.e., each group of anonymized records with the same QID (QI group) has at least 2 records in it, and there are at least 2 "well represented" sensitive values. Notice that the definition of l-diversity, which we use, defines "well represented" sensitive values as distinct values, i.e., l-diversity holds if each QI group contains records with at least l distinct sensitive values.

An attacker from the hospital P_1 may remove from T_a^* all records provided by P_1. In the first QI group there will be only one remaining record, which belongs to a patient between 20 and 30 years old. By using quasi-identifier attributes to join this record with the background knowledge BK (e.g. part of the Census database), P_1 can identify Sara as its owner (highlighted in the table) and her disease as Epilepsy. In practice, the attacker would use more attributes as quasi-identifiers and maximal BK to mount the linking attack [86]. In general, multiple providers may collude with each other, thereby having access to the union of their data, or a user may have access to multiple databases, e.g., a physician switching hospitals, and using information about her former patients.

To address this type of attack, we introduce a notion of m-privacy with respect to (w.r.t.) a privacy constraint C, which ensures that any coalition of m providers is not able to breach privacy of records provided by remaining parties [41]. For example, in Table 1.2 T_b^* is an anonymized table that satisfies m-privacy ($m = 1$) with respect to k-anonymity and l-diversity ($k = 2$, $l = 2$). We also prove that both problems of m-privacy verification and anonymization, in the general setting, are computationally hard. For m-privacy verification, we propose a few different strategies and an adaptive algorithm of selecting an efficient approach to be used. We also propose an algorithm to anonymize data, such that its result is m-private w.r.t. any C. All verification and anonymization algorithms have been implemented to be run in a distributed environment by a TTP.

SMC Protocols for m-Privacy. For settings without a TTP, we introduce a group of new secure multiparty computation protocols [102]. Our protocols implement m-privacy

verification algorithms as well as m-privacy anonymization algorithm. For each protocol, we prove its security and analyze its communication complexity. In addition, for all protocols we extensively test their performance in a distributed environment [42].

1.2.2 Semantic Privacy Notions (Differential Privacy) in Distributed Environments

For many scenarios, e.g., participatory sensing [11] and data surveillance [35], data subjects would like to hide their participation in computed statistics especially if it would bring negative consequences to them. In such settings, information about participation is sensitive and should be protected from linking with the data owner as well as the contributed data itself. In order to satisfy such a privacy requirement, we employ *differential privacy*, which is a semantic privacy notion that assures a strong and provable privacy guarantee for aggregated data regardless of background knowledge. To use differential privacy, independence of data subjects needs to be assumed, i.e., deleting one subject's data is equivalent of hiding all evidence of her participation in the dataset. Without such assumption hiding all evidence of participation would require modifying dependent records or removing them. Furthermore, we assume that no deterministic statistics about the participating data subjects have been previously released. If any additional information has been publicized earlier, it has to be taken into account to ensure that differential privacy for anonymized data is achieved. Under such assumptions, differential privacy guarantees negligible change of perturbed computation results, when a single data subject opts out of the data collection. A common way of achieving differential privacy is perturbation of results by carefully calibrated noise. Security of privacy-preserving statistics aggregation needs to be ensured either by a TTP or by an SMC protocol [38].

Data Statistics. The level of privacy preserved by differential privacy is defined by its parameter — *a privacy budget*. We allow each data provider to customize its privacy budget value and to set it to any value that will be accepted by data owners, and which is spent while answering queries. Such a setting can also be an outcome of a query workload that covers different subsets of records by each differentially private query. Therefore, every query has to

be issued with a privacy *cost* that will be subtracted from budgets of all records participating in this query execution. Notice that if each data owner is also a data provider, then the number of different values of privacy budgets can be enormous.

For such customized privacy budget settings, we propose a new two-phase approach. In the first phase, data records are deterministically grouped (partitioned) according to their required privacy level, and in the second phase, for each partition a differentially private histogram is generated. This approach maximizes utilization of the privacy budget by each record, therefore reduces the noise of the resulting histograms. Such customization of privacy budgets is very helpful in pricing privacy and compensating data owners for their loss of privacy. For example, a researcher pays individuals in order to use their data records to compute data statistics. In such settings, efficient managing of record budgets is crucial to prepare accurate statistics. Notice that records are distributed among semi-honest and mutually untrusted data providers. They are forbidden from sharing data records, but they can collaborate in order to perform any secure computations that will not breach privacy of their record owners.

Collusion Resistant SMC Protocols. Existing differential privacy mechanisms do not address all challenges of settings with distributed data. For example, dealing with colluding data providers receives very little attention, and to address it a naïve approach is often chosen, i.e., each provider ensures differential privacy of its data independently. Such solution adds a high magnitude of redundant noise, which may be avoided. We introduce a new differential privacy mechanism, which generates small amount of redundant noise that is necessary to ensure collusion resistance [43].

For settings without a TTP, we define an enhanced fault tolerant secure scheme (EFT), which is efficient, secure, and can be used to define a variety of secure multiparty aggregation operations. The new differential privacy mechanism has been implemented in a distributed environment using secure EFT scheme. Both, privacy mechanism and our new security scheme have been extensively analyzed and tested.

1.3 Organization

The remainder of this dissertation discusses all contributed privacy mechanisms as well as security schemas and protocols. First, Chapter 2 gives an overview of research in related areas.

Then, Chapter 3 introduces m-privacy w.r.t. a privacy constraint as a new collusion resistant privacy notion for settings with a TTP. Then, m-privacy is analyzed and tested for any privacy constraints, but especially for equivalence-group monotonic constraints, i.e., constraints that remain fulfilled for a group of anonymized records when a new record is added to that group. We also propose algorithms to verify and anonymize datasets with respect to m-privacy.

In Chapter 4, we present all m-privacy verification and anonymization algorithms implemented as secure multi-party protocols. Security of protocols is proved, and their performance is analyzed and tested extensively.

Next, Chapter 5 introduces a new method of preparing differentially private data histograms. Presented method has two phases, in the first one data are partitioned based on their privacy budget, and in the second one, based on data. Such approach allows to choose partitioning algorithms for each phase independently to adapt them to the distribution of privacy budgets as well as to the distribution of attribute values.

Chapter 6 provides new methods of ensuring semantic privacy in distributed environment with colluding data providers and limited reliability of resources. In this chapter, we analyze and test security schemes and privacy mechanisms, which are used to compute data statistics in such distributed environments. The new approach of ensuring differential privacy by distributed noise generation is very efficient in terms of performance and the magnitude of redundant noise.

Finally, Chapter 7 concludes and describes a few potential future directions of our research.

Chapter 2

Related Work

Both research areas, data privacy and security of computations experience tremendous growth and are very advanced in their development. However, exploration of their overlapping is still in its infancy.

2.1 Data Privacy

Roots of data privacy research are mainly in the database community. Increasing needs for protecting sensitive information has motivated many researchers to find new techniques of data manipulation without compromising privacy of their subjects.

2.1.1 Syntactical Privacy Notions

A large body of privacy preserving data analysis and publishing literature [33] assumes limited background knowledge of attackers and defines privacy using relaxed *adversarial* notion [60] by considering specific types of attacks. Representative principles include k-anonymity [80, 85], l-diversity [60], and t-closeness [57]. In [87], authors have modeled the instance level background knowledge as corruption, and studied perturbation techniques under these syntactic privacy notions.

In [48, 49, 67], authors studied distributed anonymization for vertically partitioned data with k-anonymity. Zhong *et al.* [98] studied classification of data collected from individual data owners (each record is contributed by different data owner), while maintaining k-

anonymity. Jurczyk *et al.* [51] proposed a notion called l'-site-diversity to ensure anonymity for data providers in addition to privacy of the data subjects.

Gal *et al.* [34] proposed a new way of anonymization of multiple sensitive attributes, which could be used to implement m-privacy w.r.t. l-diversity with providers as one of sensitive attributes. However, this approach uses the same privacy requirements for all sensitive attributes, while our notion of m-privacy has no such limitation.

Nergiz *et al.* [69] proposed a *look ahead* approach in anonymizing horizontally distributed data. In their approach providers disclose some information about data in order to decide, if collaborative anonymization will gain more information than individual one. We leave for the future research applying the *look ahead* approach to colluding scenarios considered with m-privacy.

In [10] authors designed an anonymization algorithm for frequent itemsets. Since the *support* function is monotonic, they took advantage of the *dual-pruning* to improve performance of their approach. The main difference with our approach (Chapter 3) is the goal of constraint verifications. To find frequent itemsets, all itemsets need to be decided either by checking or pruning. Thus, after simple modifications (e.g., not using *early stop*) our approach can find frequent itemsets, and the dual-pruning algorithm can verify newly presented notion of m-privacy. However, in either case, it will not be an efficient approach.

2.1.2 Differential Privacy

The notion of differential privacy was defined in [31]. Authors proposed a method of achieving it by perturbing results of computations with Laplace distributed noise. Several works studied the problem of distributed data aggregation with differential privacy. For example, Dwork *et al.* developed distributed algorithms of noise generation, in which random shares are drawn from either binomial or Poisson distributions [30].

McSherry implemented the Laplace mechanism in his framework PINQ and introduced a composition theorem [62]. The theorem describes how the privacy budget of each data record is spent, when data are used to answer multiple different queries with differential privacy costs.

In [90], Xiao *et al.* proposed several approaches that reduce relative errors of noisy

statistics, while still ensuring their differential privacy. Their algorithm obtains estimates of the query answers with large noise and iteratively refines its estimates to minimize relative errors.

In [91], Xiao and Tao presented the concept of personalized anonymity for attribute generalization. Their technique minimize generalizations and satisfy data subject privacy requirements with efficiency.

In [65], Mohammed *et al.* generated contingency tables (multidimensional histograms). Counts in such tables are perturbed in order to achieve differential privacy. Their non-interactive approach is flexible and can be adapt to use different criteria in building contingency table records. However, authors did not consider configurations of attribute values that are not represented in the dataset, i.e., their original count is equal to zero. They probabilistically generalize records and introduce noise to preserve differential privacy of data owners. Values of each attribute are generalized to the same level of their taxonomy trees for all records.

Alhadidi *et al.* presented a secure two-party differentially-private data release algorithm for horizontally partitioned data [4]. Differential privacy of released data is achieved by an exponential mechanism [63]. The mechanism is used in domains where noise perturbation is not possible.

2.2 Security of Computations

Security of computations has been a challenge since the first computer network was built and a distributed computer system established. Empowered by the cryptography community and other communities, ensuring security of computations becomes a very active research topic.

2.2.1 Secure Multiparty Computations

Secure two-party computations have been defined by Yao, who presented a solution to the Millionaires' Problem, where two millionaires want to find out, who is richer without disclosing their actual wealth [96]. Yao's protocol has been generalized to secure multiparty

computations (SMC) in [40] and to scenarios with active adversaries in [58]. Protocols implemented in a general SMC scheme are computationally expensive. However, many specialized and efficient protocols have been developed for variety of specific operations. Efficient SMC protocols have been also introduced for aggregation operations, e.g., a secure sum, a secure union, and a secure scalar product [20, 59].

2.2.2 Secret Sharing Schemes

Shamir introduced the first distributed secret sharing scheme [81]. A secret is decomposed into n shares, by randomly generating a polynomial of order s, such that its value in zero is equal to the secret. A set of at least s distinct points and values of the polynomial in such points are securely distributed. Any group of s shares are enough to interpolate the polynomial and to reveal the secret. Notice that having there are no limits on number of generated shares. The Shamir's secret sharing scheme, with secure communication channels, is information-theoretically secure [6].

Note that Shamir scheme was introduced only for integer numbers. Catrina *et al.* adapts his scheme to use floaring point numbers, by implementing different arithmetic operation protocols [18].

Brun and Medvidovic introduced a distributed secret sharing scheme using a concept of "tiles" [9]. In their scheme data are represented by tiles, which are distributed in an untrusted network. Each tile discloses a single bit of data, but the scheme is generally secure if at least half of them are not corrupted. Note that some statistical information is always revealed with revealing each tile.

2.2.3 Encryption Schemes

Paillier [71] presented an additively homomorphic cryptosystem that computes the encrypted sum using for this only encrypted numbers. Its threshold variant has been introduced by Damgård *et al.* [24] and used to implement an electronic voting system. Hazay *et al.* presented a threshold Paillier scheme for two-party settings [46]. Cramer *et al.* applied homomorphic encryption cryptosystems to many secure protocols [23].

Pedersen *et al.* compared encryption-based and secret sharing schemes used in privacy preserving data mining [72]. They presented both schemes, but their comparison does not include performance evaluations and does not cover the latest security schemes and privacy mechanisms.

Chu *et al.* proposed a threshold security scheme to collect data in a wireless sensor network [19]. In their scheme, encryption keys are symmetric and valid only for a requested period of time. The scheme is not homomorphic, i.e., a data recipient needs to decrypt all ciphertexts before aggregating them, therefore she has to be trusted.

2.3 Secure Multiparty Computations with Differential Privacy

Combining benefits of both data privacy and security of computations is a relatively new direction of research. A third party that uses large scale systems and manages sensitive data has to preserve privacy of data subjects and ensure that any computation will leak nothing, i.e., all distributed and local computations will be secure.

Ács *et al.* applied privacy-preserving and secure data aggregation to the smart metering system [1]. Their scheme was inspired by previous work on secure sensor data aggregation in wireless networks [16, 17]. The scheme uses differential privacy model and homomorphic properties of a modulo addition-based encryption scheme to ensure privacy of results and security of computations.

In a similar scenario individual users collect and aggregate time-series data. PASTE is the system that implements that and ensures differential privacy of results [76]. Differential privacy is achieved by perturbing the most significant coefficients of the discrete Fourier transform of the query answers by a Distributed Laplace Perturbation Algorithm (DLPA). Each participant generates partial noise that is a vector of four Gaussian random variables. Security of computations is ensured by the threshold Paillier cryptosystem.

Shi *et al.* also utilized a homomorphic encryption scheme and minimized its communication complexity by generating an encryption key from the current round number and the existing key [83]. Their privacy mechanism is distributed and ensures approximate

differential privacy with noise drawn from the discrete two-sided geometric distribution [37].

Mohammed *et al.* presented a secure two-party differentially-private data release algorithm for vertically partitioned data in the semihonest adversary model [64]. Differential privacy of released data is achieved by an exponential mechanism [63].

2.4 Secure Multiparty Data Statistics with Differential Privacy

Jagadish *et al.* introduced a dynamic programming algorithm to create a single dimensional v-optimal histogram for a given number of buckets [47].

Barak *et al.* introduced algorithms, in which they compute frequency matrix, and apply Fourier transform to it, and add Laplace noise in this domain [5]. To eliminate negative coefficients, authors employed linear programming to find values, which minimizes additional perturbation introduced to data.

Hay *et al.* improved accuracy of differentially private histograms by using a constrained inference postprocessing tuning [45].

Xu *et al.* employed partitioning to lower the noise magnitude of differentially private histogram [95].

Xiao *et al.* used the wavelet transformation to introduce a Privelet method [92]. After transforming data they added polylogarithmic noise to achieve differential privacy.

Cormode *et al.* introduced a new heuristic for differentially private spatial decompositions (PSD), which can be employed to build a multidimensional histogram with privacy guarantees [21, 22]. Both *DPCube* introduced by Xiao *et al.* in [93] and PSD belong to the family of Binary Space Partitioning (BSP) techniques.

In [44], Haddadi *et al.* discussed challenges and some potential avenues for addressing issues in privacy analytics. Authors proposed a framework that distributes execution of verified queries within a community of data providers. Although authors describe results aggregation and a few approaches to dilute them in order to achieve differential privacy, they do not consider distributed noise generation as a possible approach.

Chapter 3

Distributed Data Aggregation with m-Privacy

3.1 Introduction

Two settings are commonly used to anonymize and publish horizontally distributed data. In the first setting (*anonymize-and-aggregate*) data providers anonymize their data independently and then aggregate them (Figure 3.1a). A more desirable approach (*aggregate-and-anonymize*) that saves more data utility is *collaborative data publishing* [33, 48, 49, 66], in which providers securely aggregate their data and then anonymize them (Figure 3.1b), using either a trusted third-party (TTP) or Secure Multi-party Computation (SMC) protocols [39, 59].

For either scenario an external attacker P_0 has access only to anonymized data T^* and some background knowledge (BK), which represents any publicly available information about anonymized data. Both T^* and BK are not enough to breach privacy of any data subject. However, an attacker P_1 that is working for a data provider has an advantage of knowing T_1 as well as T^* and BK. Such additional knowledge is useless when each data provider anonymized its dataset independently, but may allow a privacy breach for the scenario, when data are first aggregated and then anonymized. Preserving privacy of data records in such environment is more challenging when providers and external attackers

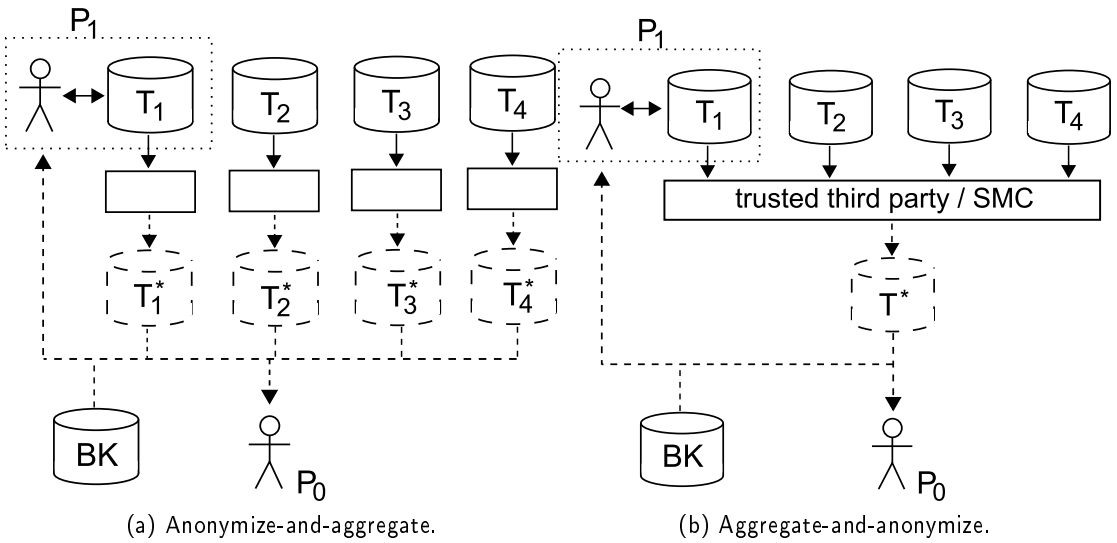


Figure 3.1: Two types of distributed data publishing settings for four providers.

collaborate.

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

Many privacy notion C are designed to protect against certain type of attacks. Therefore, achieving C by an anonymized dataset guarantees that data owners of its records are protected against this type of attacks.

Definition 3.1 (Dataset Achieving a Privacy Constraint)**.** *For syntactical privacy notions, an anonymized dataset T^* achieves C if and only if every group of records with the same quasi-identifier attribute values (quasi-identifier equivalence group, QI group) achieves C as well.*

Existing Solutions. Collaborative data publishing can be considered as a multi-party computation problem, in which multiple providers wish to compute an anonymized view of their data without disclosing any private and sensitive information. We assume the data providers are semi-honest [39, 59], which is commonly assumed in distributed computations. A trusted third party (TTP) or Secure Multi-Party Computation (SMC) protocols [49] can

be used to guarantee lack of *intermediate* information disclosure *during* the anonymization. However, neither TTP nor SMC protects against inferring information from the anonymized data.

The problem of inferring information from anonymized data has been widely studied in a single data provider settings [33]. A data recipient that is an attacker, e.g., P_0, attempts to infer additional information about data records using the published data, T^*, and background knowledge, BK. For example, k-anonymity [80, 85] protects against identity disclosure attacks by requiring each QI group (Definition 3.1) to contain at least k records, i.e., each group contains at least k records, which quasi-identifiers have the same values. l-Diversity requires each QI group to contain at least l "well-represented" sensitive values [60], and t-closeness, which requires the distribution of a sensitive attribute in each QI group to be close to the global distribution [57]. Differential privacy [27, 29] guarantees that the presence of a record cannot be inferred from a statistical data release, while assuming very little about background knowledge of attackers.

New Challenges. Collaborative data publishing introduces a new attack that has not been studied so far. Each data provider, such as P_1 in Figure 3.1, 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, each provider has additional data knowledge of its own records, which can help with the attack. This issue can be further worsened when multiple data providers collude with each other.

In the social network or recommendation setting, a user may attempt to infer private information about other users using the anonymized data or recommendations assisted by some background knowledge and her own account information. Malicious users may collude or even create artificial accounts as in a shilling attack [12].

Contributions. We define and address a new type of "insider attack" by an *m-adversary*, i.e., a coalition of m colluding data providers or data owners that attempts to infer data records contributed by others. Notice that a 0-adversary models the external data recipient, who has access only to the external background knowledge. Since each provider holds a subset of the overall data, this inherent data knowledge has to be explicitly modeled, and

considered when the data are anonymized.

We address the new threat introduced by m-adversaries, and make several important contributions. First, we introduce the notion of m-privacy that explicitly models the inherent data knowledge of an m-adversary, and protects anonymized data against such adversaries with respect to a given privacy constraint. For example, in Table 1.2 T_b^* is an anonymized table that satisfies m-privacy ($m = 1$) with respect to k-anonymity and l-diversity ($k = 2$, $l = 2$).

Second, for scenarios with a TTP, to address the challenges of checking a combinatorial number of potential m-adversaries, we present heuristic algorithms for efficient m-privacy verification given a set of records. Our approach utilizes effective pruning strategies exploiting the equivalence group monotonicity property of privacy constraints, and adaptive ordering techniques based on a novel notion of privacy fitness. We also present a data *provider-aware* anonymization algorithm with adaptive strategies of checking m-privacy fulfillment, to ensure high utility and m-privacy of sanitized data with efficiency.

3.2 m-Privacy Definition

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

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

If the records contain multiple sensitive attributes, then we could treat each of them as the sole sensitive attribute, while others would be included to the quasi-identifier [60]. However, in our scenarios we use an approach, which preserves more utility without sacrificing privacy [34].

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.

3.2.1 m-Privacy

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

In our example (Table 1.2), the privacy constraint C is defined as $C = C_1 \wedge C_2$, where C_1 is k-anonymity with $k = 2$, [85], and C_2 is l-diversity with $l = 2$, [60]. 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, which, whether achieved, is enough 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 3.2 (m-Privacy with respect to C)**.** *Given n data providers, a set of records T, and an anonymization mechanism $\mathcal{A}$, an m-adversary I ($m \leqslant n-1$) is a coalition of m providers, which jointly contribute a set of records T_I. $\mathcal{A}(T)$ satisfies m-privacy with respect to a privacy constraint C if and only if, any anonymized superset of records $\mathcal{A}(T')$ from non-m-adversary providers satisfies C, i.e.,*

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

Corollary 3.3. *For all $m \leqslant n-1$, if $\mathcal{A}(T)$ is m-private, then it is also $(m-1)$-private. If $\mathcal{A}(T)$ is not m-private, then it is also not $(m+1)$-private.*

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

***m*-Privacy with Duplicate Records.** m-Privacy can be also guaranteed when there are duplicate records (such as records from a patient transferred between hospitals). In our initial example Olga has records in two hospitals P_2 and P_4 (Table 1.2). For such cases, the duplicates are treated as a single record shared by a few providers. If any of the providers is a member of an m-adversary, the record will be considered as a part of its background knowledge.

***m*-Privacy and Syntactic Privacy Constraints.** Let C be a syntactic privacy constraint, i.e., a constraint that preserves data truthfulness at the record level, e.g., k-anonymity, l-diversity, t-closeness [57]. T^* satisfying C will only guarantee 0-privacy w.r.t. C, i.e., C is not guaranteed to hold for every QI group after excluding records belonging to any data provider (Definition 3.1). Thus, each data provider may be able to breach privacy of records provided by others. These guarantees are enough when there is a single data provider, but when there are many data providers they are insufficient. In our example from Table 1.2, T_a^* satisfies only 0-privacy w.r.t. C, while T_b^* satisfies 1-privacy w.r.t. the same C. Thus, T_b^* preserves privacy (defined by C) from being breached by malicious data users and any single data provider.

m-Privacy is defined w.r.t. a privacy constraint C, and hence it will inherit all strengths and weaknesses of C. For example, if C is defined as k-anonymity, then ensuring m-privacy w.r.t. C will not protect against homogeneity attacks [60] and deFinetti attack [53]. m-Privacy w.r.t. C protects against privacy attacks issued by any m-adversary if and only if, C protects against the same attacks by an external data recipient. m-Privacy notion is orthogonal to the privacy constraint C being used, and enhances privacy it defines to distributed settings, where up to m data providers collude.

3.2.2 Monotonicity of Privacy Constraints

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

Generalization based monotonicity of privacy constraints has been already defined in the literature (Definition 3.4) [57, 60]. Its fulfillment is crucial for designing efficient generalization algorithms [56, 57, 60, 85]. We will refer to it as *generalization monotonicity*.

Definition 3.4 (Generalization Monotonicity of a Privacy Constraint [57, 60]). *A privacy constraint C is generalization monotonic if and only if, for any two equivalence groups $\mathcal{A}_1(T)$ and $\mathcal{A}_1(T')$ that satisfy C, their union satisfies C as well,*

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

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

Definition 3.5 (Equivalence Group Monotonicity of a Privacy Constraint, EG Monotonicity). *A privacy constraint C is EG monotonic if and only if, for a group of records T such that its equivalence group $\mathcal{A}_1(T)$ satisfies C, and any group of records $\widetilde{T}$, their anonymized union satisfies C,*

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

Notice that $\widetilde{T}$ can be any set of records, which makes EG monotonicity more general than generalization monotonicity. If a constraint is EG monotonic, it is also generalization monotonic, but vice versa does not always hold. k-Anonymity and l-diversity, which requires l distinct values of sensitive attribute in a QI group, are examples of EG and generalization monotonic constraints. Entropy l-diversity [60] and t-closeness [57] are examples of generalization monotonic, but not EG monotonic constraints at the same time. For example, consider a subset of two anonymized records with 2 different sensitive values satisfying entropy l-diversity ($l = 2$), i.e., the 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 change the entropy of sensitive values significantly. However, it is generalization monotonic because it will still hold if two QI groups satisfying entropy l-diversity ($l = 2$) are (generalized) into a new group.

Corollary 3.6. *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 corollary 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 [60], C is EG monotonic as well.

Theorem 3.7. *m-Privacy with respect to any constraint C is EG monotonic if and only if, C is EG monotonic.*

This theorem holds also when applied for generalization monotonicity. Proofs of this theorem for both EG and generalization monotonicities defined with respect to records and not m are as follows.

Proof of Theorem 3.7 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 3.5), 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 (Corollary 3.3), 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 $C(\mathcal{A}(T \backslash T_I)) = true$. 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 of Theorem 3.7 for generalization monotonicity. Assume that all records contributed by n providers $P = \{P_1, \ldots, P_n\}$ are split into two sets 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 3.4), $\mathcal{A}(T_1 \cup T_2)$ 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 (Corollary 3.3), 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. □

Corollary 3.8. *If a constraint C is EG monotonic, then the definition of m-privacy w.r.t. C (Definition 3.2) may be simplified such that only $T' = T \setminus T_I$ are checked, i.e.,*

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

Indeed, if $\mathcal{A}(T \setminus T_I)$ satisfies C, then EG monotonicity of C guarantees that any anonymized superset of $T \setminus T_I$ satisfies C as well. Thus, $\mathcal{A}(T)$ fulfills definition of m-privacy w.r.t. C. In addition, if a coalition I is unable to breach privacy, then any its sub-coalition with fewer records cannot do so either (Definition 3.5). Unfortunately, generalization monotonicity of C is not enough to guarantee this property. [1].

3.3 m-Privacy Verification

Checking whether a set of anonymized records satisfies m-privacy w.r.t. a privacy constraint C creates a potential computational challenge due to the combinatorial number of m-

[1] Generalization monotonicity of C does not guarantee fulfillment of C for a QI group of two (or more) sets of records, where at least one of them has a QI group that does not fulfill C.

adversaries and variety of privacy definitions of C. In this section, we first analyze the problem by modeling the adversary space. Then, we present heuristic algorithms with effective pruning strategies and adaptive ordering techniques for efficiently checking m-privacy w.r.t. an EG monotonic constraint C. Finally, we present implementation of introduced algorithms that can be run by a trusted third party (TTP) to verify m-privacy w.r.t. EG monotonic and non-EG monotonic privacy constraints.

3.3.1 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 graph shown in Figure 3.2. 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.

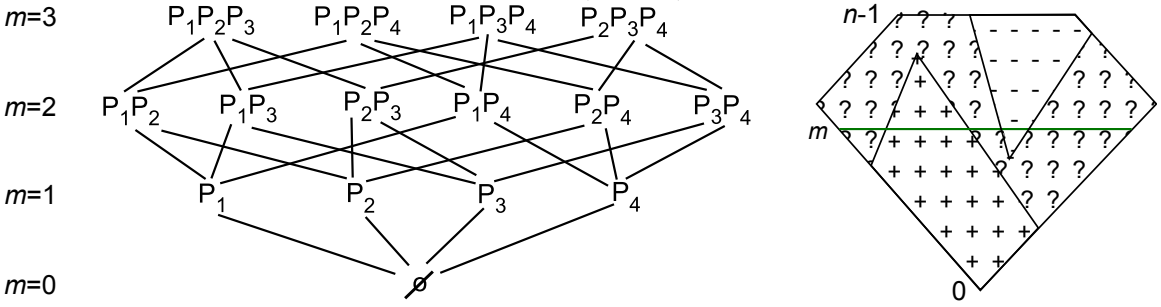


Figure 3.2: The domain of coalitions for data providers $\{P_1, P_2, P_3, P_4\}$ and its simplified representation for n providers with two types of pruning. Plus signs represent coalitions that cannot breach privacy, while minus signs coalitions that can breach privacy of given anonymized data records.

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 (Corollary 3.8), i.e., adversaries on the horizontal line. For example, in Figure 3.2, given m, all coalitions that need to be checked are below and at the green

horizontal line. For each possible coalition we need to check scenarios where any possible subset of their records is excluded from anonymized data. If C is EG monotonic, then it is sufficient to check only coalitions at the green horizontal line.

Given an EG monotonic constraint, the straight forward way of verifying m-privacy is to sequentially generate all possible $\binom{n_G}{m}$ m-adversaries and then check privacy of the corresponding remaining records (a *direct* algorithm). 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 and we propose a few heuristics that utilize two strategies and limit the number of privacy checks, i.e., pruning and adaptive ordering.

3.3.2 Heuristic Algorithms for EG Monotonic Constraints

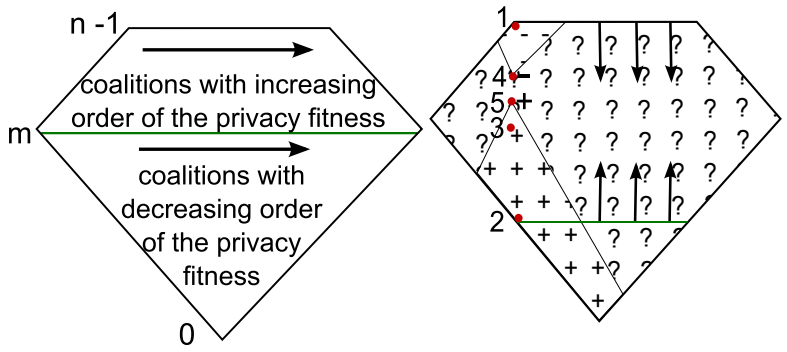
In this section, we present heuristic algorithms for efficiently checking m-privacy w.r.t. an EG monotonic constraint. Then, we modify them to check m-privacy w.r.t. a non-EG monotonic constraint.

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

Pruning Strategies. The pruning is possible thanks to the EG monotonicity of m-privacy (Corollaries 3.3, and 3.8). If a coalition is not able to breach privacy, then all its sub-coalitions will not be able to do so as well, and hence do not need to be checked (downward pruning). On the other hand, if a coalition is able to breach privacy, then all its super-coalitions will be able to do so as well, and hence do not need to be checked (upward pruning). In fact, if a sub-coalition of an m-adversary is able to breach privacy, then the upward pruning allows the algorithm to terminate immediately as the m-adversary will be able to breach privacy (*early stop*). Figure 3.2 illustrates the two pruning strategies where $+$ represents a case when a coalition does not breach privacy and $-$ otherwise.

Adaptive Ordering of Adversaries. In order to facilitate the above pruning in both directions, we adaptively order the coalitions based on their attack powers (Figure 3.3(a)).

This is motivated by following observations. For downward pruning, super-coalitions of m-adversaries with limited attack powers are preferred to be checked first as they are less likely to breach privacy, and hence increase the chance of downward pruning. In contrast, sub-coalitions of m-adversaries with significant attack powers are preferred to be checked first as they are more likely to breach privacy, and hence increase the chance of the early stop.



(a) Adaptive ordering.

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

Figure 3.3: Adaptive ordering for efficient pruning and an example run of the *binary* m-privacy verification algorithm.

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

Definition 3.9 (Privacy Fitness Score)**.** *Privacy fitness F_C for a set of anonymized records T^* is a level of fulfillment of the privacy constraint C. A privacy fitness score is a function f of privacy fitness with values greater or equal to 1 only if $C(T^*) = true$,*

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

Notice that privacy fitness score can be defined by any function that follows the above definition. The definition depends on privacy constraint C, therefore it cannot be exactly formulate without knowing the C. In our setting, C is defined as (k-anonymity $\wedge$ l-diversi-

ty). The privacy fitness score is 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:

$$score_{F_C}(T^*) = (1-\alpha) \cdot \frac{|T^*|}{k} + \alpha \cdot \frac{|\{t[A_S] : t \in T^*\}|}{l} \tag{3.1}$$

The privacy fitness score quantifies the attack power of attackers. The higher their privacy fitness scores are, the more likely they are able to breach the privacy of the remaining records. In order to maximize the benefit of both pruning strategies, the super-coalitions of m-adversaries are generated in the order of ascending fitness scores (ascending attack powers), and the sub-coalitions of m-adversaries are generated in the order of descending fitness scores (descending attack powers) (Figure 3.3(a)).

Now we present several heuristic algorithms that use different search strategies, and hence utilize different pruning directions. All of them use the adaptive ordering of adversaries to enable fast pruning. Notice that for a record contributed by many providers (duplicated records), if any of them is an attacker then the record is considered as it would have been provided only by the attacker.

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.

The Bottom-Up Algorithm. The *bottom-up* algorithm is similar to the *top-down* algorithm. The main difference is in the sequence of coalition checks, which is in a bottom up fashion starting from 0-adversary, and moving up (line 3). The algorithm stops whether a privacy violation by any adversary is detected (early stop) or all m-adversaries are checked (lines 6 to 9).

The Binary Algorithm. The *binary* algorithm (Algorithm 2), inspired by the binary search algorithm, checks coalitions between $(n_G - 1)$-adversaries and m-adversaries, and takes advantage of both upward and downward prunings (Figure 3.3b). The goal of each iteration is to search for a pair of coalitions I_{sub} and I_{super}, such that I_{sub} is a direct sub-

Algorithm 1: The *top-down* m-privacy 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.
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
```

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 works as follows. First, it starts with $(n_G - 1)$-adversaries, finds the first coalition of attackers that violates privacy, and assigns it to I_{super} (lines from 4 to 7). Then, it finds an I_{sub}, i.e., a sub-coalition of I_{super}, which does not breach privacy (line 8). At each step, a new coalition $I : I_{sub} \subset I \subset I_{super}$ (such that $|I| = \frac{|I_{super}|+|I_{sub}|}{2}$; line 12) is checked (line 13). If I can breach privacy, then I_{super} is updated to I (line 14). Otherwise, I_{sub} is updated to I (line 16). The algorithm continues until the direct parent-child pairs I_{super} and I_{sub} are found (line 11). Then pruning in both directions is performed (lines 17 and 18), and the algorithm starts the next iteration. The algorithm stops when m-privacy can be determined (line 3).

Adaptive Selection of Algorithms. Each of the above algorithms focuses on different search strategy, and hence utilizes different pruning. Which algorithm to use is largely dependent on the characteristics of a given group of providers. Intuitively, the privacy fitness score (Equation 3.1), which quantifies also 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. Defining the

Algorithm 2: The *binary* m-privacy verification algorithm.

Data: Anonymized records T^* from providers P, an EG monotonic C, a fitness scoring function $score_F$, and the m.

Result: *true* if T^* is m-private w.r.t. C, *false* otherwise.

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

exact strategy of choosing the m-privacy verification approach can be done based on the background knowledge (data statistics) or during computations, i.e., when characteristics of data contributed by each provider are computed. We utilize such an adaptive strategy in the anonymization algorithm (discussed in Section 3.4), and experimentally evaluate it.

3.3.3 m-Privacy Verification Algorithm for Non-EG Monotonic Constraints

If a privacy constraint C applied to the definition of m-privacy (Definition 3.2) 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 is expanded to cover all possible sets of attacking records. However, the adaptive ordering of providers is still very useful in finding an m-adversary that breaches privacy. Coalitions of m-adversaries with significant attack powers are preferred to be generated first as they are more likely to breach privacy, and hence increase the chance of the early-stop.

A general verification algorithm for m-privacy w.r.t. any C (Algorithm 3) 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 participates to this set with at least one record (line 4). Finally, it verifies if such subsets can be used in attacks to breach 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 attacker, which 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, therefore it is skipped.

Algorithm 3: The verification algorithm of m-privacy w.r.t. any C.

Data: Anonymized records T^* from providers P, and the m.
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
```

3.3.4 The Worst-Case Time Complexity

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

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

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 parties P, the maximal number

of privacy checks follows the formula,

$$\sum_{i=0}^{m} \sum_{\substack{I \subset P \\ |I|=i}} \prod_{P_j \in I} \left(2^{|T_j|} - 1\right) = O(2^{|T|}) \tag{3.2}$$

where T_j is a set of records provided by P_j, and T is a set of all records from all providers.

3.3.5 The Average Time Complexity

Computing the average time complexity for majority algorithms is very difficult. Finding the exact average time complexity depends on many factors, therefore 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 $H_M(x)$,

$$E(m, n_G) = \frac{\sum_{x=0}^{m-1} \binom{n_G}{x} H_m(x) + \sum_{x=m}^{n_G-1} \binom{n_G}{x} H_M(x)}{2^{n_G}} \tag{3.3}$$

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

$$\begin{aligned} H_m(x) &= \sum_{i=1}^{n_G-x-2} \binom{n_G - x}{i} + n_G - 1 \\ &= \sum_{i=0}^{n_G-x} \binom{n_G - x}{i} + x - 3 \\ &= 2^{n_G-x} + x - 3 \end{aligned} \tag{3.4}$$

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.

$$H_M(x) = 2^{n_G-x} + x - 2 - \sum_{i=1}^{m-x} \binom{n_G - x}{i} \tag{3.5}$$

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

$$\begin{aligned} E(m, n_G) &= \left[\sum_{x=0}^{m-1} \binom{n_G}{x} H_m(x) \right. \\ &\quad \left. + \sum_{x=m}^{n_G-1} \binom{n_G}{x} H_M(x) \right] / 2^{n_G} \\ &= \left[2^{n_G} \cdot \sum_{x=0}^{n_G-1} \binom{n_G}{x} \cdot 2^{-x} + \sum_{x=0}^{n_G-1} \binom{n_G}{x} x \right. \\ &\quad -2 \sum_{x=0}^{n_G-1} \binom{n_G}{x} - \sum_{x=0}^{m-1} \binom{n_G}{x} \\ &\quad \left. - \sum_{x=0}^{m-1} \binom{n_G}{x} \sum_{i=1}^{m-x} \binom{n_G - x}{i} \right] / 2^{n_G} \end{aligned} \tag{3.6}$$

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, $\sum_{k=0}^{n_G} k \binom{n_G}{k} = 2^{n_G-1} n_G$ [7] we obtain,

$$\begin{aligned} E(m, n_G) &= 2^{-n_G} \left[2^{n_G} \cdot \left((3/2)^{n_G} - 2^{-n_G} \right) \right. \\ &\quad + 2^{n_G-1} n_G - n_G - 2(2^{n_G} - 1) \\ &\quad \left. - \sum_{x=0}^{m-1} \binom{n_G}{x} \left(1 + \sum_{i=1}^{m-x} \binom{n_G - x}{i} \right) \right] \\ &= (3/2)^{n_G} + \frac{n_G}{2} - 2 - \frac{n_G - 1}{2^{n_G}} \\ &\quad - \frac{1}{2^{n_G}} \sum_{x=0}^{m-1} \binom{n_G}{x} \left(1 + \sum_{i=1}^{m-x} \binom{n_G - x}{i} \right) \end{aligned} \tag{3.7}$$

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

$$\begin{aligned} E(1, n_G) &= (3/2)^{n_G} + n_G/2 + n_G 2^{1-n_G} - 2 \\ &= O((3/2)^{n_G}) \end{aligned} \tag{3.8}$$

$$\begin{aligned} E(n_G - 1, n_G) &= (3/2)^{n_G} + n_G/2 - 2 - \frac{n_G - 1}{2^{n_G}} \\ &\quad -2^{-n_G} \sum_{x=0}^{n_G-1} \binom{n_G}{x} \left(2^{n_G - x} - 1\right) \\ &= (3/2)^{n_G} + n_G/2 - 2 - \frac{n_G - 1}{2^{n_G}} \\ &\quad - \sum_{x=0}^{n_G-1} \binom{n_G}{x} \cdot 2^{-x} \\ &\quad +2^{-n_G} \sum_{x=0}^{n_G-1} \binom{n_G}{x} \\ &= (3/2)^{n_G} + n_G/2 - 2 - \frac{n_G - 1}{2^{n_G}} \\ &\quad -(3/2)^{n_G} + 2^{-n_G} + 1 - 2^{-n_G} \\ &= n_G/2 - 1 - \frac{n_G - 1}{2^{n_G}} \\ &= O(n_G) \end{aligned} \tag{3.9}$$

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,

$$\begin{aligned} E(m,n_G) &= 2^{-n_G}\sum_{x=m+1}^{n_G-1}\binom{n_G}{x}\sum_{i=0}^{m}\binom{n_G}{i} \\ &+ 2^{-n_G}\sum_{x=1}^{m}\binom{n_G}{x}\left[\sum_{i=0}^{x-1}\binom{n_G}{i}+\binom{n_G}{x}/2\right] \end{aligned} \tag{3.10}$$

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

$$\begin{aligned} E(1,n_G) &= n_G+1-\frac{(n_G+1)^2+2}{2^{n_G+1}} \\ &= O(n_G) \end{aligned} \tag{3.11}$$

$$\begin{aligned} E(n_G-1,n_G) &> 2^{-n_G}\cdot\sum_{x=1}^{n_G-1}\binom{n_G}{x}\binom{n_G}{x}/2 \\ &= 2^{-n_G-1}\cdot\left[\sum_{x=0}^{n_G}\binom{n_G}{x}^2-2\right] \\ &= 2^{-n_G-1}\cdot\binom{2n_G}{n_G}-2^{-n_G} \\ &\geqslant 2^{-n_G-1}\frac{4^{n_G}}{\sqrt{4n_G}}-2^{-n_G} \\ &= \frac{2^{n_G}}{4\sqrt{n_G}}-2^{-n_G} \\ &= O\left(2^{n_G}n_G^{-1/2}\right) \end{aligned} \tag{3.12}$$

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,

$$p = 1 - \frac{\binom{n_G - x}{m-x}}{\binom{n_G}{m}} \tag{3.13}$$

$$q = \frac{\binom{n_G - x}{m-x}}{\binom{n_G}{m}} \tag{3.14}$$

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

$$\begin{aligned} H_m(x) &= q + 2pq + 3p^2q + \ldots + \binom{n_G}{m} p^{\binom{n_G}{m}-1} q \\ &= q \sum_{k=1}^{\binom{n_G}{m}} k p^{k-1} \\ &= q \frac{(p-1)\binom{n_G}{m} p^{\binom{n_G}{m}} - p^{\binom{n_G}{m}} + 1}{(p-1)^2} \end{aligned} \tag{3.15}$$

The overall average number of privacy checks follows the Equation 3.3, and can be simplified,

$$\begin{aligned} E(m, n_G) &= \frac{\sum_{x=0}^{m-1} \binom{n_G}{x} H_m(x) + \sum_{x=m}^{n_G-1} \binom{n_G}{x} H_M(x)}{2^{n_G}} \\ &= \sum_{x=0}^{m-1} \binom{n_G}{x} \frac{1 - \left(\binom{n_G - x}{m-x} + 1\right) p^{\binom{n_G}{m}}}{2^{n_G} q} \\ &\quad + \binom{n_G}{m} \left(1 - \frac{\sum_{x=0}^{m-1} \binom{n_G}{x} + 1}{2^{n_G}}\right) \end{aligned} \tag{3.16}$$

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

$$p = 1 - \frac{\binom{n_G - x}{1-x}}{n_G} \tag{3.17}$$

$$q = \frac{\binom{n_G - x}{1-x}}{n_G} \tag{3.18}$$

$$\begin{aligned} E(1, n_G) &= \frac{1 - (n_G + 1) \cdot 0^{n_G}}{2^{n_G}} + n_G(1 - 2^{1-n_G}) \\ &= n_G + \frac{1 - 2n_G}{2^{n_G}} \\ &= O(n_G) \end{aligned} \tag{3.19}$$

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

$$p = 1 - \frac{\binom{n_G - x}{n_G/2 - x}}{\binom{n_G}{n_G/2}} \tag{3.20}$$

$$q = \frac{\binom{n_G - x}{n_G/2 - x}}{\binom{n_G}{n_G/2}} \tag{3.21}$$

$$\begin{aligned} E(\frac{n_G}{2}, n_G) &= \sum_{x=0}^{\frac{n_G}{2}-1} \binom{n_G}{x} \frac{1 - \left(\binom{n_G - x}{n_G/2 - x} + 1\right) p^{\binom{n_G}{n_G/2}}}{2^{n_G} q} \\ &\quad + \binom{n_G}{n_G/2} \left(1/2 + \frac{\binom{n_G}{n_G/2} - 1}{2^{n_G}}\right) \end{aligned} \tag{3.22}$$

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 bound $E(n_G/2, n_G)$ as follows,

$$E(n_G/2, n_G) = O\left(2^{n_G} n_G^{-1}\right) \tag{3.23}$$

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 finish the current iteration after finding an x-adversary. In order to do so, it will run $\log_2(n_G - m - 1)$ privacy checks. Then, it will prune downwards all subcoalitions of x-adversary, including $\binom{x}{x-m}$ m-adversaries. The minimal number of iterations that are necessary to prune all m-adversaries is equal to $\binom{n_G}{m}/\binom{x}{x-m}$. Thus, the expected number of privacy checks follows the formula,

$$\begin{aligned} E(m, n_G) &= \frac{m+1}{n_G - 1} \\ &\quad + \binom{n_G}{m} \frac{\log_2(n_G - m - 1)}{n_G - 1} \sum_{x=m+1}^{n_G - 1} \frac{\binom{n_G}{x}}{\binom{x}{m}} \end{aligned} \tag{3.24}$$

$$
\begin{aligned}
E(1, n_G) &= \frac{2}{n_G - 1} + \frac{n_G \log_2(n_G - 2)}{n_G - 1} \sum_{x=2}^{n_G - 1} \frac{\binom{n_G}{x}}{x} \\
&\geqslant \frac{2}{n_G - 1} + n_G \frac{\log_2(n_G - 2)}{(n_G - 1)^2} (2^{n_G} - n_G - 2) \\
&= O\left(2^{n_G} \cdot \frac{\log_2 n_G}{n_G}\right)
\end{aligned} \tag{3.25}
$$

$$
\begin{aligned}
E(\frac{n_G}{2}, n_G) &= \frac{n_G/2 + 1}{n_G - 1} \\
&\quad + \binom{n_G}{n_G/2} \frac{\log_2(\frac{n_G}{2} - 1)}{n_G - 1} \sum_{x=n_G/2+1}^{n_G - 1} \frac{\binom{n_G}{x}}{\binom{x}{n_G/2}} \\
&\geqslant \frac{n_G/2 + 1}{n_G - 1} + 2^{n_G - 4} \cdot \frac{\log_2(\frac{n_G}{2} - 1)}{n_G - 1} \\
&= O\left(2^{n_G} \cdot \frac{\log_2 n_G}{n_G}\right), \text{ for } n_G \geqslant 4
\end{aligned} \tag{3.26}
$$

$$
E(n_G - 2, n_G) = 1 \tag{3.27}
$$

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 3.2.

An m-adversary I can use any of its records in its attacks. Thus, to ensure m-privacy, all possible subsets of these 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,

$$
\sum_{i=0}^{m} \sum_{\substack{I \in P \\ |I| = i}} \prod_{R \in I} 2^{|records\_of(R)|} \tag{3.28}
$$

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 a scenario with an $(m-1)$-adversary that has been already verified. 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 the number of attackers is increased 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 in attacks 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 3.2.

Conclusions. The average time complexity analysis is more involved, and its results depend on the parameter m. For each of them the lower bound of the average time complexity is $O(n_G)$, but the upper bound is different, 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*, and $O\left(2^{n_G} \frac{\log_2 n_G}{n_G}\right)$ for the *binary*. Thus, adapting verification strategy to different settings is crucial to achieve, on average, a low runtime.

3.4 Anonymization for m-Privacy

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

For a privacy constraint C that is generalization monotonic, m-privacy w.r.t. C is also generalization monotonic (Theorem 3.7), and most existing generalization-based anonymization algorithms can be easily modified to guarantee m-privacy w.r.t. C. The adoption is straightforward, every time a set of records is tested for privacy fulfillment,

we check m-privacy w.r.t. C instead. As a baseline algorithm to achieve m-privacy, we adapted the multidimensional Mondrian algorithm [56] designed for k-anonymity. The main limitation of such adaptation is that groups of records are formed *oblivious* of the data providers, which may result in over-generalization in order to satisfy m-privacy w.r.t. C.

Anonymization Algorithm. We introduce a simple and general algorithm based on the Binary Space Partitioning (BSP) (Algorithm 4). Similar to the Mondrian algorithm, it recursively chooses an attribute to split data points in the multidimensional domain space until the data cannot be split any further without breaching m-privacy w.r.t. C. However, the algorithm has three novel features: 1) it takes into account the data provider as an additional dimension for splitting; 2) it uses the privacy fitness score as a general scoring metric for selecting the split point; 3) it adapts its m-privacy checking strategy for efficient verification. The pseudo code for our *provider-aware* anonymization algorithm is presented in Algorithm 4.

Algorithm 4: The *provider-aware* anonymization algorithm.

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

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

Provider-Aware Partitioning. The algorithm first generates all possible splitting points, π, for QI attributes and data providers (lines 1 to 2). In addition to the multidimensional QI domain space, we consider the data provider of each record as its additional attribute A_0. For instance, each record t contributed by data provider P_1 will have $t[A_0] = P_1$. Introducing this additional attribute adds also a new dimension for partitioning. Using A_0 to split data points decreases number of providers in each partition, and hence increases the chances that

more sub-partitions will be m-private, and feasible for further splits. This leads to a more precise view of the data, and have a direct impact on the anonymized data utility. To find the potential split point along this dimension, we impose a total order on the providers, e.g., sorting the providers alphabetically or based on the number of records they provide, and partition them into two groups with approximately the same size.

Adaptive Verification for EG-Monotonic m-Privacy. m-Privacy is then verified for all possible splitting points, and only those satisfying it are added to a candidate set π' (line 3). In order to minimize the time, our algorithm adaptively selects an m-privacy verification strategy using the fitness score of the partitions. Intuitively, in the early stage of the anonymization algorithm, the partitions are large and likely m-private. The *top-down* algorithm, which takes advantage of the downward pruning, may be used for fast privacy verification. However, as the algorithm continues, the partitions become smaller, the downward pruning is less likely, and the *top-down* algorithm will be less efficient. The *binary* algorithm may be used instead to take advantage of upward pruning. We experimentally find the threshold of privacy fitness score for selecting the best algorithm, and confirm the benefit of this strategy.

Privacy Fitness Score Based Splitting Point Selection. Given a non-empty candidate set π' (Algorithm 4), the privacy fitness score (Definition 3.9) is used to find the best split (line 7). Intuitively, if the resulting partitions have higher fitness scores, they are more likely to satisfy m-privacy, and allow further splitting. Notice that the fitness score does not have to be exactly the same function used for adaptive ordering in m-privacy check. For example, if in the Equation 3.1, the weight parameter used to balance fitness values of privacy constraints, should have, most likely, different value. After choosing the splitting point, the partition is divided, and the algorithm is run recursively on each sub-partition (lines 8 and 9).

3.5 Experimental Evaluation

3.5.1 Experiment Setup

We merged the training and testing sets of the Adult dataset[2] into a single data set. 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. Records are randomly distributed among n providers following uniform or exponential distribution for non-EG and EG monotonic constraints, respectively.

Privacy Constraints. 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. The EG monotonic privacy constraint is defined as a conjunction of k-anonymity [85] and l-diversity [60]. Both m-privacy verification and anonymization algorithms use privacy fitness scores (Equation 3.1), but with different values of the weight α. Values of α can be defined in a way that reflects restrictiveness of privacy constraints. The impact of the weight to overall performance was experimentally investigated and values of α for the most efficient runs have been chosen as default.

All experiments have performed on Sun Microsystems SunFire V880 with 8 CPUs, 16 GB of RAM, and running Solaris 5.10. All algorithm parameters, and their default values are listed in the Table 3.1.

Table 3.1: Experiment parameters and default values for experiments with EG and non-EG monotonic constraints, which are outside and within parentheses, respectively.

Name	Description	Verification	Anonymization
m	Power of m-privacy	5 (3)	3 (1)
n	Number of data providers	–	10 (5)
n_G	Number of data providers contributing to a group	15 (5)	–
$\|T\|$	Total number of records	–	45222 (30)
$\|T_G\|$	Number of records in a group	{150, 750} (25)	–
k	Parameter of k-anonymity	50 (4)	30 (4)
l	Parameter of l-diversity	4	(3)
t	Parameter of t-closeness	(0.5)	(0.5)

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

Metrics. The efficiency of algorithms is measured by their runtime. To evaluate the utility of the anonymized data, we used the query error metric defined similar to prior work [22, 91]. 2,500 queries have been randomly generated, and each query had qd predicates p_i, defining a range of a randomly chosen quasi-identifier, where $qd \in \left[2, \frac{q}{2}\right]$ and q is the number of quasi-identifier attributes,

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

Query error is defined as the normalized difference in the results Q coming from anonymized and original data: $query\_error = (Q(T^*) - Q(T))/Q(T)$.

3.5.2 m-Privacy Verification

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

Attack Power. In this experiment we compare m-privacy verification heuristics against different attack powers. We use two different groups of records with relatively small and large average numbers of records per data provider. Figure 3.4 shows the runtime with varying m for all heuristics for the former group of records.

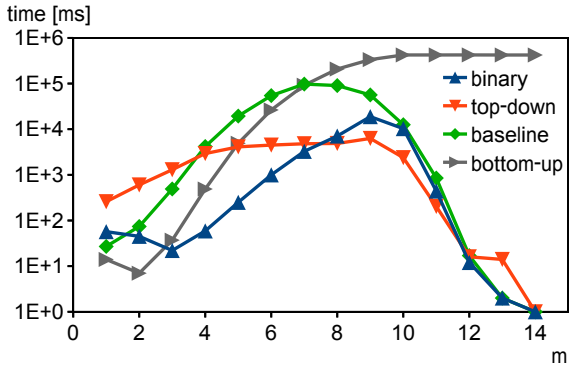


Figure 3.4: Runtime (logarithmic scale) vs. power of m-privacy for $|T_G|/n_G = 10$.

The group has 150 records, and small average fitness score per provider (equal to 0.867), which reflects to 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, performance of the *top-down* algorithm is comparable with *binary* for $m > n_G/2$.

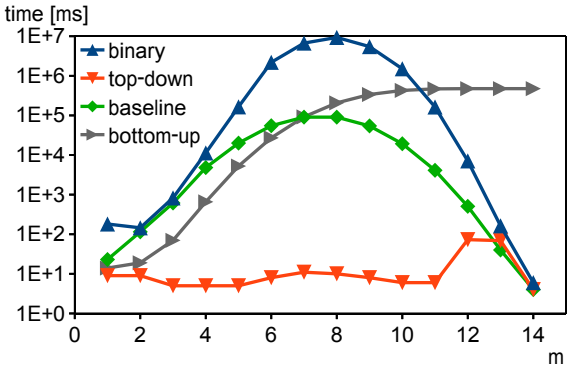


Figure 3.5: Runtime (logarithmic scale) vs. power of m-privacy for $|T_G|/n_G = 50$.

Figure 3.5 shows the runtime with varying m for all heuristics for the group 750 records, and a larger average fitness score per provider (equal to 2.307). Therefore intuitively, it is very unlikely that an m-adversary may breach privacy, and the downward pruning can be applied often. This intuition is confirmed by results, which show that the *top-down* algorithm is significantly faster 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, e.g., $m \approx n_G/2$ incurs the longer run.

Number of Contributing Data Providers. In this experiment we analyze the impact of increasing number of data providers, n_G, on the different algorithms for the small and large set of records, respectively. Notice that the average number of records per provider is constant. Figure 3.6 and Figure 3.7 show the runtime of different heuristics with varying n_G.

We observe that increasing the number of contributing data providers has different impact on different algorithms in both group settings. In the first group (Figure 3.6), 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. Increasing the number of providers will make the domain of possible m-adversaries, which are considered exponentially bigger.

Similar trend is found for the other group (Figure 3.7) for *binary*, *direct*, and *bottom-up*

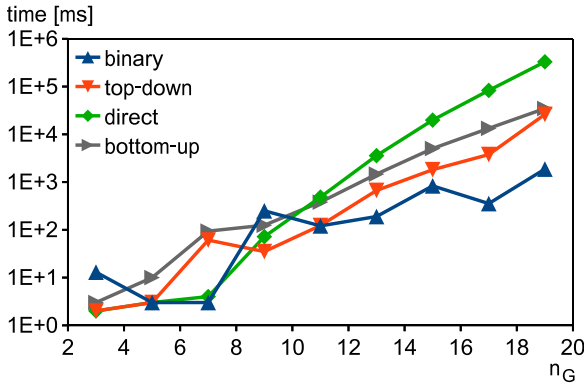


Figure 3.6: Runtime (logarithmic scale) vs. number of data providers for $|T_G|/n_G = 10$.

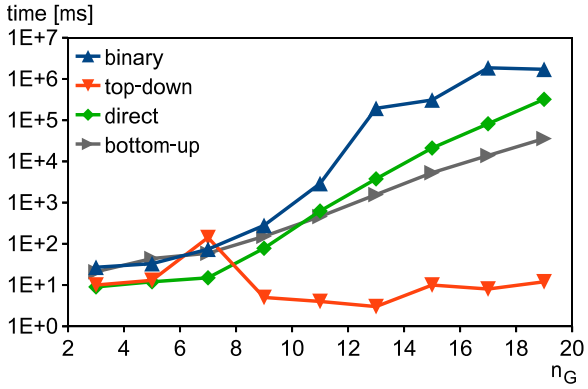


Figure 3.7: Runtime (logarithmic scale) vs. number of data providers for $|T_G|/n_G = 50$.

algorithms. For the *top-down* algorithm runtime grows linearly with the number of providers, which is due to its effective use of downward pruning.

The Average Number of Records Per Provider. In this experiment we systematically evaluate the impact of the average number of records per provider ($|T_G|/n_G$) on the efficiency of the algorithms. Figure 3.8 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 the large average number of records per provider, i.e., when $|T_G|/n_G \geqslant 15$, the *top-down* algorithm outperforms others.

Figure 3.9 presents the runtime with varying the average fitness score of contributing providers. It yields an almost identical trend as the result for average number of records per provider (Figure 3.8). 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.

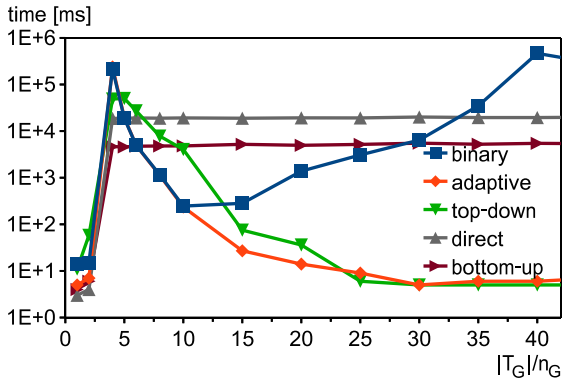


Figure 3.8: Runtime (logarithmic scale) vs. $|T_G|/n_G$.

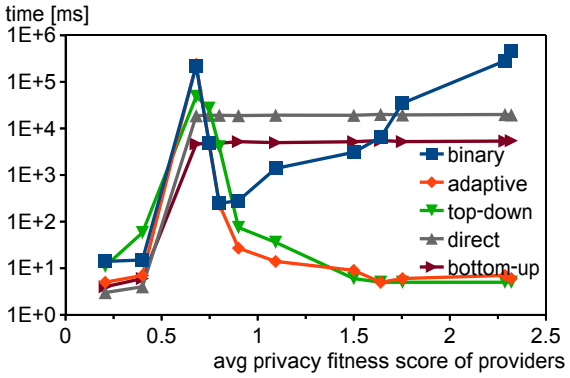


Figure 3.9: Runtime (logarithmic scale) vs. the average fitness score of data providers.

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.

3.5.3 m-Privacy Anonymization

This set of experiments compares our *provider-aware* algorithm with the *baseline* algorithm, and evaluates the benefit of provider-aware partitioning for m-privacy w.r.t. an EG monotonic constraint.

Attack Power. We first evaluate both anonymization heuristics with varying attack power m. Figure 3.10 shows the runtime with varying m for both algorithms. As a reference we added results of anonymization applied independently by each data provider. Since its

runtime and query error are independent of m, and can be run in parallel, it outperform other approaches, but anonymized data have low utility.

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 3.6 and Figure 3.7). 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 $m \approx n/2$.

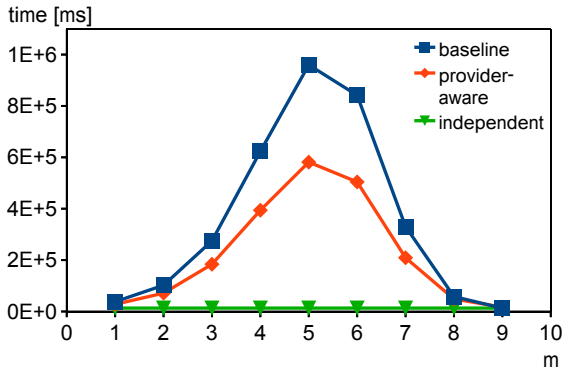


Figure 3.10: Runtime vs. power of m-privacy.

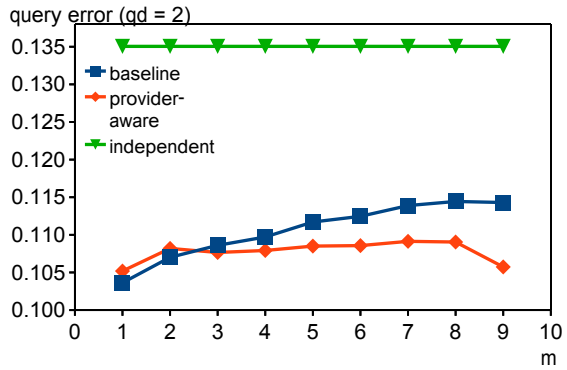


Figure 3.11: Query error vs. power of m-privacy.

Figure 3.11 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 groups with increasing m. In contrast, the *provider-aware* algorithm takes into account the data providers, and returns groups with smaller number of contributing providers (on average 1 for $k = 15$). Therefore, it maintains a more precise view of the data, and significantly outperforms the *baseline* algorithm. The query error may increase with m eventually, but it will not be as significant growth as for the *baseline* algorithm.

Number of Data Records. This set of experiments evaluates algorithms for different dataset sizes. Figure 3.12 shows the runtime with varying number of records for both algorithms. As a reference we added results of anonymization applied independently by each data provider. However, its query error (not presented), is on average 40% greater than for data anonymized by running other 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 caused by the significant reduction of the verification time in our algorithm, which limits the number of providers represented in each group. The query error (not presented) is at the same rate for both algorithms.

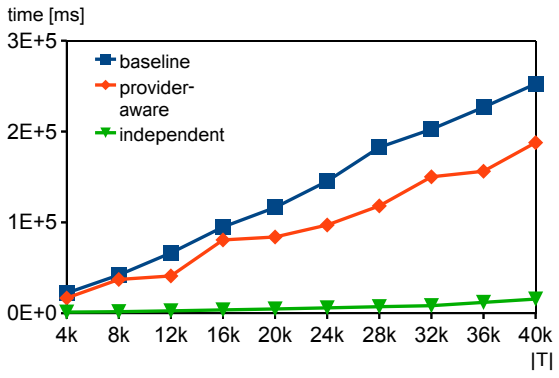


Figure 3.12: Runtime (logarithmic scale) vs. $|T|$ for anonymization algorithms.

Adaptive m-Privacy Verification. In this experiment we evaluate the benefit of the

adaptive selection of a m-privacy verification algorithm. Figure 3.13 presents runtimes of adaptive anonymization algorithm using selected verification strategies with varying $|T|$. For small values of $|T|$, the algorithm using adaptive verification strategy follows the *binary*, and, for more records, the *top-down* algorithm, as we expected. However, for values of $|T| > 300$, our algorithm outperforms the non-adaptive strategies. The reason is that anonymization of numerous records requires verification of m-privacy for many subgroups of different sizes. Adapting to such variety of groups results in higher efficiency comparing to the choice of a single strategy.

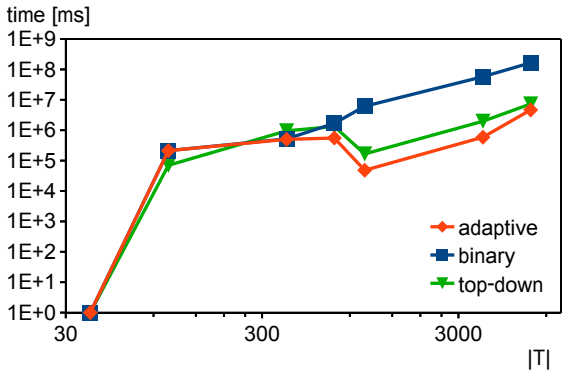


Figure 3.13: Runtime (logarithmic scale) vs. $|T|$ for different verification strategies.

Impact of Privacy Constraints. We 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 3.14 and Figure 3.15 show runtime and query errors with varying privacy constraint restrictiveness (varying k and l, respectively).

As expected, more restrictive constraints, i.e., greater values of k or l, require more records or more distinct values of the sensitive attribute in each QI group, and thus results in higher query error. However, execution times are shorter comparing to *weaker* privacy constraints, which is a consequence of fewer partitions.

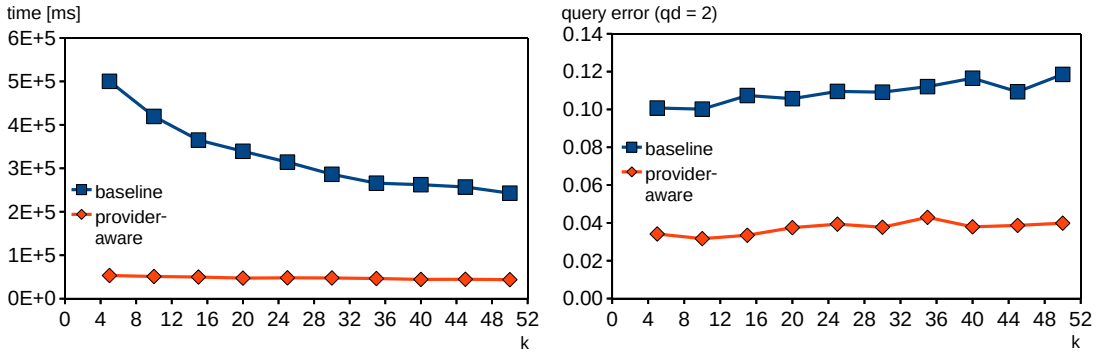


Figure 3.14: Runtime and query errors vs. k in m-privacy with respect to k-anonymity.

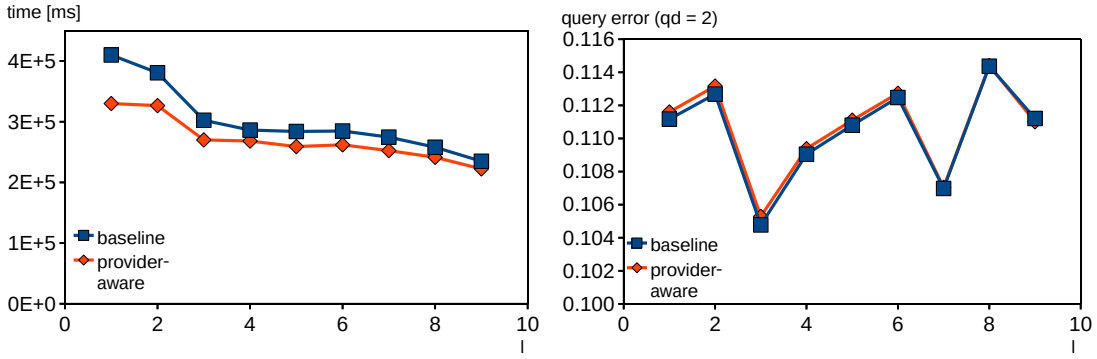


Figure 3.15: Runtime and query errors vs. l in m-privacy with respect to l-diversity.

3.5.4 m-Privacy Verification Experiments for non-EG Monotonic Constraints

The goal of this set of experiments is to evaluate the efficiency and complexity of the m-privacy verification algorithm w.r.t. a non-EG monotonic constraint C.

Attack Power. In this experiment we present the impact of the m-privacy power m for different sets of records. We use four different sets of records, generated randomly and independently from each other. Figure 3.16 shows the runtime with varying m for different sets of records. For each set the maximal runtime is reached in the maximal value of m for which records are m-private ($m = 2$ for $|T_G| = 15$, and $m = 3$ for others). For lower values, the runtime increases exponentially, while for greater values drops significantly, which is caused by early identification of attacking records that can breach privacy (*early stop*).

Number of Contributing Data Providers. In this experiment we analyze the impact of

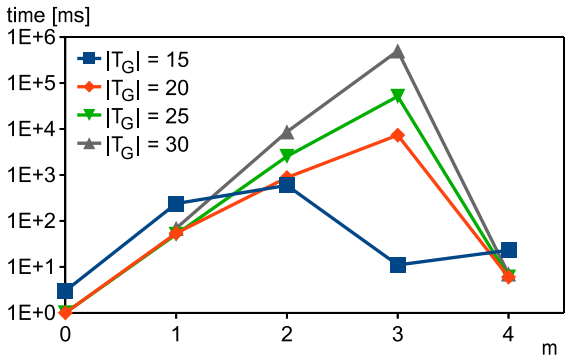


Figure 3.16: Runtime (logarithmic scale) vs. power of m-privacy.

increasing the number of data providers, n_G, while preserving the average number of records per provider for m-privacy verification ($m = 3$). Figure 3.17 shows the runtime with varying n_G for a few different values of the average number of records per provider. As expected, the m-privacy verification runtime increases exponentially with number of contributing providers for m-private sets of records ($m = 3$). For sets, which are not m-private, i.e., for $n_G = 3$, and for $n_G = 5$ with $|T_G|/n_G = 5$, their runtimes are very low, due to early finding a set of attacking records that breaches privacy.

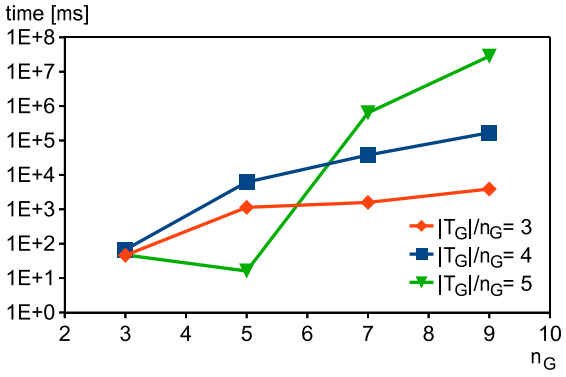


Figure 3.17: Runtime (logarithmic scale) vs. number of data providers.

3.5.5 m-Privacy Anonymization Experiments for non-EG Monotonic Constraints

In this set of experiments we analyze performance of the m-privacy anonymization algorithms w.r.t. a non-EG monotonic constraint C for different parameter values. The

privacy constraint is defined as a conjunction of k-anonymity [85] and t-closeness [57].

Attack Power. We first evaluate the impact of varying size of malicious coalitions. Figure 3.18 and Figure 3.19 show, respectively, the runtime and query errors for different powers of m-privacy, i.e., the parameter m.

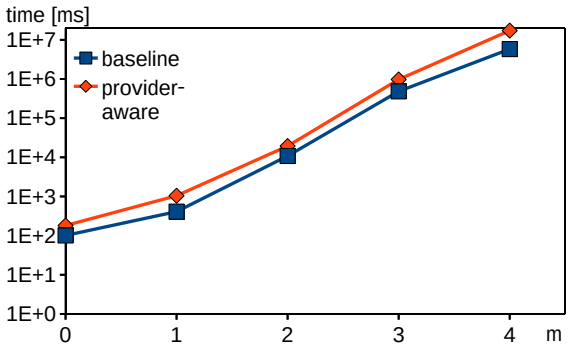


Figure 3.18: Runtime (logarithmic scale) vs. power of m-privacy.

As expected, runtimes for both algorithms increase exponentially with m (Figure 3.18). The *provider-aware* algorithm runs slightly longer, which is expected due to consideration of additional splits of records, but the query error is significantly lower.

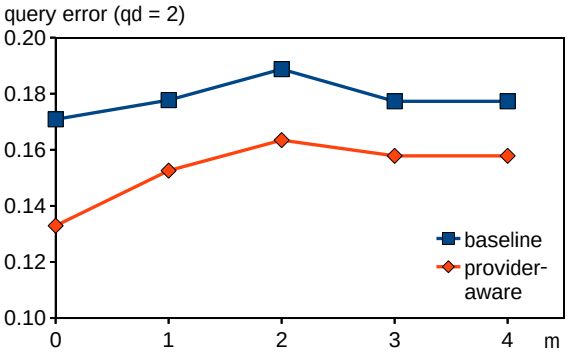


Figure 3.19: Query error vs. power of m-privacy.

We expect that increasing restrictiveness of the m-privacy would increase query error as well (Figure 3.19). This intuition is confirmed for low values of m, but for ($m > 2$) query errors slightly decrease, and maintain the same level of values. This counter-intuitive behavior is a side effect of the dataset size, for which it is very likely to get an anonymized m-private ($m = 3$) dataset, which is also m-private ($m = 4$). The same reason stays behind

a higher error rate for m-private ($m = 2$) anonymized dataset.

Number of Data Records. The goal of this experiment is to evaluate our algorithm for different number of records. Each bigger set of records is a superset of the smaller set considered earlier. Figure 3.20 and Figure 3.21 present runtime and query error results for different numbers of records, respectively.

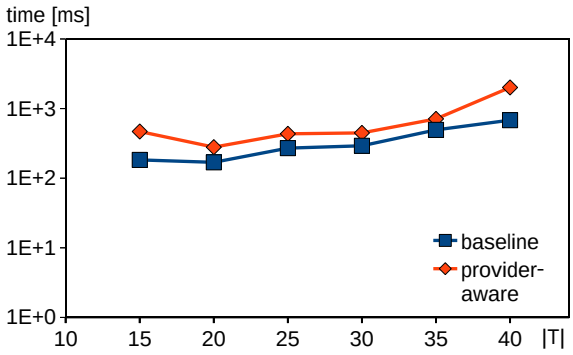


Figure 3.20: Runtime (logarithmic scale) vs. number of records.

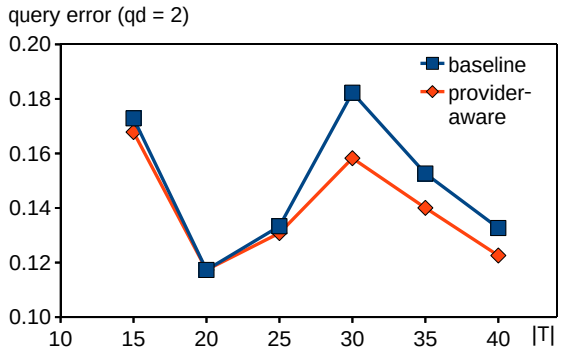


Figure 3.21: Query error vs. number of records.

Figure 3.20 confirms our intuition that the runtime increases exponentially with the 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 depends less on the size of the dataset (Figure 3.21). At the first glance this seems to be counter-intuitive, but it can be explained. Adding records increases query error, but only up to some point, after which the anonymization algorithm is able to generate more

QI groups, and the error of query answers decreases. Thus, the query error is 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 achieves lower query errors for larger datasets.

Privacy Constraints. We also perform a set of experiments evaluating the impact of the privacy constraint restrictiveness on the utility of anonymized data. In these experiments we evaluate our algorithm for different levels of privacy constraints restrictiveness, 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 it more restrictive), while t-closeness is disproportional to values of t (i.e., increasing t makes it less restrictive). Figure 3.22 and Figure 3.23 show runtime and query error, while varying k and t values, respectively.

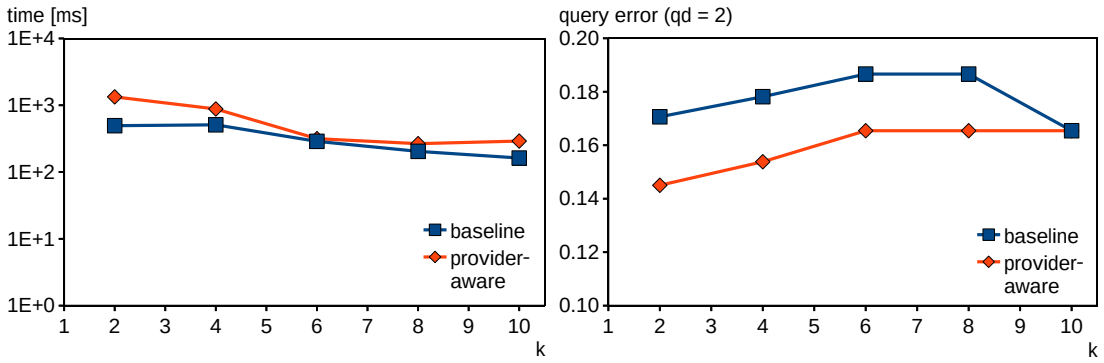


Figure 3.22: Runtime (logarithmic scale) and query errors vs. k in k-anonymity used in a privacy constraint C.

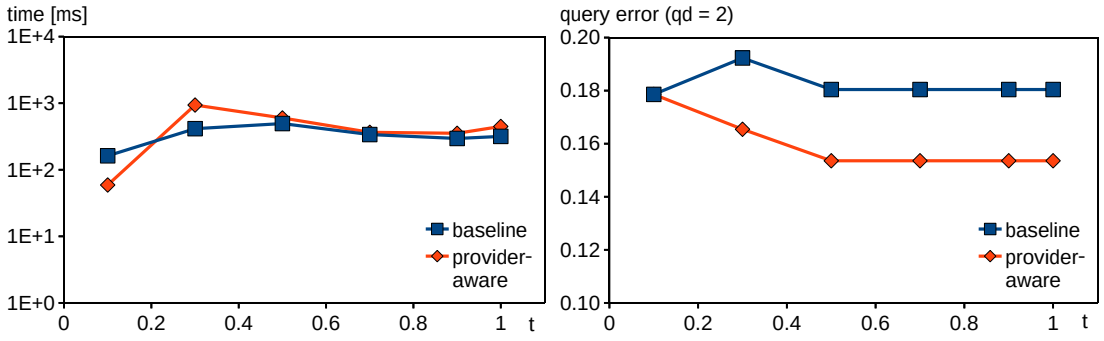


Figure 3.23: Runtime (logarithmic scale) and query errors vs. t in t-closeness used in a privacy constraint C.

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 the presence 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 confirm also that our algorithm returns anonymized data with the same or better utility than the *baseline* algorithm in all scenarios. The price for preserving more utility is just a slight increase of the runtime.

Chapter 4

Secure Multiparty Data Aggregation with m-Privacy

4.1 Introduction

In Chapter 3 we assumed that there is a trusted third-party (TTP) that collects data from all providers and runs all necessary algorithms. For settings without such party, data providers need to run an SMC protocol. We assume that all providers are semi-honest (honest but curious), i.e., they follow each protocol they run, but inspect all intermediate results of computations.

In this chapter we present SMC protocols for that implement all presented m-privacy verification and anonymization algorithms as well as subprotocols to verify privacy fulfillment and anonymize records. SMC protocols are designed and implemented in the Shamir's secret sharing scheme [81], but encryption and other secure schemas are also employed. In a secret sharing scheme, the owner of a secret message s prepares and distributes n shares, such that each party gets a few shares (usually one). An algorithm reconstructing s requires any r shares as its input. To prevent any coalition of up to m providers to reveal intermediate results, we set $r = m+1$. Notice that receivers of shares do not have to be neither providers nor trusted. They could be run as separate processes within a distributed environment (e.g. cloud) and computations still would stay information-theoretically secure [6]. The

implementation and complexity analyzes have been built on top of the SEPIA framework [14, 103].

4.2 Secure Privacy Constraint Verification Protocols

To allow using any privacy constraint in our m-privacy verification protocol, secure privacy verification is implemented as a separate protocol, and results of its runs are disclosed. Some secure m-privacy verification protocols designed for a specific privacy constraint C may disclose less intermediate computation results than our approach, but will be limited only to a single privacy constraint and will not be able to deal with other constraints. In addition, changing C requires redesigning the whole protocol from scratch. Presenting verification protocols for any privacy constraint is out of the scope of this paper, but we present secure protocols to verify k-anonymity and l-diversity. All implementations use Shamir's secret sharing [81] as their main scheme. For a few subprotocols we use encryption (commutative, homomorphic, etc.), and other secure schemas for efficiency. Assume that there are n_G data providers, and each data provider P_i provides T_i records.

4.2.1 Secure k-Anonymity Verification

To securely verify k-anonymity, the leader counts all records $s = |T|$ using the secure sum protocol [20, 71, 82], and securely compares s with k. Our implementation of the secure sum protocol uses only Shamir's secret sharing scheme (Algorithm 5).

First, all data providers run secure sum protocol in order to compute total number of records s. To avoid disclosing s it is stored in distributed shares $[s]$ (line 1). Finally, all providers securely compare $[s]$ with k [14, 103]. As the result, each provider gets a share of 1 if k-anonymity holds or a share of 0 otherwise (line 2).

Algorithm 5: The secure k-anonymity verification protocol.

Data: $P_1, \ldots, P_{n_G}$ providing $T_1, \ldots, T_{n_G}$ records respectively.
Result: Each P_i gets $[1]_i$ if $s \geqslant k$, $[0]_i$ otherwise.

1 $[s] = secureSum\left(|T_1|, \ldots, |T_{n_G}|\right)$
2 **return** $1 - lessThan([s], k)$

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

Proof. Assuming secure communication channels, the Shamir's secret sharing scheme were proven correct and information-theoretically secure [6]. Thus, knowing up to m shares of any value does not disclose it. Correctness and security of both *secureSum* and *lessThan* subprotocols were proven in [14]. The protocol does not reveal anything, but the result of the comparison $s \geqslant k$. □

Complexity Analysis. Computation complexity of the protocol is equal to the sum of complexities for both subprotocols. In [14] complexities are given as functions of secure multiplications. Each secure multiplication requires additional shares generation, and secret reconstruction, which take $O(mn_G)$ time. Assuming that number of bits used to represent a number in our protocols is constant, secure comparison protocol requires constant number of multiplications, i.e., its time complexity is $O(mn_G)$. Secure sum protocol (including shares generation) has the same complexity. Thus, the overall time complexity is $O(mn_G)$.

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

4.2.2 Secure *l*-Diversity Verification

The goal of this protocol is to securely verify if the total number of sensitive values from all records, is at least l (Algorithm 6). The protocol has two phases. In the first phase, each data provider P_i finds the set of sensitive values S_i of its records. Then, it randomly generates p_i fake values, and adds them to S_i (line 1). Notice that each provider generates fake values from a different domain. In the last step of this phase, the leader runs the secure size of set union subprotocol to compute $\bar{s}$, i.e., the size of the set of sensitive values of all records with a few additional fake values (line 2). The subprotocol is run in the same way as the secure size of set intersection [20, 88] with a few minor modifications. Notice that the

use of commutative encryption scheme in the subprotocols ensures that duplicated sensitive values are properly handled.

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

Algorithm 6: The secure l-diversity verification protocol.

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

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

Proof. Using commutative encryption scheme in implementation of the $secureSizeOfUnion$ subprotocol guarantees its correctness and security. Adding distinct fake values ensures that the local number of sensitive values will not be disclosed. Since each data provider generates different fake values, the sum of their counts is equal to the count of their union. The only information that is revealed, is $\bar{s}$, i.e., the upper bound of the number of sensitive values. However, allowing large and random number of fake values guarantees the low probability of guessing the real number of sensitive values. The second phase of the protocol utilizes Shamir's secret sharing scheme for secure sum, and comparison subprotocols, which are secure [6, 14]. Thus, the protocol is also secure. □

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

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

to $O(n_G^2)$.

Secure Privacy Verification. Above protocols return the verification result as shares of [1] if privacy constraint is fulfilled, and [0] otherwise. Each provider holds a single share for a constraint C_i. Any $r = m + 1$ providers are able to check if $C = C_1 \wedge \ldots \wedge C_w$ holds, by securely multiplying their results for all constraints, and comparing it against zero [14]. If the final reconstructed value is equal to 1, then C holds, otherwise it does not.

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

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

4.3 Secure m-Privacy Verification Protocols

In this section we present secure multiparty protocols to verify m-privacy w.r.t. a privacy constraint C. Notice that a secure m-privacy verification protocol for a non-EG monotonic constraint is an extension of the *bottom-up* approach (Algorithm 8).

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

To compute sums we run a secure sum protocol, which securely computes the sum of numbers held by providers. Implementation of such protocol is based on Shamir's secret sharing scheme, and has been introduced in SEPIA framework [14, 103]. Another protocol that is provided by SEPIA is secure comparison, which securely compares two numbers. By running this protocol for a set of numbers, we find the minimum and maximum values in the set, and set its elements in order.

In our protocols we also use secure size of set union subprotocol, which is a slight modification of the secure size of set intersection protocol [20]. The modification is to count all distinct encrypted items, and not only ones that are contributed by every provider.

Correctness, security, and complexity of these protocols and their implementations have been proven in [14, 20].

4.3.1 Secure Leader Election Protocol

All protocols are initiated by a leader P', i.e., a chosen provider, which is found by running a secure leader election protocol (SLE). Our SLE protocol (Algorithm 7) runs a secret sum protocol over randomly generated numbers in order to elect the leader. The implementation utilizes Shamir's secret sharing scheme, and does not disclose any information about data and its providers. The leader is considered untrusted, therefore any honest but curious party (also external) can participate in the election. Each data provider can simulate, monitor, and verify the leader actions to detect any malicious behavior. After running this protocol, each provider knows index of the leader from the list of all providers. In our algorithm $[r_i] = ([r_i]_1, \ldots, [r_i]_n)$ represents a vector of shares generated for a number r_i, which is owned by a provider P_i.

The protocol that is run by P_i works as follows. First, P_i generates a random number r_i, which shares are then distributed among other providers such that $[r_i]_j$ is sent to P_j. Then, P_i sums up all i^{th} shares, in order to compute $[r]_i$, and collaboratively reconstruct $r = \sum_{i=1}^{n} r_i$. Finally, the leader is identified as $P_{r \pmod n}$.

Algorithm 7: The Secure Leader Election protocol (SLE).

Data: A list of n data providers P.
Result: A chosen leader in P.

1 $r_i =$ `get_natural_random_number()`
2 $[r_i] =$ `get_secret_shares(`r_i`)`
3 To each $P_j \in P$ sends $[r_i]_j$, and receives $[r_j]_i$ from it.
4 Sums up all i^{th} shares: $\sum_{i=1}^{n}[r_i]_j = [\sum_{i=1}^{n} r_i]_j = [r]_j$
5 $r =$`reconstruct(`$[r]$`)`
6 **return** $P_{r \pmod n}$

Security. The SLE protocol is secure as long as communication channels among providers are secure [14]. With such assumption, and without loosing generality, let consider the worst-case scenario, i.e., $I = \{P_1, \ldots, P_{n-1}\}$ are malicious, and collude.

Since each data provider participates a random number r_i in order to compute $r =$

$\sum_{i=1}^{n} r_i$, colluding providers I are not able to bias the value of r. The presence of r_n in the sum guarantees that r is uniformly random, and unbiased.

Although malicious providers I can modify all shares they have access to, there modifications will not bias the randomness of r_n, which is enough to guarantee unbiased randomness of r. Due to lack of access to n^{th} share $[r_n]_n$, their modifications of other shares change r_n (and r) randomly. Thus, such random modification of r_n does not bias the randomness of the leader choice.

Complexity. Complexity of generating n shares, which are enough to reconstruct the original value, is equal to $O(n)$, and is caused by generating a polynomial of degree n, and generating n shares. Summing up n shares has complexity $O(n)$. Reconstruction of r requires running Lagrange interpolation algorithm, which complexity is equal to $O(n^2)$. Thus, the overall computation complexity is equal to $O(n^2)$.

Each provider, e.g. P_i, sends (and receives) $(n-1)$ shares before computing $[r]_j$. Then, $[r]_j$ is further sent to $(n-1)$ providers, while receiving other shares of r. Thus, P_i sends (and receives) $(2n-2)$ messages. The overall computation complexity is equal to $O(n^2)$.

4.3.2 Secure Sorting and Adaptive Ordering

The main responsibility of the leader is to determine m-privacy fulfillment with as little privacy checks as possible. Our heuristic minimizes the number of privacy checks by utilizing EG monotonicity of C and adaptive ordering of m-adversary generation (Section 3.3.2). To define such order, P' runs any sorting algorithm, which sorts providers by fitness scores of their local records, with all comparisons run securely.

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

Our implementations of secure sorting protocol utilizes the Shamir's secret sharing scheme with r shares required to reconstruct a secret. To ensure m-privacy we set $r = m+1$. Thus, for n_G data providers the protocol requires running a sorting algorithm, which takes $O(n_G \log n_G)$ secure comparisons. Each secure comparison has the same complexity, i.e., requires a few secure multiplications, where each multiplication takes $O(m^2)$ time [14]. Thus,

the secure sorting time complexity is equal to $O(m^2 n_G \log n_G)$. Each secure multiplication requires passing $n_G(n_G - 1)$ messages in total, although only $(m+1)^2$ of them are needed to get the result. Thus, the communication complexity is equal to $O(n_G^3 \log n_G)$.

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

4.3.3 Secure m-Privacy Verification Protocol

After finding the order of data providers, the leader P' starts verifying privacy for different coalitions of attackers, which are generated in specific order. A general scheme of secure m-privacy verification is the same for all heuristic algorithms (Algorithm 8). Common steps are as follows. In the main loop P' verifies privacy of records for m-adversaries until m-privacy can be decided (line 3). Notice that in order to determine m-privacy w.r.t. EG monotonic C, it is enough to check privacy for all scenarios with exactly m attackers (Corollary 3.8). In the loop, P' generates, and broadcasts a coalition of potential adversaries I, so each party can recognize its status (attacker/non-attacker) for the current privacy check. Then, the leader runs the secure privacy verification protocol for I (line 6). If privacy could be breached, and I has no more than m data providers, then the protocol stops, and returns negative answer (line 7). Otherwise, the information about privacy fulfillment is used to prune (upwards or downwards) a few potential m-adversaries (line 9). Finally, if m-privacy w.r.t. C can be decided, P' returns the result of the verification (line 10).

Secure m-Privacy Verification for the *binary* Algorithm. Similar to 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. For the *binary* algorithm, secure m-privacy verification protocol is also run by P', which executes all steps of the Algorithm 2. The only difference is privacy verification, which is implemented as an SMC protocol.

Proposition 4.3. *Assuming security of subprotocols, all m-privacy protocols are secure*

Algorithm 8: The secure m-privacy verification protocol w.r.t. EG monotonic constraint C for *top-down*, *bottom-up*, and *direct* algorithms; code run by the leading provider P'.

Data: List of providers P, an EG monotonic C, and the m.
Result: *true* if $\mathcal{A}_1(T)$ is m-private w.r.t. C, *false* otherwise.

```
1  sites = securely_sort_providers(P, increasing_order, score_F)
2  use_adaptive_order_generator(sites, m)
3  while is_m-privacy_decided() == false do
4      I = generate_next_coalition(P)
5      Broadcast coalition I.
       // Runs secure privacy verification protocol.
6      privacy_breached = is_privacy_breached_by(I)
7      if privacy_breached and |I| ⩽ m then
8          return false // early stop
9      prune_coalitions(I, privacy_breached)
10 return is_m-private()
```

except revealing results of potential attacks of generated m-adversaries.

Proof. Results of all privacy checks are publicly known, and, by applying pruning, one can determine privacy of records for a few potential m-adversaries. Thus, the security disclosure depends on data, and the sequence of generated m-adversaries I is very important to minimize it. In this proof, we analyze security for all heuristics that are presented above (Section 3.3.2).

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

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

If the records are not m-private w.r.t. C, i.e., there is an m-adversary that can breach privacy, perfect security of the protocol cannot be guaranteed. Due to pruning property all

heuristics reveal information about all coalitions from group 1), and about a single coalition of size up to m from group 2). In addition, *top-down* and *binary* algorithms reveal also results of privacy checks for coalitions from group 2) having more than m providers. □

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

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

In Algorithm 8, a single loop iteration executes following operations: generating next coalition of attackers ($O(\log n_G)$), broadcasting generated coalition ($O(\log n_G)$), verifying if m-privacy can be determined ($O(n_G)$), and pruning ($O(n_G)$). Among them privacy verification has the highest complexity. Assuming that its time complexity is equal to $\mathcal{V}$ (computed below), and complexity of a single verification loop is equal to $V = \mathcal{V} + O(n_G)$. The *direct* algorithm will check privacy for at most $\binom{n_G}{m}$ possible m-adversaries. Thus, the complexity of m-privacy verification is equal to $O\left(V \cdot n_G^m\right)$. The *bottom-up* algorithm will check 0-adversary (external data recipient) up to all m-adversaries, which requires $\sum_{i=0}^{m} \binom{n_G}{i} = O\left(n_G^m\right)$ checks, therefore for this case complexity is equal to $O\left(V \cdot n_G^m\right)$. The *top-down* algorithm will check all $(n_G - 1)$-adversaries first, then smaller coalitions down to all m-adversaries, which requires $\sum_{i=n_G-1}^{m} \binom{n_G}{i} = O\left(n_G^{n_G-1-m}\right)$ checks, and the overall complexity of the protocol is equal to $O\left(V \cdot n_G^{n_G-1-m}\right)$. The *binary* algorithm will run $\binom{n_G}{m}$ iterations with $O(\log(n_G - m))$ privacy checks in each of them. Thus, when used, the protocol time complexity is equal to $O\left(V \cdot n_G^m \log(n_G - m)\right)$.

Communication Complexity. During each loop iteration of the m-privacy verification protocol (Algorithm 8) the leader sends $(n_G - 1)$ messages to providers with information if they should act as attackers or not. Assume that $\mathcal{V}_C$ is a communication complexity for a privacy verification protocol (computed below), and $V_C = \mathcal{V}_C + n_G - 1$ is the total

communication. Thus, the total communication complexities depend on the number of privacy checks, which is different for each algorithm, i.e., *direct*, $O\left(V_C \cdot n_G^m\right)$; *bottom-up*, $O\left(V_C \cdot n_G^m\right)$; *top-down*, $O\left(V_C \cdot n_G^{n_G-1-m}\right)$; and *binary*, $O\left(V_C \cdot n_G^m \log\left(n_G - m\right)\right)$.

4.4 Secure m-Privacy Anonymization Protocols

Algorithm 4 can be executed in a distributed environment by a TTP or by all providers running an SMC protocol. In this section we present a secure protocol for semi-honest providers. As an SMC schema we use Shamir's secret sharing, but, when needed, we employ also encryption.

4.4.1 Secure *Provider-aware* Anonymization Protocol

The protocol uses already existing SMC protocols. The first step for all providers is to elect the leader P' by running a secure election protocol (Algorithm 7, [79]), which then runs Algorithm 9.

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

Secure anonymization protocol runs as follows. First, the median of each attribute A_i is found by running the secure median protocol (line 4, [2]). All records with the A_i values less than the median, and some records with the A_i values equal to the median establish the distributed set $T^{s,i}$. Remaining records define the distributed set $T^{g,i}$. Then, m-privacy w.r.t. C is verified for $T^{s,i}$ by running the secure verification protocol, i.e., either Algorithm 8 or 3 (line 8). If $\mathcal{A}_1\left(T^{s,i}\right)$ is m-private w.r.t. C, then the same verification protocol is run for $T^{g,i}$ (line 11). If $\mathcal{A}_1\left(T^{g,i}\right)$ is also m-private w.r.t. C, then this split becomes a candidate split. For each candidate split, minimum fitness score of $T^{s,i}$ and $T^{g,i}$ is computed (secure fitness score protocol is described below). Among candidate splits, the one with the maximal fitness score is chosen, and the protocol is run recursively for its subpartitions (lines 21 to 22). If no such attribute can be found for any group of records, the protocol stops.

Secure m-privacy anonymization protocol calls three different SMC subprotocols: the

Algorithm 9: The secure *provider-aware* anonymization protocol.

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

Result: An anonymized view of distributed records $\mathcal{A}(T)$ that is m-private w.r.t. C.

$i_{max} = -1$
$[f_{max}] = [0]$
foreach $i \in \{0, \ldots, q\}$ **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.
    Locally split set T_j into $T_j^{s,i} = \{t \in T_j : t[A_i] < s_i\}$, and $T_j^{g,i} = \{t \in T_j : t[A_i] > s_i\}$.
    Locally distribute records $\{t \in T_j : t[A_i] = s_i\}$ among $T_j^{s,i}$ and $T_j^{g,i}$ to reduce their uneven distribution.
    Securely verify m-privacy w.r.t. C of a distributed set $T^{s,i} = \bigcup_{j=1}^{n} T_j^{s,i}$ (using Algorithm 8 or 3).
    if $T^{s,i}$ *is not m-private w.r.t.* C **then**
        continue
    Securely verify m-privacy w.r.t. C of a distributed set $T^{g,i} = \bigcup_{j=1}^{n} T_j^{g,i}$ (using Algorithm 8 or 3).
    if $T^{g,i}$ *is not m-private w.r.t.* C **then**
        continue
    $[f(T^{s,i})]$ =`secure_fitness_score(`$T^{s,i}$`)`
    $[f(T^{g,i})]$ =`secure_fitness_score(`$T^{g,i}$`)`
    $[f] = \min([f(T^{s,i})], [f(T^{g,i})])$
    if $reconstruct(lessThan([f_{max}], [f])) == 1$ **then**
        $[f_{max}] = [f]$
        $i_{max} = i$
if $i_{max} \geqslant 0$ **then**
    Run this protocol for $T^{s,i_{max}}$.
    Run this protocol for $T^{g,i_{max}}$.

secure median [2, 13], the secure m-privacy verification (Section 4.3), and the secure fitness score (Algorithm 10). The last protocol needs to be defined for each privacy constraint C (described below). For the sake of this analysis, we assume that all these protocols are perfectly secure, i.e., all intermediate results can be inferred from the protocol outputs.

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

Theorem 4.4. *The m-privacy anonymization protocol is secure except median values for each attribute, m-privacy fulfillments for records split by these medians, and the order of fitness score values for m-private QI groups.*

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

If none of possible splits is m-private, then the simulator finishes splitting the current set of records. No other intermediate and undisclosed results are computed during the protocol computation. Finally, since the secure median protocol, and the m-privacy verification protocol, as well as the secure fitness score protocol are assumed to be secure, and from the composition theorem [39] the m-privacy anonymization protocol is secure as well. □

Complexity Analysis. Before analyzing complexity of the secure anonymization protocol, let us make a note about complexity of the secure median protocol. A secure median protocol for an attribute A_i uses the binary search to find the median. To verify if the

median is found, one needs to make sure that there are $n/2$ records with values of A_i not greater and not less than the value, i.e., if both sets split by the value are $n/2$-anonymous (Algorithm 5). The time complexity of such protocol is equal to $O(n^2 \log(domain(A_i)))$. The communication complexity is also equal to $O(n^2 \log(domain(A_i)))$.

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

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

4.4.2 Secure Fitness Score Protocol

Many privacy constraints (including ones we have used in our running example) base on threshold values $\mathcal{T}$. In order to securely compare fitness scores of constraints, they need to be *scaled*, e.g., using the least common multiple (lcm) of all threshold values. After that the secure fitness score can be computed by running the following protocol (Algorithm 10). The elected leader computes the least common multiple of all thresholds from the privacy constraints (line 1). Then, values measured, and compared with thresholds in each privacy constraints can be securely computed (line 3), and *scaled* (line 4). Shares of the minimal one are scaled back, and returned (line 5).

In our running example, we require fulfillment of k-anonymity and l-diversity. Thus, for P_i, $\gamma_1 = |T|$, and γ_2 is equal to the number of distinct sensitive values of local records T. In order to compute γ_1 and γ_2, we run secure k-anonymity, and l-diversity protocols (Algorithm 5 and Algorithm 6 respectively). However, in both protocols we skip comparison of computed values with their thresholds (k and l, respectively), and return shares of such values.

Algorithm 10: The secure fitness score protocol.

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

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

The Shamir's secret sharing scheme, with secure communication channels, is information-theoretically secure [6]. Correctness and security of the *multiplicate* subprotocol has been discussed in details in [14]. The above protocol reveals the fitness score value. However, if this protocol is used as a subprotocol, and revealing of the minimal fitness score value is not necessary, then the protocol would return shares of the minimal value, i.e., $\min([F_1], \ldots, [F_w])$.

Complexity Analysis. Computation complexities of shares generation, as well as multiplication for n providers, are equal to $O(n^2)$ each [14]. Secure minimum protocol requires $(\log_2 w)$ comparisons, which takes $O(n^2)$ time. Thus, the overall time complexity is equal to $O(n^2 \log_2 w) + \sum_{i=1}^{w} time\_complexity(\gamma_i)$. For our running example, the time complexity is equal to $O(n^2 + np_S)$, where p_S is the maximal number of fake values in the l-diversity protocol.

While running the above protocol, each data provider exchanges $w(n-1)$ messages for all multiplications. Secure minimum protocol is implemented using *lessThan* comparison subprotocol, and hence its communication complexity is equal to $O(n \log w)$ [14]. Therefore, the overall communication complexity is the sum of all such complexities and is equal to $O(wn^2) + \sum_{i=1}^{w} communication\_complexity(\gamma_i)$, which for our running example is equal to $O(n^2)$.

4.5 Experimental Evaluation

We run two sets of experiments for m-privacy w.r.t. C with the following goals: 1) to compare, and evaluate the different m-privacy verification algorithms, and 2) to evaluate,

and compare the proposed anonymization algorithm with the baseline algorithm in terms of both utility and efficiency.

4.5.1 Experiment Setup

We merged the training and testing sets of the Adult dataset[1]. Records with missing values have been removed. All remaining 45,222 records have been randomly distributed among n providers. As a sensitive attribute A_S we have chosen *Occupation* with 14 distinct values.

To implement SMC protocols, we have enhanced the SEPIA framework [14, 103], which utilizes Shamir's secret sharing scheme [81]. Security of communication is guaranteed by the SSL using 128-bit AES encryption scheme. For the secure l-diversity protocol we have used commutative Pohlig-Hellman encryption scheme with a 64-bit key [73].

Privacy Constraints. The EG monotonic privacy constraint is defined as a conjunction of k-anonymity [85] and l-diversity [60]. Privacy fitness score is defined by Equation 3.1. All algorithm parameters, and their default values are listed in the Table 4.1.

Table 4.1: Experiment settings and default values of SMC protocols.

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

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

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

4.5.2 Secure m-Privacy Verification

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

Attack Power. In this experiment, we compare m-privacy verification heuristics against different attack powers, and different number of data providers.

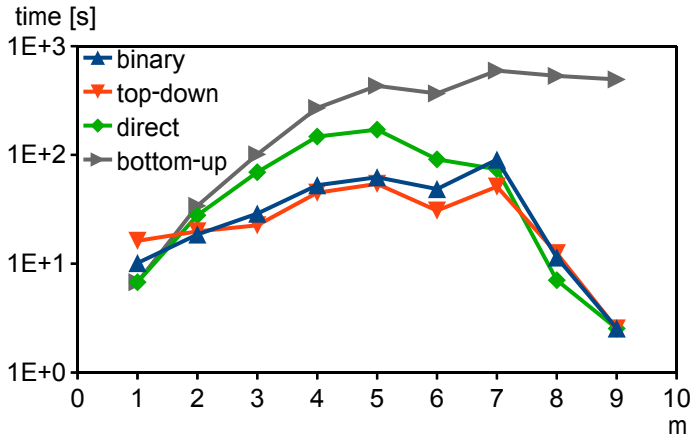


Figure 4.1: Computation time (logarithmic scale) vs. power of m-privacy.

Figure 4.1 shows computation time with varying m for all heuristics. Similar to the TTP implementation, the secure protocols for the *top-down* and *binary* algorithms demonstrate the best performance. The difference between these two approaches is negligible for most values of m. The *direct* approach is not that efficient as the above algorithms except small and large values of m. The *bottom-up* approach is useful only for very small values of m.

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

Number of Contributing Data Providers. In this experiment we analyze the impact of increasing number of data providers, n_G, on different algorithms.

Figure 4.2 shows the runtime of different heuristics with varying n_G.

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

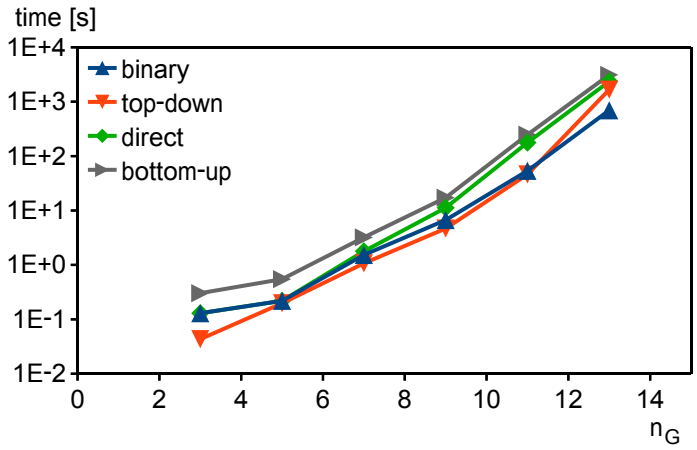


Figure 4.2: Computation time (logarithmic scale) vs. number of data providers.

slowest among others.

4.5.3 Secure m-Privacy Anonymization

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

As a comparison, we implemented an independent approach in which each provider anonymizes its data on its own. We observe that its runtime is independent of m and n, and equals to 1.2 seconds (not shown). However, the query error is significantly worse than for the collaborative setting (Section 3.5.3).

Attack Power. We first evaluate both anonymization heuristics with varying attack power m.

Figure 4.3 shows the estimated computation time with varying m for both approaches. As expected for EG monotonic constraints, increasing m results in stopping anonymization process significantly earlier. In addition, both approaches have comparable computation times with negligible differences.

Number of Data Providers. In this experiment, we estimate computation times for different number of data providers n, but with the same average number of records per

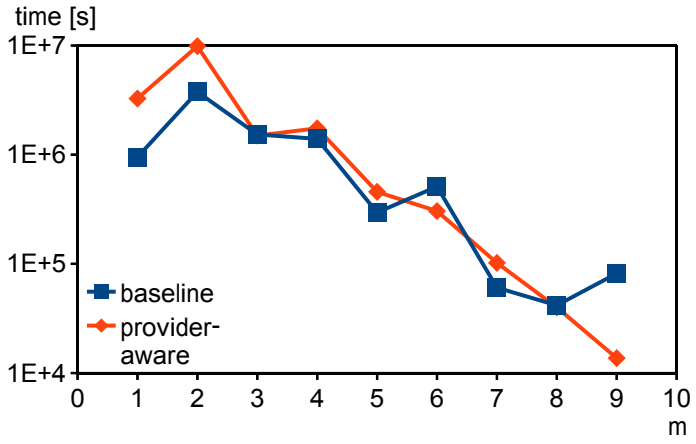


Figure 4.3: Computation time vs. power of m-privacy.

provider ($|T|/n = 100$).

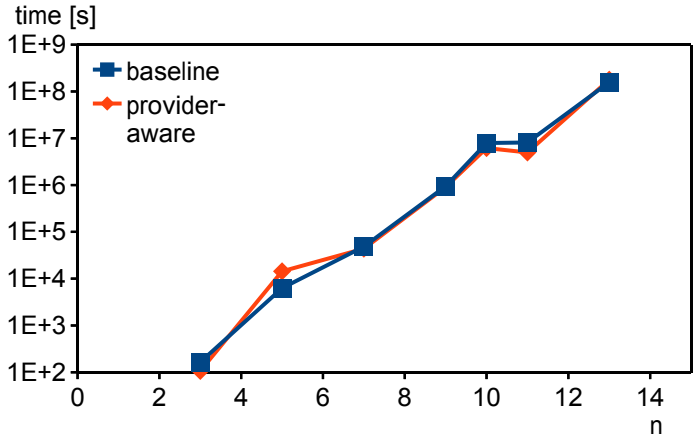


Figure 4.4: Computation time vs. number of data providers.

Figure 4.4 shows estimated time with varying the number of providers for both algorithms. As expected, the computation time is similar for both approaches, and increases exponentially with n.

Chapter 5

Distributed Data Aggregation with Customized Differential Privacy

While introducing m-privacy, we assumed that attackers corrupted a few records and they use that knowledge in their attacks. In the worst-case scenario all but one data providers would collude and each provider would have to achieve privacy of their records independently. However, even then attackers may learn if a record describing a *data subject* is in the dataset or not, which is already a valuable information. Therefore, privacy of data subjects is protected, only if a data recipient does not learn anything about them including their participation in the data collection.

5.1 Differential Privacy

Similar motivation inspired Dwork *et al.* to define *differential privacy*, which is the state-of-the-art semantic privacy model that gives a strong and provable privacy guarantee [27, 29, 30, 54]. Differential privacy assures an individual that her participation in the data collection can be guessed only with negligible probability regardless prior knowledge of an attacker. Informally, the attacker that knows all but one records and the result of computations, is not able to find out, if the remaining record has been used in computations or not. Formally, differential privacy is defined as follows.

Definition 5.1 ((α, δ)-Differential Privacy [28, 30, 54]). *A randomized mechanism $\mathcal{A}$ satisfies δ-approximate α-differential privacy, which is denoted also as (α, δ)-differential privacy, if for any neighboring databases D_1 and D_2, where D_1 can be obtained from D_2 by either adding or removing one record, and any possible output set S,*

$$Pr[\mathcal{A}(D_1) \in S] \leqslant e^{\alpha} \cdot Pr[\mathcal{A}(D_2) \in S] + \delta$$

If $\delta = 0$, then the mechanism satisfies α-differential privacy.

Dwork *et al.* introduced Laplace privacy mechanism that adds Laplace distributed noise to the final results and hence achieves differential privacy [28]. For results, which are integers, Ghosh *et al.* proposed a geometric mechanism, which integer noise is drawn from the two-sided symmetric geometric distribution [37].

5.1.1 Query Sensitivity

Let $D \in \mathbb{D}$ be a database, and Q be an aggregated query, e.g., a count query. In addition, let $\mathbb{R}$ be real numbers, $\mathbb{Z}$ be integer numbers, $\mathbb{N}$ be natural numbers, and $\mathcal{N}$ be a random variable (r.v.) representing noise.

All privacy mechanisms introduce perturbation to results of an aggregation query Q. Such perturbation is carefully calibrated with respect to the global sensitivity of Q, which is defined as follows.

Definition 5.2 (Global Sensitivity [31]). *The sensitivity of any aggregate function $Q : \mathbb{D} \rightarrow \mathbb{R}$, is equal to $\Delta_Q = \max_{D_1, D_2} ||\mathcal{A}_Q(D_1) - \mathcal{A}_Q(D_2)||_1$ for all D_1, D_2 differing in at most one record.*

5.1.2 m-Privacy and Differential Privacy

m-Privacy with respect to any syntactic privacy constraint C can be used to compute privacy-preserving microdata, i.e., to anonymize data such that its truthfulness is preserved. On the other hand, differential privacy preserves privacy of data owners in macrodata, i.e., in data statistics. At the same time it assumes the worst case scenario, i.e., all but one records

are compromised. A similar scenario for m-privacy is when each data provider contributes a single record and ($m = n - 1$), i.e., all but one data providers collude. Other than that both notions have no common properties.

5.2 Customized Privacy Budget

Differential privacy (DP) is often used as a notion of privacy by many data providers. The level of privacy preservation in DP is defined by a *privacy budget*. Each dataset has its privacy budget set to a certain value α_i. Every time a query with a specified differential privacy parameter α is issued against a dataset i, the answer is perturbed by adding noise that magnitude depends on α if ($\alpha \leqslant \alpha_i$), or is dropped (or drawn randomly), otherwise. In the former case, after the query is answered the effective budget is reduced by α, i.e., we *spent* α from the current privacy budget α_i to answer the query. If the α is not specified, we set $\alpha = \alpha_i$. If the query is issued against a few datasets, then we consider $\min(\alpha_i)$ as their overall privacy budget. The amount of budget that is spent to answer a statistical query over sensitive data describes the distribution of noise added to its result. The more budget is spent, the more accurate answer is returned.

Often queries are issued against a subset of sensitive data, hence the budget is spent only by the selected subset of data records. Thus, the distribution of the budget α_i over all data records depends on query workload and is not uniform (Table 5.1). In addition, data owners may independently customize privacy budgets of their records and set them based on their subjective judgements. Thus, we assign a privacy budget to each record independently from other records and any of its sensitive attribute values. Notice that the value of the privacy budget should not be correlated with any sensitive information, which would make it also sensitive. In our example (Table 5.1) a sensitive attribute Salary is not correlated with the privacy budget.

For both scenarios (multiple queries and personalized budgets) privacy budget of data records may vary. Treating all records as a single dataset limits budgets they can spend to the minimal budget among records, e.g., 0.01 for Table 5.1. This approach is not optimal, since a few records will still have a non-zero budget, which would not be spent. Spending

Table 5.1: Example records with different privacy budgets.

Name	**Age**	**Zip**	**Salary**	**Budget** α_i
Alice	22	02152	70000	0.01
Emily	32	02112	180000	0.02
John	31	02130	105000	0.05
Olga	27	02114	110000	0.07
Frank	36	02232	90000	0.09
Bob	35	01245	140000	0.11
Mark	33	04323	110000	0.14
Cecilia	39	02121	100000	0.15

all budgets would improve utility of computation results. On the other hand, considering each data record as an independent dataset would saturate budgets of all records. However, the total amount of noise added to results would reduce utility significantly.

Finding a strategy that generates the optimal buckets of data records is the main goal of this chapter. The optimal partitioning is defined as the one, which minimizes the difference between results generated from noisy and original datasets. We ensure privacy of data records by accessing them only through a differentially private interface. We implement such interface as a multidimensional DP-preserving histogram, and study different approaches of generating it.

5.3 Differentially Private Histograms

For given data records we want to build a multidimensional DP-preserving histogram, which will be as similar as possible to the optimal histogram without privacy guarantees, but with the same number of buckets. Notice that there are two sources of errors in histograms, approximation of records' distribution within a bucket and noise introduced to ensure differential privacy. If the number of buckets is a parameter, then increasing it would reduce approximation error, but at the same time it would increase the total amount of noise. Reducing the number of buckets would reduce the noise error, but it would also increase the approximation error. Hence finding the optimal number of buckets is a tradeoff between approximation and noise errors. We measure errors using variance of their distributions, therefore for a given number of buckets a histogram is optimal, if and only if it is v-optimal.

Definition 5.3 ([74]). *In a v-optimal histogram H, a weighted variance of the source values is minimized. That is, the quantity $V(H) = \sum_{j=1}^{k} n_j V_j$ is minimized, where n_j is the number of entries in the bucket j and V_j is the variance of the source values in the bucket j.*

The definition of v-optimality is very general, and can be applied to any error that can be characterize with variance. For DP-preserving histograms variance V_j of the bucket j is a sum of variances of approximation error V_j^A and noise V_j^N.

Notice that differential privacy for the bucket j is achieved by adding to its count noise drawn from the Laplace distribution with the mean 0 and the scale $1/\alpha_j$, where $\alpha_j = \min_i(\alpha_{j,i})$, and $\alpha_{j,i}$ is the privacy budget for a record i in the bucket j, and $x_j = \sum_i(x_{j,i})$. Therefore, $V_j^N = 2/\alpha_j^2$ and V_j^A is defined as the average *sum of squared errors* (SSE),

$$\begin{aligned} V_j^A &= \mathbb{E}((x_{j,i} - \mathbb{E}(x_{j,i}))^2) \\ &= \sum_{i=1}^{n_j} \frac{(x_{j,i} - \mathbb{E}(x_{j,i}))^2}{n_j} \\ &= SSE_j/n_j \end{aligned} \tag{5.1}$$

In [95], Xu *et al.* assumed that all records have the same privacy budgets ($\forall_{i,j}\alpha_{j,i} = \alpha$), and they defined the weighted variance as $\sum_{j=1}^{k} SSE_j$ for k buckets. For such settings authors proposed algorithms to build optimal DP-preserving single dimensional histograms. Inspired by their approach, we relaxed their assumptions by allowing multidimensional histograms, and customized privacy budgets. Such relaxation is necessary to perform complex tasks that produce privacy-preserving results.

Privacy Budget as an Attribute. For data records, the expected value $\mathbb{E}(x_{j,i})$ is defined as the average (Equation 5.1), i.e., $\mathbb{E}(x_{j,i}) = x_j/n_j$, where $x_j = \sum_i(x_{j,i})$. However, the expected value of privacy budget is defined by the minimum, i.e., $\mathbb{E}(\alpha_{j,i}) = \alpha_j$, where $\alpha_j = min_i(\alpha_{j,i})$. We apply both data and privacy budget to the definition of v-optimality (Definition 5.3), however to avoid confusion with the original definition, and to emphasize differences, we will use the notion of generalized v-optimality.

Notice that privacy budget and its definition of the expected value apply to all algorithms

of computing v-optimal histogram introduced in [47]. Authors used the average as the most common definition of the expected value, but they also mentioned other possible functions, including multidimensional ones.

Our goal is to build a generalized v-optimal multidimensional histogram H, which minimizes the variance of all errors $V(H)$, and is subject to privacy budget limitations. Notice that we do not set the number of buckets k, but keep it as an additional parameter. Since approximation error and noise variances reach maximum values for different k ($k = n$ and $k = 1$, respectively), then the minimal variance is reached for $k_{opt} \in \{1, \ldots, n\}$.

5.3.1 Data-driven Histograms

The straightforward solution (denoted as NO) is to consider privacy budgets as not important and use a heuristic to compute a histogram. In such approach the maximal privacy budget that can be spent is the minimal value of the budget among all records. If records have different privacy budgets, then they will not be able to spend all of it. A multidimensional histogram is generated by a greedy top-down partitioning algorithm (Algorithm 11), which partitions records until the variance of record buckets decreases. The algorithm was initially introduced by Cormode *et al.* in [22] as PSD. Such solution increases odds that privacy budget of each bucket will be low, i.e., a bucket will have a record with low privacy budget. As baselines we used a single cell/bucket histograms ($single\_cell$) and histograms with unit cells/buckets ($cells\_histogram$). Both of these baseline approaches are independent of the data.

5.3.2 Privacy- and Data- Driven Histograms

In order to consider privacy while creating histograms, we propose a new two-phase method of building histograms that take into account both privacy and data. In the first phase, we partition records based on their privacy budget values using different algorithms. In our approach (MIN) we modified the original v-optimal histogram building algorithm presented in [47], by using minimum as the expected value of the privacy budget (Algorithm 11). As baselines we use random partitioning (RND), the original algorithm from [47] (AVG), and we skipped partitioning (NO). In the second phase, we build histogram using methods

Algorithm 11: The PSD: a greedy heuristic of finding the k-histogram of $(x_1, \ldots, x_i)$, based on [22].

```
   Input: (x_1, ..., x_i) ∈ buckets
   Input: k
 1 foreach j ∈ {1, ..., k} do
 2     v = ∞
 3     B_1^min = null
 4     B_2^min = null
 5     foreach B ∈buckets do
 6         foreach attribute a do
 7             B_1, B_2 = splitMinVariance(B, a)
 8             currentVariance = variance(B_1, B_2)
 9             if currentVariance < min( variance(B), v ) then
10                 v = currentVariance
11                 B_1^min = B_1
12                 B_2^min = B_2
13         if v = ∞ then
14             return buckets
15         buckets = buckets \{B}
16         buckets = buckets ∪{ B_1^min, B_2^min }
17 return buckets
```

presented in the previous section (*single_cell*, *PSD*, *cells_histogram*). Combining different approaches for each phase we identify the best performing one. For example, all histograms that have been built with *single_cell* are privacy-driven histograms, while all histograms created by *NO* algorithm are data-driven.

Privacy-aware Data Partitioning. The goal of this phase is to group records with similar privacy budgets, such that the amount of wasted budget is minimized. In the extreme case, each partition has only one record, which is able to spend all its privacy budget. However, in this setting the amount of generated noise could be significant, which is not optimal. To spend as much budget as possible by each record and to avoid generating too many partitions, we introduced the *MIN* algorithm that partitions records based on their privacy budgets.

The remaining degree of freedom in the *MIN* algorithm is the order of records, while creating a histogram. Considering all possible combinations of n records among k buckets has too high complexity to be computed efficiently. However, since privacy budget of a

bucket is determined by the minimal budget record from that bucket, we can skip many combinations. Currently we sort all records in the ascending order of their privacy budgets. Notice that sorting records in the descending order of privacy budgets generates the same histograms.

Notice that in [47] authors used in their definition of the sum of squared errors (SSE) the average as a function describing the expected value. For privacy budget, the expected value is minimum, therefore for a bucket j, i.e.,

$$SSE_j = SSE(\alpha_{j,1}, \ldots, \alpha_{j,n_j}) = \sum_{i=1}^{n_j} (\alpha_{j,i} - \min_i(\alpha_{j,i}))^2 \tag{5.2}$$

With such definition of the SSE_j, the MIN algorithm is defined as follows.

Algorithm 12: The dynamic programming algorithm SSE^* of finding the optimal k-histogram of $(x_1, \ldots, x_i)$ for a given definition of the function SSE. Based on [47].

```
   Input: (x_1, ..., x_i)
   Input: k
1  if k = 0 then return 0
2  minSSE = ∞
3  foreach j ∈ {k-1, ..., i} do
4  |  currentSSE = SSE*((x_1, ..., x_j), k − 1) + SSE(x_{j+1}, ..., x_i)
5  |  if currentSSE < minSSE then
6  |  |  minSSE = currentSSE
7  |  |  buckets [k − 1] = (x_{j+1}, ..., x_i)
8  return minSSE
```

Comparing Partitioning Algorithms. Running the original AVG algorithm (with the SSE defined using the average value) to find the v-optimal histogram of privacy budgets with 3 buckets (A, B, and C) as shown in Figure 5.1.

With such partitioning 0.14 of privacy budget will not be spent (0.0175 on average). Running Algorithm 12 for the same data and also 3 buckets would produce buckets as show in Figure 5.1 With such partitioning only 0.12 of privacy budget will not be spent (0.015 on average).

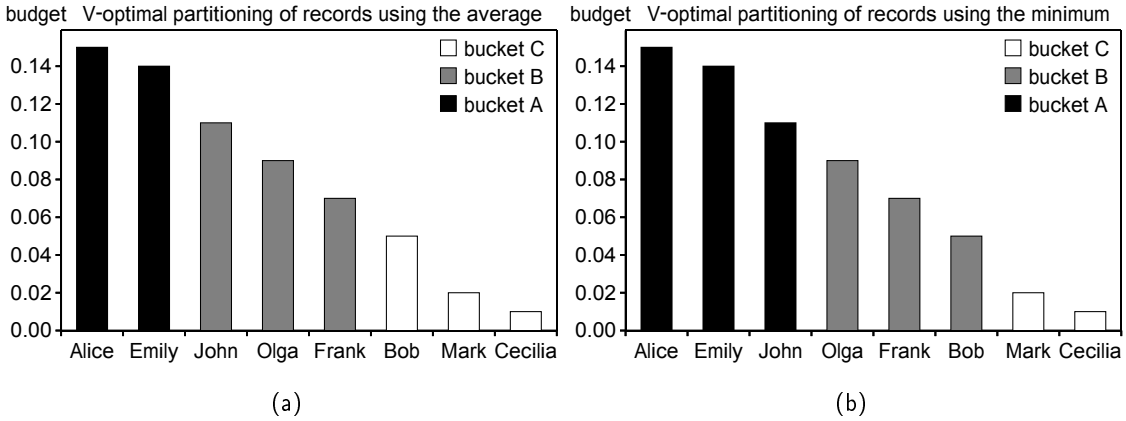


Figure 5.1: V-optimal partitioning of privacy budgets among 3 buckets using (a) the average and (b) the minimum as a target value of each bucket.

5.3.3 Strategies of Spending Privacy Budgets

Despite our efforts to minimize variance of privacy budget distribution, records in each bucket may have different privacy budgets. The privacy budget of a bucket is always equal to the minimum budget among all records. However, we utilize privacy budgets from all records using saturation or sampling. Such approach helps to improve quality of final results, which is very important if the variance of records distribution to the histogram is high.

Budget Saturation. The goal of the budget saturation is to create buckets with records, which privacy budgets are not greater than γ from their minimum, i.e., in bucket j $\forall_i \alpha_{j,i} < \min_k \alpha_{j,k} + \gamma$. Parameter γ controls granularity of budget partitioning. For γ equal to zero, the number of buckets would be equal to the number of distinct privacy budgets. If differences between budgets are small, a few buckets would have very small privacy budget, and would introduce the amount of noise, which would lower the overall utility of results.

We propose a new method of saturating privacy budgets by running Algorithm 13. Note that the function *copy* copies a record and sets for a new privacy budget for it. Given a bucket j we create an empty bucket j'. A record $x_{j,i}$ with privacy budgets equal or greater than $(\min_i(\alpha_{j,i}) + \gamma)$ is copied into a bucket j', with a new privacy budget $(\alpha_{j,i} - \min_i \alpha_{j,i})$. At the same time the budget of $x_{j,i}$ is reset to $\min_i \alpha_{j,i}$. After processing all records from the bucket j, the algorithm is run iteratively with new buckets, i.e., $j = j'$.

Algorithm 13: The *createBucket* algorithm of creating a saturated bucket.

Input: γ
Input: bucket j
$j' = \{\}$
$\alpha_j = \min_i(\alpha_{j,i})$
foreach $i = 1, \ldots, n_j$ **do**
    if $\alpha_{j,i} + \gamma \geqslant \alpha_j$ **then**
        $j' = j' \cup \{\ copy(x_{j,i},\ \alpha_{j,i} - \alpha_j)\ \}$
        $\alpha_{j,i} = \alpha_j$
return j'

Applying Algorithm 13 to our example dataset (Table 5.1) for $\gamma = 0.05$ and records sorted on privacy budget, we would group records into three buckets A, B, and C as it is shown in Figure 5.2.

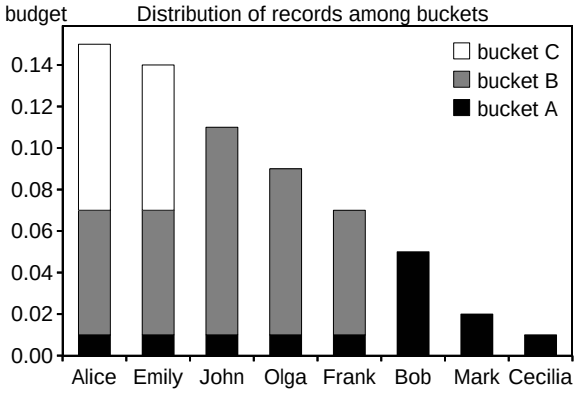


Figure 5.2: Saturation of the example record privacy budgets for buckets.

Note that each record from the second or the third buckets was copied from the first bucket. Sum of privacy budgets from all copies of a record is equal to its overall privacy budget.

Undersampling and Oversampling. Having a bucket of records with different privacy budgets we still want to utilize as budget as possible. At the same time, we want to drop records with too small budget, i.e., the budget that could noise the final result substantially. To meet these requirements we use undersampling and oversampling of records.

Records with a very small budget values are undersampled, i.e., dropped (or sampled with low probability). Oversampling helps to saturate more privacy budget than is assigned to the bucket. For example, a bucket has a privacy budget equal to α, and also it contains records

with higher budget. Thus, we generate a new bucket from records having a privacy budget greater than $\alpha + \epsilon$. Values of their budget in the new bucket is reduced by α. Repeating such procedure recursively, we ensure that each bucket has privacy budgets within ϵ length range. Note that to avoid buckets with low privacy budget, will limit necessity of record undersampling.

5.4 Experimental Evaluation

5.4.1 Settings

In our experiments we used a sample of 10,000 records from the Census dataset. Each record is described by three categorical attributes *martial status*, *sex*, and *salary*. To every record we have added one more numeric attribute *budget* with values equal to a privacy budget set to this record. Values of the *budget* have been drawn from normal, inversed exponential, and binomial distributions. All values that have been drawn outside of the budget values range, are set to the closest value in the range.

Privacy Budget Distributions. When using normal distribution $N(\mu, \sigma)$, we set its mean μ to the mean of all budget values, and standard deviation equal to $\sigma = 0.1$. Inversed exponential distribution $Exp(\lambda)$ for a range $[a, b]$ is a distribution $b - Exp(\lambda)$. We set $\lambda = \frac{2}{a+b}$. This distribution models population with majority of people that are willing to share their privacy with others, i.e., they set high privacy budgets for their records. The bimodal distribution is obtained from two normal distributions with different means. In our experiments we defined a range of all privacy budgets $[a, b]$ and normal distributions as $N(\mu_1, 0.1)$ and $N(\mu_2, 0.1)$, where $\mu_1 = a + \frac{b-a}{4}$, and $\mu_2 = a + \frac{3(b-a)}{4}$.

Metrics. To compare quality of generated histograms we issue $100,000$ random queries against each histogram and compute the average relative error, i.e., value 0.2 means that answers from the privacy-aware histogram will be, on average, 20% off from the real query answer.

5.4.2 Partitioning

To evaluate our partitioning method (MIN) we compare it against two baseline methods (NO, RND) and one non-privacy-aware method (AVG). The NO method does not partition records at all, i.e., all records are in a single partition. RND is a random partitioning, in which partition borders have been drawn randomly with uniform probability. AVG is a v-optimal partitioning method introduced in [47]. In this method records are partitioned based on their privacy budget values such that the variance of partitions is minimal. Notice that the expected value for every partition is defined as the average privacy budget among all records belonging to the partition. In our method MIN, we adapted the v-optimal partitioning algorithms with some modifications. The most important is setting the minimum privacy budget as the expected value for each partition.

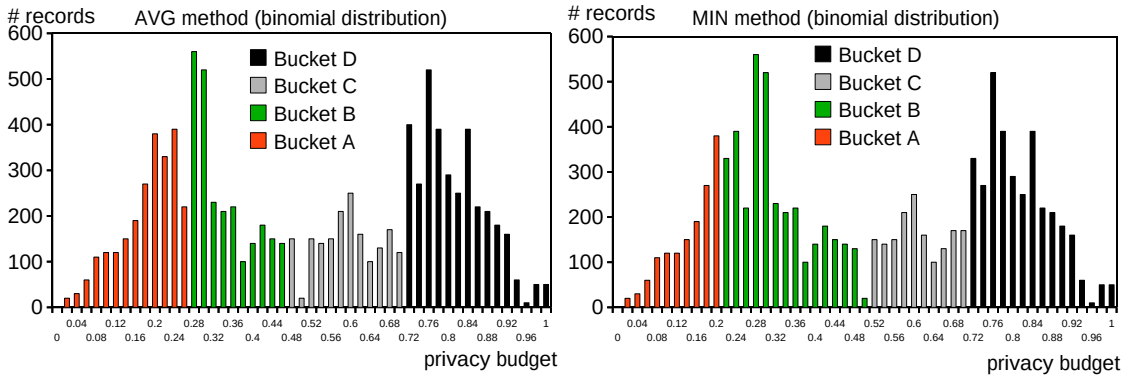


Figure 5.3: Partitions of records by AVG and MIN methods for binomially distributed privacy budgets.

Figure 5.3 shows partitioning by AVG and MIN methods for binomially distributed privacy budget. The first partition generated by the AVG method has more records than the first partition produced by our method MIN. Notice that the first partition contains the record wit the minimal privacy budget.

Figure 5.4 shows results of partitioning by AVG and MIN methods for inverse exponentially distributed privacy budget. Similar as for the binomial distribution, the first partition of the AVG method has more records than the first partition of the MIN method. Therefore, the AVG method partitions records in such way that more privacy budget (on average) cannot be spent in computations.

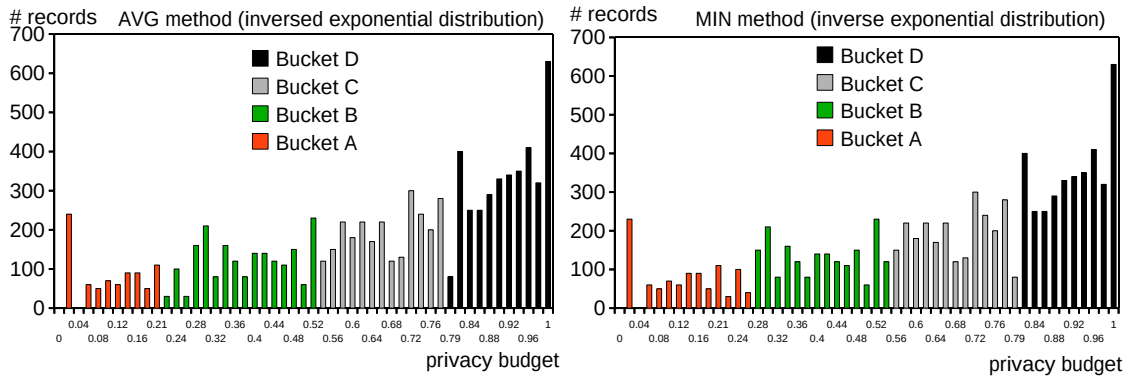


Figure 5.4: Partitions of records by AVG and MIN methods for inversed exponentially distributed privacy budgets.

5.4.3 Partitioning Methods

Responses to queries in privacy-preserving histograms are altered by approximation and perturbation errors. Approximation error is inherent to all histograms, while perturbation error depends on noise added to bucket counts. All three, distribution of privacy budgets as well as partitioning method, and a method used to build a histogram have impact to the overall error. In addition, the number of partitions k impacts overall error even more. For small k, the approximation error is the largest one, while for large k, perturbation is the main source of the error. Therefore, there is k for which the overall error is minimal, but finding it efficiently is not trivial. However, our experiments suggest that the number of buckets should be set around 3–4.

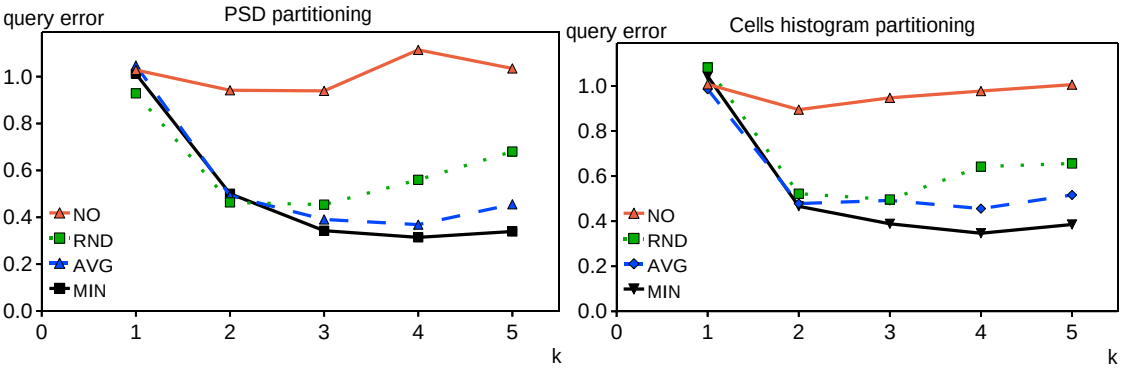


Figure 5.5: Query error for histograms built from different partitionings of records with binomially distributed privacy budgets vs. number of buckets k.

Figure 5.5 shows relative query error for histograms generated by different partitioning methods and for binomial distribution of privacy budgets. For all but NO methods we can find a local minimum of query errors for $k \in [2, 4]$. For greater k relative query error increases due to greater perturbation error. Among all methods, our approach (MIN) produces histograms with the smallest error for all methods of building histogram. We present only results for PSD and *cell histogram* approaches, and skip *one cell*, i.e., no partitioning approach. The AVG method generates partitions, which histograms answer queries with more error than MIN, but less than random partitioning RND. The number of buckets k, which generates the minimal error for MIN is equal to 4.

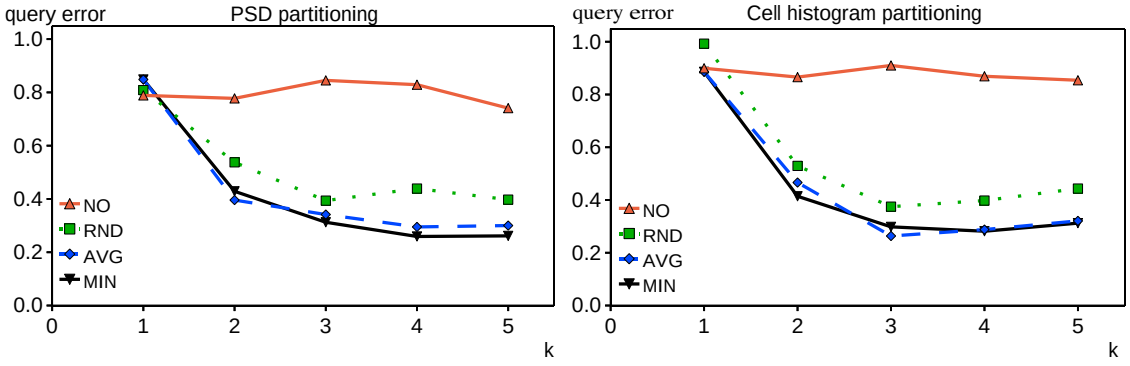


Figure 5.6: Query error for histograms built from different partitionings of records with budgets drawn from normal distribution vs. number of buckets k.

Figure 5.6 shows query error for histograms generated by different partitioning methods and for privacy budgets drawn from the normal distribution $N(0.5, 0.1)$. Similar to the binomially distributed settings (Figure 5.5) both AVG and MIN methods generates partitions for which histograms have the minimal error. However, results for MIN and AVG are more similar, and for $k = 2$ and the PSD method as well as for $k = 3$ and the cell histogram method the AVG partitioning generates histograms with lower query error. The number of buckets k, which generates the minimal error is equal to 4.

Figure 5.7 shows query error for histograms generated by different partitioning methods and for privacy budgets drawn from the inversed exponential distribution. Similar as for the binomial distribution, our MIN method generates histograms with the smallest error. In addition, the minimal error is achieved for k equal to 3 or 4.

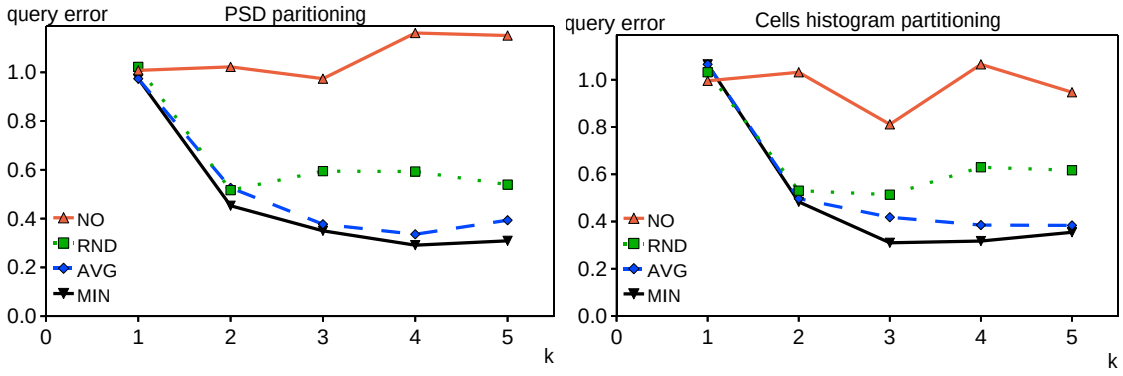


Figure 5.7: Query error for histograms built from different partitionings and for different number of buckets k for records with inversted exponentially distributed budgets.

Minimal Privacy Budget. For all previous experiments we drew privacy budgets from the range $[0.01, 1.0]$. In this set of experiments we change the range from which budgets are drawn. The distribution of budget values is binomial with maximums in 0.25 and 0.75 length of the range. Each range will have length equal to 0.5 and will start at different value.

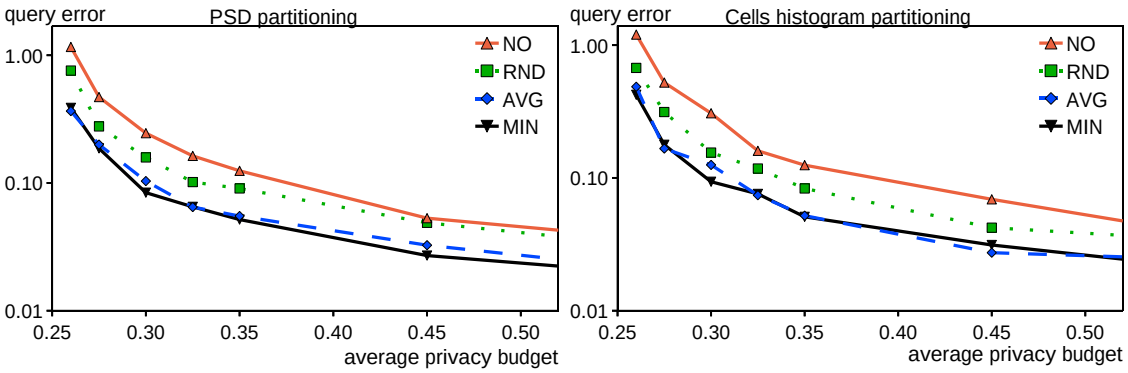


Figure 5.8: Query error (logarithmic scale) for histograms with records having different average privacy budgets, which were drawn from binomial distribution.

Figure 5.8 shows in logarithmic scales query error for histograms generated by different partitioning methods. For both *PSD* and *cells histogram* approaches *NO* and *RND* methods generate histograms with higher query error than *AVG* and *MIN*. Results for the latter ones are similar, but in most settings histograms produced based on *MIN* partitions preserve more utility, i.e., have lower query error.

5.4.4 Histogram Building Approaches

Each partition was used to generate a histogram using one of three methods. In the *single cell* method, a partition is treated as a one-cell histogram. In the *PSD* method we have adapted the method introduced in [21]. The *cells histogram* approach generates the most fine-grained histograms in which each bucket is a unite cube.

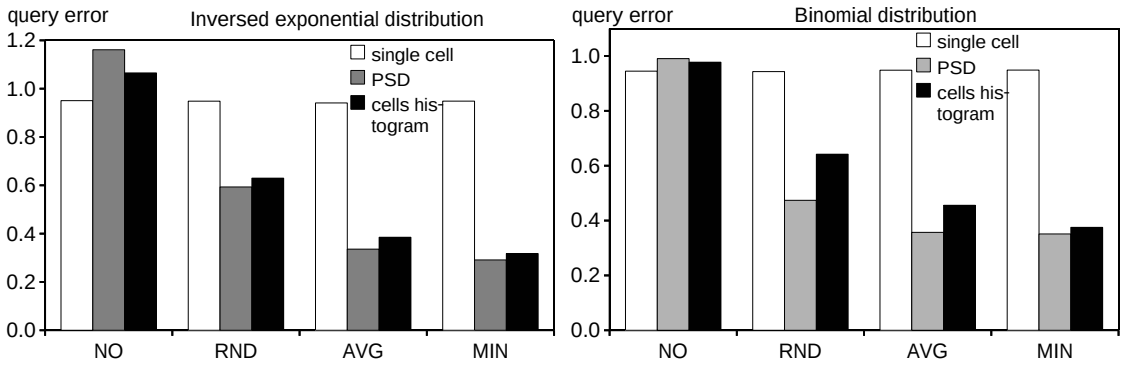


Figure 5.9: Query error for different methods of building histograms for four partitions and two different distributions of privacy budgets.

Figure 5.9 shows query error for different heuristics of generating histograms. Query error for single cell histograms are the highest. Cells histograms introduce histograms with less error than one cell histograms, but more than ones generated by the *PSD* heuristic. Notice that the difference between histograms built by *PSD* and *cells histogram* methods introduce similar amount of error. It is caused by relatively dense distribution of points in cells. For high dimensional histograms, the error of cell histograms would be even higher comparing to histograms generated by the *PSD* method.

Chapter 6

Secure Multiparty Data Aggregation with Customized Differential Privacy

6.1 Motivation

Participatory sensing and data surveillance [11, 35] are gradually integrated into an inseparable part of our society. In many applications, a data aggregator may wish to collect personal data from multiple individuals to study patterns or statistics over a population. *Data privacy and security* issues arise frequently and increasingly in such surveillance systems [52, 68, 77, 84]. An important challenge is how to protect the privacy of the data subjects, when the data aggregator is untrusted or not present.

System Settings. We consider a dynamic set of *data contributors* that contribute their own data (self surveillance) or other data (third party surveillance) in a surveillance system. In our running example contributors D_i $(1 \leqslant i \leqslant n)$ collect data x_i independently, in order to compute noisy $f(x_1, \ldots, x_n)$ (Figure 6.1). In the self surveillance scenarios the individuals represented in the collected data (*data subjects*) are also data contributors. We assume that collected data are used by an untrusted application or an application run by an untrusted party for analysis and modeling (e.g. disease outbreak detection or intelligence analysis).

Privacy of data subjects is defined by *differential privacy*, which is the state-of-the-art privacy notion [27, 29, 54] that assures a strong and provable privacy guarantee for

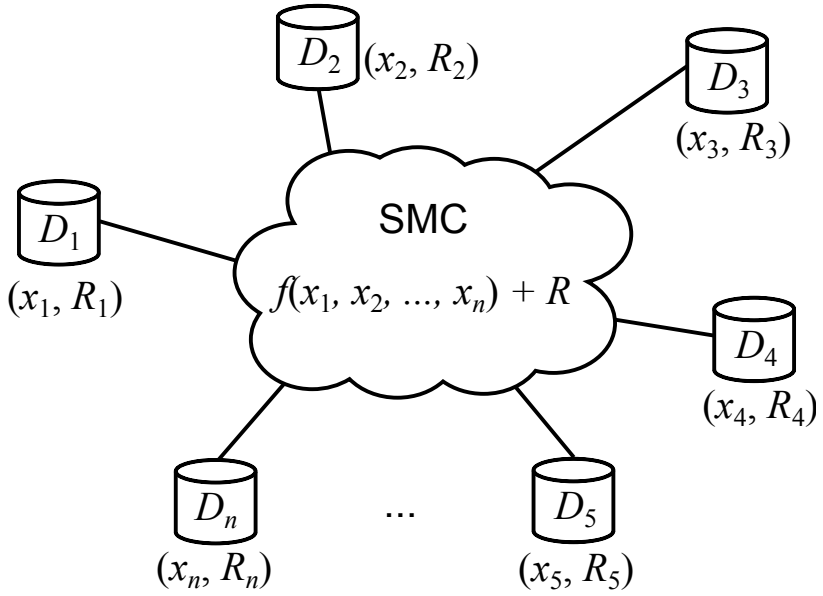


Figure 6.1: System settings with distributed data contributors D_i, which contribute their values x_i and noise shares R_i to securely compute a function f and ensure differential privacy of data subjects.

aggregated data. To use it we assume independence of data subjects, i.e., deleting one subject's data is equivalent to hiding all evidences of her participation in the dataset. Furthermore, we assume that no deterministic statistics about the participating data subjects have been previously released [54]. Under such assumptions differential privacy requires ensuring negligible change of computation results, when a single data subject had opted out of the data collection. Therefore, this assures an individual that any privacy breach will not unveil existence of its record. A common way of achieving differential privacy is perturbation of the aggregated statistics by calibrated noise R, such that $f(x_1, \ldots, x_n) + R$ is returned (Figure 6.1).

Centralized Model. In a *centralized model* a *trusted aggregator (TA)*, which is a TTP, (e.g. CDC offices in the syndromic surveillance scenario) securely collects the data and outputs perturbed aggregates with privacy guarantee. In such model, each data contributor maintains a secure communication channel with the TA. Both security and privacy of computations are guaranteed by the TA, but at cost of making it a single point of failure for the entire system.

Decentralized Model. In a *decentralized model* without a TA (e.g. in the intelligence collec-

tion scenario), the data contributors perform aggregations and perturbations collaboratively through protocols implemented in a secure multiparty computation (SMC) *schemes* (Figure 6.1). Such protocols are reliable, but are also complex to design and run. In order to securely aggregate information that contains personal data without involving the TA, each protocol must fulfill two important constraints: 1) preserve privacy of individuals or *data subjects* whose data are being collected, and 2) ensure security of all data contributors who should be anonymous and who need to protect their data from other contributors and any untrusted aggregator. Notice that using privacy mechanisms to ensure both constraints would make the final results useless. Using only security schemes to protect both computations and data contributors would not guarantee privacy of data subjects either. Therefore, privacy mechanisms and security schemes need to be employed together in our scenarios.

Goals. In the simplest and the most naïve approach, each data contributor perturbs its own data to guarantee their differential privacy independently. In such approach the noise in the aggregated result is superfluous, therefore our goal is to find distributed privacy mechanisms, which ensure privacy and minimize any redundant noise.

Distributed implementations of such mechanisms require security of computations that is achieved by SMC schemes and their protocols. The protocol shall remain secure if a few participants faulted, i.e., it should be fault tolerant. Therefore, SMC protocols that return differentially private results guarantee that privacy of data subjects and data providers is protected at all time.

Traditional approaches to data anonymization, such as removing identifying attributes, generalizing, or perturbing individual attribute values, are susceptible to various attacks [33]. They preserve truthfulness of data, i.e., if needed an anonymized record can be linked with its original record, but the privacy level they guarantee depends on background knowledge of attackers. In addition, secure anonymization of distributed data is complex and in the worst-case scenario every approach will be inefficient. However, in many scenarios getting anonymized data records is superfluous and their statistics are enough, e.g., the number of records.

6.2 Distributed Differential Privacy Mechanisms

Differential privacy can be achieved by a few privacy mechanisms. An example of non-perturbation mechanism that also achieves differential privacy is the exponential mechanism [63], in which privacy is achieved by sampling a dataset. In [43], we studied various distributed perturbation-based mechanisms that ensure differential privacy. In addition, we identified important features of such mechanisms, e.g., small redundant noise, the same level of contribution by each provider in noise generation, and drawing noise from specified domain (integer or real numbers).

In distributed settings any group of data contributors is equally likely to collude, therefore a mechanism shall ensure equal participation of each contributor in both computations and perturbations in order to minimize power of such group in its potential attacks. Equal participation in perturbing results means that all contributors generate the same level of noise, i.e., all noise shares are independent and identically distributed (i.i.d.) random variables.

In the simplest and the most naïve distributed approach, each data contributor perturbs statistics of its own data to achieve differential privacy for them. In such approach some noise in the aggregated result is redundant. Minimizing such redundant noise has been a motivation to find more efficient privacy mechanism.

6.2.1 Distributed Laplace Mechanism

Laplace Perturbation Algorithm. A common mechanism to achieve α-differential privacy is the Laplace perturbation algorithm (LPA) [29]. LPA ensures that results of an aggregated query Q are α-differentially private, by perturbing them with a r.v. $\mathcal{L}$, which is drawn from the Laplace probability distribution function $Lap(\theta, \alpha)$ with mean θ and scale α. The LPA mechanism is defined as follows.

Definition 6.1 (Laplace Mechanism, LPA [31],[29]). *For $Q : \mathbb{D} \rightarrow \mathbb{R}^k$, the mechanism $\mathcal{K}_Q$ that adds independently generated noise $\mathcal{L}$ with distribution $Lap(0, \Delta_Q/\alpha)$ to each of the k*

output terms enjoys α-differential privacy,

$$Pr(\mathcal{L} = x) = \frac{\alpha}{2\Delta_Q} e^{-|x|\alpha/\Delta_Q} \tag{6.1}$$

Notice that $\mathcal{L}$ can be also simulated by two exponential distributed random variables as follows.

Lemma 6.2 ([55]). *Let $\mathcal{X}_\lambda$ and $\mathcal{Y}_\lambda$ be random variables with the exponential distribution, i.e., $Pr(\mathcal{X}_\lambda = x) = Pr(\mathcal{Y}_\lambda = x) = \lambda e^{-\lambda x}$ for $x \geqslant 0$. Noise $\mathcal{L}$ in the LPA with the query sensitivity Δ_Q and parameter α (Definition 6.1) can be simulated as $\mathcal{L} = \frac{\Delta_Q}{\alpha}(\mathcal{X}_1 - \mathcal{Y}_1) = \mathcal{X}_\lambda - \mathcal{Y}_\lambda$ for $\lambda = \frac{\alpha}{\Delta_Q}$.*

Laplace mechanism can be directly applied by the TA in a centralized model. In a decentralized model the TA is not present and has to be simulated by data contributors. In such model differential privacy is achieved by distributed perturbation Laplace algorithms (DLPA). In DLPA each party generates partial noise, such that the aggregated noise follows the Laplace distribution, which is enough to guarantee differential privacy.

Due to infinite divisibility of Laplace distribution [55], a r.v. with such distribution can be computed by summing up n other random variables. We utilize this property and study a few algorithms named after distributions they use to draw partial noise, i.e., Gamma, Gauss, and Laplace. Gamma and Gauss have been already introduced in [1, 76], respectively, while Laplace is their alternative. We describe and compare all three DLPA mechanisms below.

Gamma DLPA. This mechanism utilizes infinite divisibility of Laplace distribution and generates partial noise by drawing it from the gamma distribution [1]. Formally, a Laplace distributed random variable $\mathcal{L} \sim Lap(\theta, \alpha)$ can be simulated by the sum of $2n$ random variables as follows,

$$\mathcal{L} = \frac{\theta}{n} + \sum_{i=1}^{n} (\mathcal{G}_i - \mathcal{H}_i) \tag{6.2}$$

where $\mathcal{G}_i$ and $\mathcal{H}_i$ are i.i.d. gamma distributed random variables with densities following the formula ($x \geqslant 0$),

$$Pr(\mathcal{G}_i = x) = Pr(\mathcal{H}_i = x) = \frac{(1/\alpha)^{1/n}}{\Gamma(1/n)} x^{1/n-1} e^{-x/\alpha} \tag{6.3}$$

Γ is the gamma function, such that $\Gamma(a) = \int_0^\infty x^{a-1}e^{-x}\mathrm{d}x$.

Notice that generating a r.v. with the gamma distribution requires drawing at least 2 random variables [3]. Assuming uniform distribution of its parameter a, on average 2.54 uniformly distributed random variables need to be drawn.

Gauss DLPA. A random variable $\mathcal{L} \sim Lap(0, \alpha)$ can be also simulated by 4 i.i.d. random variables $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4$, each is drawn from the normal distribution $Gauss(0, \alpha/2)$ with variance $(\alpha/2)$ [55] and applied as follows,

$$\mathcal{L} = \mathcal{N}_1^2 + \mathcal{N}_2^2 - \mathcal{N}_3^2 - \mathcal{N}_4^2 \tag{6.4}$$

Since infinite divisibility of the normal distribution is stable, drawing a single r.v. $\mathcal{N}_i \sim Gauss(0, \alpha/2)$ $(i = 1, 2, 3, 4)$ is simulated by the sum of n i.i.d. Gaussian random variables $\mathcal{N}_{i,j} \sim Gauss(0, \frac{\alpha}{2n})$ as follows,

$$\mathcal{N}_i = \sum_{j=1}^{n} \mathcal{N}_{i,j} \tag{6.5}$$

The above formulas can be implemented in distributed settings, but secure computation of squares of random variables is a challenge. An SMC protocol implementing such computations has high complexity [76].

Generating a random variable from the Gaussian distribution requires, on average, drawing a single uniformly distributed number [8]. Additional approaches, which are, on average, more efficient, but in the worst-case scenario are slow, have been described in [61].

Laplace DLPA. We propose a Laplace distributed differential privacy mechanism, which reduces the amount of redundant noise in the final solution [43]. The mechanism simulates drawing a Laplacian random variable $\mathcal{L} \sim Lap(0, \alpha)$ by r i.i.d. random variables $\mathcal{L}_i \sim Lap(0, \alpha)$ and a $\mathcal{B} \sim Beta(1, r-1)$, as defined in the following formula,

$$\mathcal{L} = \sqrt{\mathcal{B}} \cdot \sum_{i=1}^{r} \mathcal{L}_i \tag{6.6}$$

Notice that $\mathcal{B}$ is a single random variable, which is drawn with probability $Pr(\mathcal{B} = x) =$

$(r-1)(1-x)^{r-2}$ for $x \in (0,1)$. Assuming that at most m out of n data providers can be inactive the mechanism shall be run with $r = n-m$, i.e., aggregated partial noise from $n-m$ providers shall be enough to ensure differential privacy. Partial noise generated by additional providers is redundant, but very small comparing to other distributed differentially private mechanisms that were studied.

6.2.2 Geometric Mechanisms

The geometric mechanism presented by Ghosh *et al.* is centralized, i.e., noise is generated as a single random variable.

The LPA (Definition 6.1) perturbs query results with noise drawn from the continuous Laplace distribution. When returned results are expected to be integers, such mechanism cannot be directly applied. To address this, Ghosh *et al.* introduced two privacy mechanisms that ensure α-differential privacy by perturbing their results with integer noise and truncating noisy value if needed [37]. Their mechanisms work only for queries with sensitivity equal to one (e.g. count queries). We leave for future study finding a discrete perturbation mechanism for different sensitivities.

Definition 6.3 (Geometric Mechanism, GPA [37]). *For a function $Q : \mathbb{D} \to \mathbb{N}$, a parameter $\epsilon \in (0,1)$, the ϵ-geometric mechanism is an oblivious mechanism with integer range $\mathbb{Z}$, defined as follows. When the true query result is $Q(D)$, the mechanism outputs $Q(D) + \mathcal{N} \in \mathbb{Z}$, where $\mathcal{N}$ is a random variable drew from the following discrete $GM(\epsilon)$ distribution,*

$$Pr(\mathcal{N} = x) = \frac{1-\epsilon}{1+\epsilon}\epsilon^{|x|} \tag{6.7}$$

If the result of the query Q is limited to natural numbers $\mathbb{N}_m = \{0, 1, \ldots, m\}$, i.e., $Q : \mathbb{D} \to \mathbb{N}_m$, the perturbed output of the GPA is outside $\mathbb{N}_m$ with non-zero probability. The *truncated geometric mechanism (tGPA)* that has range $\mathbb{N}_m$ avoids such inconsistencies by "remaping" all negative outputs to 0 and all outputs greater than m to m, which formally is defined as follows.

Definition 6.4 (Truncated Geometric Mechanism, tGPA [37]). *For an aggregated function $Q : \mathbb{D} \to \mathbb{N}$ and parameter value $\epsilon \in (0,1)$, the truncated ϵ-geometric mechanism has*

range $\mathbb{N}_n = \{0, 1, \ldots, n\}$*, when for each value* $Q(D)$ *noise* $\mathcal{N}$ *is drawn from the following distribution,*

$$Pr(\mathcal{N} = x) = \begin{cases} 0 & \text{if } x + Q(D) \notin \{0, 1, \ldots, n\} \\ \frac{1-\epsilon}{1+\epsilon}\epsilon^{|x|} & \text{if } x + Q(D) \in \{1, \ldots, n-1\} \\ \frac{1}{1+\epsilon}\epsilon^{|x|} & \text{if } x + Q(D) \in \{0, n\} \end{cases}$$

The GPA is a discretized version of the LPA, in which noise is drawn from the $Lap(\Delta_Q/\alpha)$ distribution ($\epsilon = \exp(-\alpha/\Delta_Q)$) and $\Delta_Q = 1$ for both geometric mechanisms. Ghosh *et al.* proved that both geometric mechanisms achieve α-differential privacy.

Distributed Geometric Mechanism. We proposed a distributed geometric mechanism, in which n contributors participate i.i.d. noise shares to simulate ϵ-geometric mechanism for $\epsilon \in (0, 1)$ and achieve α-differential privacy ($\alpha = -\Delta_Q \ln(\epsilon)$) [43].

In order to present the distributed GPA we prove that noise generated by the GPA can be simulated as a difference between two exponentially distributed random variables. Then, we show that such variables can be generated as sums of i.i.d. Pólya distributed random variables.

Lemma 6.5. *Let* $\mathcal{X}$ *and* $\mathcal{Y}$ *be geometrically distributed random variables with the probability of success equal to* $(1 - \epsilon)$*, i.e.,* $Pr(\mathcal{X} = x) = Pr(\mathcal{Y} = x) = \epsilon^x(1 - \epsilon)$*. Noise* $\mathcal{N}$ *in the* ϵ*-geometric mechanism (Definition 6.3) can be simulated as* $\mathcal{N} = \mathcal{X} - \mathcal{Y}$*.*

Proof. We show that both $\mathcal{N}$ and $(\mathcal{X} - \mathcal{Y})$ have the same distribution, i.e., they have the same moment-generating functions M, $M_{\mathcal{N}}(t) = M_{\mathcal{X}-\mathcal{Y}}(t)$ for all valid t.

$$M_{\mathcal{N}}(t) = \frac{(1-\epsilon)^2 e^t}{(e^t - \epsilon)(1 - \epsilon e^t)}, \text{ for } t < -ln(\epsilon) \tag{6.8}$$

Recall also that for any two i.i.d. random variables $\mathcal{S}$ and $\mathcal{T}$, $M_{\mathcal{S}+\mathcal{T}}(t) = M_{\mathcal{S}}(t) \cdot M_{\mathcal{T}}(t)$. In addition, for any constant a, $M_{a\mathcal{S}}(t) = M_{\mathcal{S}}(at)$ [50]. Notice that the moment-generation function for geometric distribution with the probability of success p is equal

to $p(1-(1-p)e^t)^{-1}$ for $t < -ln(p)$ [50].

$$\begin{aligned} M_{\mathcal{X}-\mathcal{Y}}(t) &= M_{\mathcal{X}}(t) \cdot M_{\mathcal{Y}}(-t) \\ &= \frac{(1-\epsilon)^2 e^t}{(e^t-\epsilon)(1-\epsilon e^t)}, \text{ for } t < -ln(\epsilon) \end{aligned} \tag{6.9}$$

Thus, $M_{\mathcal{N}}(t) = M_{\mathcal{X}-\mathcal{Y}}(t)$ and $\mathcal{N} = \mathcal{X} - \mathcal{Y}$. □

In order to define DGPA we recall a Pólya distribution. Then we prove that such distributed random variables can be used to generate i.i.d. noise shares, such that the final noise follows the $GM(\epsilon)$ distribution, and therefore the DGPA achieves α-differential privacy ($\alpha = -\ln(\epsilon)$).

Definition 6.6 (Pólya Distribution [50])**.** *Let $\mathcal{X}$ be a random variable following the discrete Pólya(r,p) distribution with parameters $r \in \mathbb{R}$ and $p \in (0,1)$. The probability distribution function is defined as follows,*

$$Pr(\mathcal{X} = x) = \binom{r+x-1}{r-1} p^x (1-p)^r$$

If $r \in \mathbb{N}$, then the Pólya distribution is known as a negative binomial distribution $NB(r,p)$. If $r = 1$, then it is known as a geometric distribution with probability of success equal to $(1-p)$.

Theorem 6.7. *Let n be a number of data contributors and $\mathcal{X}_i$, $\mathcal{Y}_i$ be i.i.d. random variables following the Pólya$(1/n, \epsilon)$ distribution for $i = 1, 2, \ldots, n$. A random variable $\mathcal{N}$ with the distribution $GM(\epsilon)$ (Definition 6.3) can be simulated by the following sum,*

$$\mathcal{N} = \sum_{i=1}^{n} (\mathcal{X}_i - \mathcal{Y}_i)$$

Proof. Both NB and Pólya distributions are infinitely divisible [50]. Therefore, the sum of i.i.d. random variables $\mathcal{X}_i \sim Pólya(r_i, \epsilon)$ follows the same distribution with parameters $r = \sum_{i=1}^{n} r_i$ and ϵ.

If $r_i = 1/n$, then both $\sum_{i=1}^{n} \mathcal{X}_i$ and $\sum_{i=1}^{n} \mathcal{Y}_i$ follow $Pólya(1, \epsilon)$ distribution, i.e., are geometrically distributed with probability of success equal to $(1-\epsilon)$. Therefore, by

Lemma 6.5 we conclude the proof. □

To draw a random variable $\mathcal{X}_i$ from $P\acute{o}lya(r, \epsilon)$ distribution we utilize the Poisson-Gamma mixture [50]. First, we randomly draw λ from the $Gamma(r, \epsilon/(1-\epsilon))$ distribution. Then, we draw $\mathcal{X}_i$ from the Poisson distribution with parameter λ.

The ϵ-geometric mechanism draws noise from the two-sided geometric distribution $GM(\epsilon)$, which can be simulated as a difference between two sums of n Pólya distributed random variables. Therefore, if $\mathcal{N} \sim GM(\epsilon)$ and $\mathcal{X}_i, \mathcal{Y}_i \sim P\acute{o}lya(1/n, \epsilon)$, then

$$\mathcal{N} = \sum_{i=1}^{n} (\mathcal{X}_i - \mathcal{Y}_i) \tag{6.10}$$

Notice that both GM and Pólya distributions are discrete, therefore the noise in the proposed geometric mechanism is an integer number as well.

6.2.3 Distributed Noise Approximation Mechanisms

Both LPA and GPA achieve δ-approximate α-differential privacy for all values of $\delta \geqslant 0$ (Definition 5.1). Using these mechanisms for relaxed settings ($\delta > 0$) is not optimal, i.e., disturbance of the final result is superfluous. However, they can be modified to introduce less noise and still ensure required level of approximated α-differential privacy. One group of modifications is to draw noise from approximated noise distribution.

In [30] authors presented two methods of achieving (α, δ)-differential privacy for any values of α and $\delta > 0$. One method utilizes a Poisson process to generate random variables that approximate exponentially distributed random variables. The other method ensures (α, δ)-differential privacy privacy by approximating Gaussian noise with binomially distributed random variables. In both methods random variables are drawn by tossing unbiased or biased coins, which has been implemented in distributed settings, but with significant communication and computation overheads [30]. Therefore, we skip these mechanisms in our study and consider only diluted mechanisms, which also achieve (α, δ)-differential privacy and are efficient.

6.2.4 Diluted Distributed Mechanisms

Diluted mechanisms (Definition 6.8) have been designed as mechanisms easily applicable in distributed settings. In a diluted distributed perturbation mechanism (dDPA), each partial noise $\mathcal{N}_i$ is drawn by each of n parties. All $\mathcal{N}_i$ are securely summed up and used as noise to ensure approximated differential privacy. Notice that dDPA required that the rate of non-colluding parties is at least equal to γ, which is a parameter of dDPA.

A diluted geometric mechanism (dGPA) has been introduced in [83], as a mechanism that achieves (α, δ)-differential privacy. We generalize dGPA to a privacy mechanism that uses a perturbation algorithm (PA) and ensures approximate differential privacy.

Definition 6.8 (Diluted Mechanism, dPA)**.** *Let PA be a data perturbation algorithm used by an α-differential privacy mechanism, a parameter $\alpha \in (0,1)$, and $\mathcal{X}_i$ $(i = 1, \ldots, n)$ be i.i.d. random variables drawn by PA for given α. A diluted mechanism (dPA) ensures (α, δ)-differential privacy, by adding up to n random variables $\mathcal{N}_i$ defined as follows,*

$$\mathcal{N}_i = \begin{cases} \mathcal{X}_i, \text{with probability } \beta \\ 0, \quad \text{with probability } 1-\beta \end{cases} \tag{6.11}$$

where $\beta(\delta, \gamma) = \min\left\{\frac{1}{\gamma n}\log_2(\frac{1}{\delta}), 1\right\}$, and γn is the minimal number of generated random variables $\mathcal{N}_i$.

In dGPA results are perturbed by at most n random variables $\mathcal{X}_i$, which are drawn from the discrete two-sided geometric distribution $GM(\exp(-\alpha/\Delta_Q))$ (Definition 6.3). Each r.v. $\mathcal{X}_i \sim GM(\exp(-\alpha/\Delta_Q))$ can be simulated by a difference of two geometrically distributed random variables (Lemma 6.5). Therefore, drawing $\mathcal{N}_i$ by this mechanism requires generation of at most three uniformly distributed random variables. The first r.v. is used to decide, if $\mathcal{X}_i$ should be drawn. The second r.v., which is drawn from the uniform distribution, establishes the sign of $\mathcal{X}_i$, and the third one, which is drawn from the geometric distribution, sets the value of $\mathcal{X}_i$. On average for n tries, the $\mathcal{X}_i$ will be drawn βn times, and the overall number of drawn random variables is equal to $(n + \frac{2}{\gamma}\log_2(\frac{1}{\delta}))$.

Any differentially private mechanism may be applied to the dPA to achieve approximated

differential privacy of perturbed data. For example, the diluted Laplace mechanism (dLPA) calls the LPA mechanism to generate $\mathcal{X}_i$ with the same $\beta(\delta, \gamma)$ probability (Definition 6.8). Proving that such diluted mechanism achieves approximated differential privacy requires repeating proofs from [83], with simple substitution of the noise generation function.

6.2.5 Comparison

All DLPA and DGPA mechanisms guarantee the same level of differential privacy, while diluted and noise approximation mechanisms guarantee approximated differential privacy. Drawing a noise share by each mechanism has different computation cost and noise shares have different statistical characteristics.

To compare complexities of noise generation, we consider the number of random variables that each party generates for each method. For the Laplace DLPA mechanism, each party generates a single random variable, and one party generates 2 random variables, i.e., $1 + 1/n$ random variables per party, which makes it the most efficient mechanism. The implementation of a gamma random number generator has an indeterministic number of steps, and requires generating at least 2 (on average 2.54) uniformly distributed random variables [3]. The Gauss mechanism requires each party to generate 4 different Gaussian distributed random variables.

Each diluted mechanism requires every party to generate at least one, and at most two (LPA) or three (GPA) uniformly distributed random variables. Let γ be the minimal rate of non-colluding contributing parties and δ be a parameter of approximated differential privacy. Then, on average each party of dDLPA generates $(1 + \frac{1}{\gamma}\log_2(\frac{1}{\delta}))$ random variables and each party of dDGPA generates $(1 + \frac{1}{\gamma}\log_2(\frac{1}{\delta}))$ random variables.

Redundant Noise. Distributed settings introduce additional challenges for privacy mechanisms. One of them is collusion of participants that take advantage of shared data in order to breach privacy. Since each provider knows its own data and also data from colluding parties, we need to guarantee that results computed using data provided by non-colluding parties achieve privacy. We assume that the rate of colluding parties in one group is less than γ, therefore partial noise generated by γn parties is enough to ensure required differential

privacy. Since all n parties generate partial noise, which is aggregated in the final result, noise from the remaining $(1-\gamma)n$ parties is redundant. Notice that all parties generate i.i.d. partial noise, therefore its mean and its variance are the same for each party.

If a privacy mechanism is implemented using a security scheme, then its parameters affect the value of γ and *vice versa* (Table 6.1). In order to protect privacy of data subjects no fewer than γn parties should be able to reconstruct (Shamir) or decrypt (Paillier) the final result, therefore $\gamma \leqslant s/n$. Similarly, since the maximal number of colluding parties is smaller than r for Ács and EFT schemes, noise of remaining parties should be able to guarantee privacy, therefore $\gamma \leqslant 1 - \frac{r-1}{n}$. For all schemes the value of γ limits fault tolerance levels. If there are less then γn active parties, then computations are terminated, because their results would not be differentially private. Thus, the fault tolerance levels for all protocols are always less than $(1-\gamma)n$.

6.3 Security Schemes

Privacy mechanisms are not designed to provide security of computations. They can be used to do so, but the amount of perturbation added by each of them would make the final result useless. Therefore, to ensure security of data aggregation secure multiparty computation (SMC) schemes and protocols are employed. An SMC protocol shall remain secure even if a few participants are inactive, i.e., it should be fault tolerant. On top of that the SMC protocol shall be efficient and consume as little resources as possible.

We study three groups of secure schemes: secret sharing, homomorphic encryption, and perturbation-based. Each group has its own advantages and disadvantages for different types of data aggregations. Inspired by two schemes we present a new hybrid security scheme.

6.3.1 Secret Sharing Schemes

Secret sharing schemes split a secret into multiple shares that are meaningless unless s of them are collected and the secret is reconstructed. Any set of fewer than s shares discloses nothing about the secret even for computationally unbounded adversary. The value of s defines also the minimal number of colluding parties necessary to breach the security of

computations. After generation, shares are distributed, such that each contributor receives a few (usually one) shares. They are then used in computations and combined to reveal the final result.

As an example of secret sharing schemes we present the Shamir scheme [81]. In this scheme shares of different secret numbers can be summed up to get shares of their sum, which can be then reconstructed. Other arithmetic and set operations are also possible and implemented [14, 103].

Shamir Scheme. Each participant implements and runs the same Shamir scheme protocol with outline defined as follows. A participant i computes n shares of its local secret value x_i, by randomly generating a polynomial w_i of order s, such that $w_i(0) = x_i$. For a set of selected, and publicly known non-zero distinct points $z_1, \ldots, z_n$, shares of x_i are computed as values of w_i in these points, i.e., $x_{i,j} = w_i(z_j)$ for $j = 1, \ldots, n$. Then, shares are distributed, such that from all participants $x_{i,j}$ are sent to party j. After exchanging all shares, party j computes a share of the result at z_j, e.g., sums up received shares $\sum_i x_{i,j}$. Finally, one party collects at least s shares of the result, interpolates them to compute a polynomial w, and reconstructs the result by computing $w(0)$.

Notice that Shamir scheme was introduced only for integer numbers. Adapting the scheme to floating point numbers requires implementing different arithmetic operation protocols [18] or encoding floating point numbers as integers, e.g., by multiplying them by the same power of 10.

Security of Shamir scheme is based on the randomness of generated shares and security of communication channels. Assuming that all communication channels are secure, Shamir scheme is information-theoretically secure, i.e., is secure against computationally unbounded attackers [6]. However, since efficient implementation of secure communication channels relies on encryption (e.g. a Secure Sockets Layer), the scheme is, in fact, computationally secure, i.e., an attacker that uses current computer technology and available resources will not be able to breach security.

Since disclosing any set of less than s shares does not reveal the secret, the scheme is immune to collusion of less than s parties. Fault of any party may require identifying its

shares by other parties and dropping them, if a faulted party was able to distribute less than s of its shares. If enough shares were distributed and parties faulted, the protocol is continued. If decryption requires participation of more parties than are active, the protocol is rerun.

Shamir scheme has low computation complexity, but since each participant sends at least $(n-1)$ shares to others, the amount of communication is relatively high and equal to $O(n^2)$. An example framework that implements secure operations of Shamir scheme is SEPIA [14, 103].

6.3.2 Perturbation-Based Protocols

Perturbation-based protocols are an efficient alternative for other protocols, but often they require certain topology of connections, e.g., a ring. The main idea is to perturb input data by adding some random noise, such that they become meaningless for any attacker, and then perform computations on the noisy data. This approach achieves security by obfuscating all intermediate results, but is vulnerable to collusion and is not fault tolerant. Fault of any party requires rerunning the protocol by remaining active parties.

As an example of perturbation-based protocols, we consider the secure sum protocol [20]. In this protocol all parties are connected in a ring topology. Each party generates random noise, which is added to its private input. The starting party, which is elected randomly, sends its obfuscated value to its successor that adds its own obfuscated value and passes it further. At the end, the starting party holds the sum of obfuscated values from all parties. In the next phase, each party removes its noise from the obfuscated sum, which, at the end, reveals the correct final sum.

Unfortunately, if two neighbors of the same party collude, they can easily discover both the obfuscated value (the first phase) and the random noise (the second phase) generated by the party. With such information they can compromise the value participated by the party. Many enhancements have been introduced to this protocol to increase its collusion resistance, e.g., shuffling party positions in the ring, and dividing computations into multiple rounds [20]. Although such enhancements may significantly increase communication complexity of protocols, they are still very efficient.

6.3.3 Homomorphic Encryption

An encryption scheme is homomorphic, if it allows computations to be carried out on ciphertext. Formally, for a given homomorphic encryption function E with respect to a function f, the encrypted result of f can be obtained by computing a function g over encrypted values of $x_1, \ldots, x_n$, i.e.,

$$E(f(x_1, \ldots, x_n)) \equiv g(E(x_1), \ldots, E(x_n)) \tag{6.12}$$

Homomorphism of an encryption scheme is very useful in distributed settings. An outline of computing a value of f over distributed input x_i is defined as follows. First, each party encrypts its local data x_i and sends $E(x_i)$ to a single party. Then, such party computes the function g on encrypted data. Due to homomorphism of the encryption scheme, the party gets the encrypted value of function f, which can be used for further computations or decryption.

Fully homomorphic schemes allow secure multiparty computation of any function f. The price for such flexibility in choosing f is high complexity of computations [26, 36]. However, a few partially homomorphic schemes are efficient enough to achieve both security and performance goals, e.g., multiplicatively homomorphic ElGamal [32] and RSA [78] schemes. Our choice of the encryption scheme is determined by the aggregation operation, which in our scenarios is addition. Examples of additively homomorphic schemes are Paillier [71], Ács [1], and Shi [83].

Paillier Scheme. The Paillier scheme is a probabilistic public-key encryption scheme [24, 46, 71], which works as follows. Initially, a single trusted third party (TTP) generates a pair of public and private keys. A party i encrypts its local value x_i using the encryption function E and the public key. Then, all encrypted values are collected by any participant or the TTP, which computes $g(E(x_1), \ldots, E(x_n))$. The result can be decrypted or used in further computations.

The original protocol has been enhanced to a threshold scheme, in which shares of a private key are distributed, and any s out of n shares are sufficient to decrypt the ciphertext. In such scheme the TTP is not necessary to decrypt the final result. Any s participants

can do so, by partially decrypting it using their shares of the private key, and combining computed results. Details of key generation, encryption and decryption algorithms can be found in [24, 46]. For settings where the TTP is not present, generation and distribution of public and private keys can be also securely performed, by running a separate SMC protocol [25, 70], e.g., Diffie-Hellman key exchange protocol [1].

Ács Scheme. The Ács scheme is a modulo addition-based encryption scheme [1], i.e., addition is the encryption function. Encryption keys are generated in pairs by two parties, e.g., by running Diffie-Hellman key exchange protocol. Keys in each pair are inverse to each other, i.e., they sum up to 0. Each party has r encryption keys, which inverses are held by r other parties. Since in the final result all encryption keys are summed up, they cancel out and no decryption is necessary.

Any group of less than r colluding parties is not able to disclose local value of any other party. However, bigger groups can do so with non-zero probability. In the original scheme introduced by Ács *et al.*, fault of any participant requires rerunning all computations from the beginning. The original scheme can be extended by a recovery subprotocol that finalizes computations and returns the correct result. We introduce and analyze such subprotocol as part of our enhanced scheme described below.

Establishing r keys for each party before actual computations is inefficient and requires running a key exchange protocol by $\frac{rn}{2}$ pairs of parties. Unfortunately, reusing the same keys leads to leaks in security especially when parties fault. The following scheme addresses this requirement of using different keys, by generating them from setup keys.

Shi Scheme. In the Shi scheme a separate decryption function is reduced to computing the discrete logarithm of the output in order to get the final result [83]. If aggregated value is used in further computations, decryption can be postponed until the final result is computed. Similar to the Ács scheme, the method of generating encryption keys ensures that the final result will be already decrypted. Although all parties need to participate to compute the result, they do not need to communicate in order to establish encryption keys for the next run of the protocol. For each run, an encryption key is computed from the initially established key using a one-way function, e.g., a hashing function SHA-256. Such

approach minimizes communication among parties, but requires additional computations.

Since participation of all parties is necessary to finalize computations, fault of any party during computations makes the result useless, and the protocol has to be rerun by active parties. Communication complexity is minimal, but computation complexity is high due to key generation.

6.3.4 Enhanced Fault Tolerant Scheme

We proposed an *enhanced fault tolerant* (*EFT*) security scheme, which is efficient, fault tolerant, and collusion resistant. The EFT scheme is an encryption scheme with encryption implemented by adding to or subtracting from the local data a uniformly distributed random variable, which is secretly agreed between two parties. Since each party exchanges such random variables with a few neighboring parties, encrypted value is secured unless all neighboring parties collude. The result of data aggregation does not require decryption since perturbation introduced by each party is canceled out by its inverse from a neighboring party. This feature is also used to recover the result after a few parties became inactive. Thus, the encryption function used in the EFT is very efficient and does not require any decryption function. In addition, to avoid reestablishing all encryption keys among neighboring parties in the beginning of every computation round, the keys are computed from the initially agreed secret keys and the publicly known current round number.

Each run of the protocol implemented in the EFT scheme is initiated by an untrusted party, which want to aggregate data provided by contributors. Such party collects encrypted local values and aggregates them. At the same time it detects if any party gets inactive. When at least one participant is not able to finish the protocol, the aggregator initiates the recovery protocol by informing all active parties about the inactive ones. To recover the result, parties prepare their recovery keys, which are aggregated with the previously computed result.

EFT Details. In our scheme an untrusted data aggregator, which is an independent entity or is simulated by any data contributor, initiates all protocol runs (Algorithm 14). After aggregating values returned by all parties N it returns the final result (lines 1 to 6). If any

data contributor faults, the aggregator will detect it (line 5). The result will be decrypted only partially and the aggregator will run the recovery subprotocol to compute the final result (lines 7 to 9).

Algorithm 14: The data aggregation and recovery procedures of the EFT scheme, which is run by an untrusted party.

```
1 sum = 0
2 Faulted = ∅
3 foreach j ∈ N do
4 |   sum += get_encrypted_value_from(j, timeout)
5 |   if timeout happened or no connection with j then Faulted = Faulted ∪ {j}
6 if Faulted = ∅ then return sum
  // Recovery subprotocol.
7 foreach j ∈ N do
8 |   sum += get_recovery_key_from(j, Faulted)
9 return sum
```

Data contributors participate in the aggregation by running Algorithm 15. In the setup phase, a contributor i establishes random keys $k_{i,j}$ with r randomly chosen parties N_i (lines 1 to 5). After fixing $k_{i,j}$ no further setup is needed, and no communication is generated. During encryption, each $k_{i,j}$ is hashed with the current timestamp t, and the result is added to, or subtracted from, the contributed value x_i (lines 8 to 9). The result is sent back to the aggregator.

Notice that only r collaborating neighbors N_i can breach security and reveal x_i. Therefore, when using a privacy mechanism with our scheme, the minimal number of noise shares required to ensure privacy shall be at most r.

In the recovery process (Algorithm 16), each party gets the set of all faulted parties (*Faulted*). For faulted neighbors the party computes its recovery key, which is the sum of inverses of aggregated encryption keys (lines 3 to 4), drops connection with them (line 5), and removes them from its set of neighbors N_i (line 6). If all neighbors of a party i faulted, then sending the recovery key to the aggregator would reveal the contributed value x_i. Therefore, before sending, the party subtracts x_i from the recovery key, which will remove it from the aggregated result.

Security. Security proofs of our scheme are the same as presented in [1]. Encrypted keys are

Algorithm 15: The encryption function run by a party i contributing x_i at time t with encryption keys exchanged with parties N_i of the EFT scheme.

```
if |N_i| < r then
    N'_i = connect_randomly_with_new_parties(r − |N_i|)
    foreach j ∈ N'_i do
        k_{i,j} = Diffie-Hellman_key_exchange(i, j)
    N_i = N_i ∪ N'_i
ciphertext = x_i
foreach j ∈ N_i do
    if id(i) > id(j) then ciphertext += Hash(k_{i,j}, t)
    else ciphertext −= Hash(k_{i,j}, t)
return ciphertext
```

Algorithm 16: The recovery protocol run by a party i contributing x_i at time t with neighbors N_i and *Faulted* parties failing of the EFT scheme.

```
recovery_key = 0
foreach j ∈ Faulted ∩ N_i do
    if id(i) < id(j) then recovery_key += Hash(k_{i,j}, t)
    else recovery_key −= Hash(k_{i,j}, t)
    disconnect_from(j)
    N_i = N_i \ {j}
if N_i = ∅ then recovery_key = recovery_key − x_i
return recovery_key
```

Scheme	**Communication complexity**	**Fault tolerance level**	**Max. collusion (max. $n-2$)**
Shamir (s,n)	$n(n+1)$	$n-s$	$s-1$
Perturbation-based (n)	$3n$	0	1
Paillier (s,n)	$5n$	$n-s$	$s-1$
Ásc (r,n)	$2n+2rn$	n	$r-1$
Shi (n)	$2n$	0	$n-2$
EFT (r,n)	$2n$	n	$r-1$

Table 6.1: Comparison of complexity, fault tolerance level, and max. allowed collusion for SMC schemes with n parties.

computationally secure due to the hash function, e.g., SHA-256, which cannot be reversed in a polynomial time. Notice that the EFT scheme is immune to collusion of less than r parties. Increasing $r \in [1, n)$ will increase security of the protocol, but will also reduce its scalability. The value of r should be established based on probability of faults and should be greater than the maximal number of colluding parties.

Complexity. If all parties are active and run our protocol, then they generate $2n$ messages in only two rounds — one to collect encrypted values by an untrusted aggregator and one to broadcast the final result. When a few parties faulted, a recovery process would generate additional $3n$ messages to broadcast *Faulted*, collect recovery keys, and broadcast the recovered result. Computational complexity of the EFT protocol is slightly higher than complexity of the Ács scheme, due to additional computations. However, since encryption keys are not reestablished before each computation, the communication overhead is significantly lower, while preserving fault-tolerance.

6.3.5 Comparison

To compare different schemes we have implemented a secure sum protocol in each of them and analyze their complexity and security characteristics.

A summary of the comparison is presented in Table 6.1. Shamir scheme does not require significant computational resources and its fault tolerance level depends on the number of parties required to reconstruct the secret s. However, the high communication complexity is a major drawback that limits its scalability. Additionally, all $\frac{n^2}{2}$ communication channels

need to be secure.

Each perturbation-based protocol is suited for a specific computation and requires participation of all parties. Thus, it is not fault tolerant and faults of any party requires rerunning it. In addition, in the presence of colluding parties, the scheme does not ensure security, which is its major weakness. However, such protocols are suitable for settings where parties have limited resources, but are reliable.

Among homomorphic encryptions schemes Paillier and Shi incur high computation overheads due to their heavy encryptions. Therefore, these schemes may be suitable for scenarios in which participants have more computational power, and high scalability is required. The minimal amount of communication is generated by Shi and our EFT scheme. Shi scheme is immune to $(n-2)$ colluding parties, but will not be able to recover after fault of any party. Paillier, Ács, and our schemes are fault tolerant with the level of protection against colluding parties defined as a parameter. However, after initial setup our scheme does not require any more communication, while in Ács scheme all encryption keys need to be regenerated, which causes exchanging $2rn$ additional messages. Reestablishing encryption keys before each computation (n messages) and decryption of results ($2n$ messages) increase significantly the communication complexity of Paillier scheme. Among encryption schemes, our EFT scheme is the most efficient in terms of communication and computations, is also fault tolerant, and is flexible in adjusting its collusion level. None of other schemes holds all these properties.

Fault Tolerance. All schemes can be partitioned by their fault tolerance level into three groups. Perturbation-based and Shi schemes are not fault tolerant, i.e., if any party faults, the currently run protocol is stopped and run again.

Schemes from the other group deal with faults silently (e.g. Shamir and Paillier), i.e., they do not run any recovery protocol to retrieve final results, but continue computations. However, if the number of active data contributors drops below s, then protocols implemented in either of these schemes are stopped and rerun with decreased value of s. Notice that s cannot be less then the maximal number of potential colluding data contributors, and the number of noise shares necessary to achieve differential privacy.

Remaining two schemes, i.e., Ács and our EFT scheme, finish computations and return results regardless of any faulted participants. However, when a data contributor faults, all remaining contributors shall run a recovery protocol, which we presents in Algorithm 16.

Collusion of Parties. Only Shi scheme remains secure after collusion of $(n-2)$ parties, which is the maximal number of colluding parties. Collusion of any two or more parties may breach security of Perturbation-based scheme. Rearranging topology of connections among parties and dividing computations into multiple stages may improve collusion resistance of this scheme to certain extent. For remaining schemes the maximal number of allowed colluding parties is less than the number of encryption key shares required to reconstruct the result (Shamir, Paillier) or the number of keys exchanged by each party with others (Ács, EFT).

6.4 Experimental Evaluation

In this section, we present experimental evaluations of various privacy mechanisms and security schemes used to implement a distributed secure sum protocol. Since the security and privacy levels of schemes have been formally analyzed above, we mainly focus on their performance. The questions we attempt to answer are: 1) How do the different distributed noise generation algorithms and privacy mechanisms compare with each other in terms of efficiency and redundant noise? 2) How do the different secure computation scheme protocols perform in various settings, and how do they scale, and compare with each other in terms of computation and communication cost?

6.4.1 Experiments Setup

All experiments have been run using JVM 1.6. We evaluated local computations including partial noise generation, and data preparation on three different platforms: 1) a *cluster* of 64 HP Z210 *nodes* with 2 quad-core CPUs, 8 GB of RAM each, running Linux Ubuntu system, 2) a *laptop* with Intel Core 2 Duo T5500 and 2 GB of memory running Windows XP, and 3) a shared *server* Sun Microsystems SunFire V880, with 8 CPUs and 16 GB of memory running SunOS 5.10. Notice that the sever assigns only limited amount of resources to our

applications. All protocols are evaluated in a distributed environment using the cluster of nodes, which are connected by the 100Mbit network. All reported results are averaged from $1,000$ runs for security scheme and $1,000,000$ tries for privacy mechanism experiments.

Our main software framework is built on top of SEPIA [14, 103], which uses Shamir secret sharing scheme for secure distributed computations. We extended SEPIA and implemented other SMC schemes and privacy mechanisms to achieve differential privacy of the final results. We chose implementation of the Paillier scheme from the UTD Paillier Threshold Encryption Toolbox [104]. Additionally, we used random number generators implemented in the HPC library Colt [100]. All remaining schemes and mechanisms have been implemented by authors. Default values of experiment parameters are listed in Table 6.2.

Name	**Description**	**Default Value**
n	Number of running nodes.	32
k	Size of encryption keys in bits.	128
r	The number of encryption keys exchanged with neighboring parties for Ács and EFT schemes.	3
s	The minimal number of parties required to decrypt or reconstruct results in a security scheme.	3
γn	The minimal number of noise shares to achieve privacy for privacy mechanism experiments, and the minimal number of non-colluding parties.	8
δ	A parameter of approximated differential privacy	0.1
	The key size (in bits) of the AES encryption with RSA for SSL communication channels.	128

Table 6.2: Default values of experiment parameters.

6.4.2 Privacy

The main goal of this experiment is to evaluate the overhead of the following mechanisms (Section 6.2):

- distributed LPA (DLPA): Laplace, Gamma, and Gauss,
- distributed GPA (DGPA),
- diluted: Laplace (dLPA), geometric (dGPA).

DLPA and DGPA guarantee differential privacy of the final result, while diluted mechanisms ensure approximated differential privacy. For all DLPA mechanisms the final result achieves the same level of differential privacy, i.e., its final noise is drawn from the same distribution. Therefore, we compare the local computation time of the three DLPA and the DGPA geometric mechanisms, as well as their impact on the overall protocol performance.

Noise Share Generation. In order to ensure that enough noise is added to the final result, each node adds its share of noise. The average generation time of such shares for different mechanisms is shown in Figure 6.2.

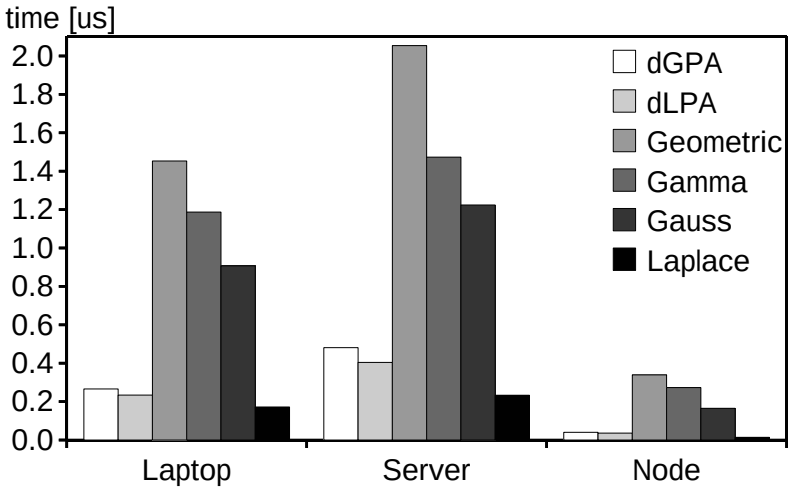


Figure 6.2: The average noise share generation times in microseconds for different mechanisms and platforms.

Generating a single noise share by the Laplace DLPA is more efficient than by other mechanisms, which confirms our expectations (Section 6.2.5). Drawing a uniformly distributed r.v. and a few arithmetic operations are enough to generate a noise share in Laplace DLPA mechanism with efficiency. Notice that Laplace DLPA requires also a r.v. drawn from the beta distribution, which is generated and broadcasted to all parties as part of the setup message for each run, therefore it is not considered here. Geometric mechanism requires drawing two random variables, one from Poisson and one from gamma distributions, which makes it slower than most DLPA mechanisms. Gauss requires generating 4 normally distributed random variables, while gamma, on average, requires slightly over 5 uniformly distributed random variables [3].

Efficiency of diluted privacy mechanisms with default parameter values (Table 6.2), which achieve δ-approximate α-differential privacy, is very high. Performance of noise generation for diluted mechanisms depends on $\beta(\delta, \gamma, n)$, i.e., probability of generating noise (Definition 6.8), which, for default values of parameters, is approximately equal to 41.52%.

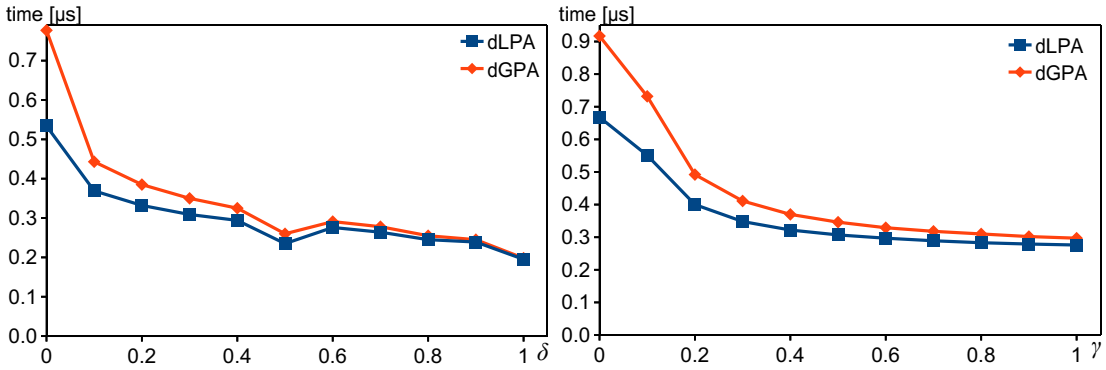


Figure 6.3: The average noise share generation times in microseconds for different δ and γ, run on the *server*.

Figure 6.3 shows the average runtimes of noise generations for dLPA and dGPA with different δ and γ on the *server*. As expected, relaxing the approximate differential privacy constraint (Definition 5.1), i.e., increasing the value of δ, decreases both the probability of noise generation β and the runtime. Similarly, increasing the fraction of non-colluding parties γ also decreases β and the runtime. Since generating Laplace noise is more efficient than generating geometric noise, the dLPA is also more efficient than dGPA, which is confirmed in our experiments.

Redundant Noise. To protect privacy of data subjects against colluding data providers, we run privacy mechanisms requesting that shares of γn participants ($\gamma n = 10$) are enough to achieve privacy. Thus, our final results have some additional noise, which characteristics are different for each mechanism. In this experiment we compare redundant noise generated by all privacy mechanisms (Figure 6.4).

Laplace, Gamma, and Geometric mechanisms generate similar amount of redundant noise. Among them the Laplace mechanism generates slightly less noise than Gamma and Geometric mechanisms for $\gamma \leqslant 0.6$ and slightly more for $\gamma > 0.6$. All three mechanisms generate significantly less redundant noise than the Gauss mechanism for any γ and α.

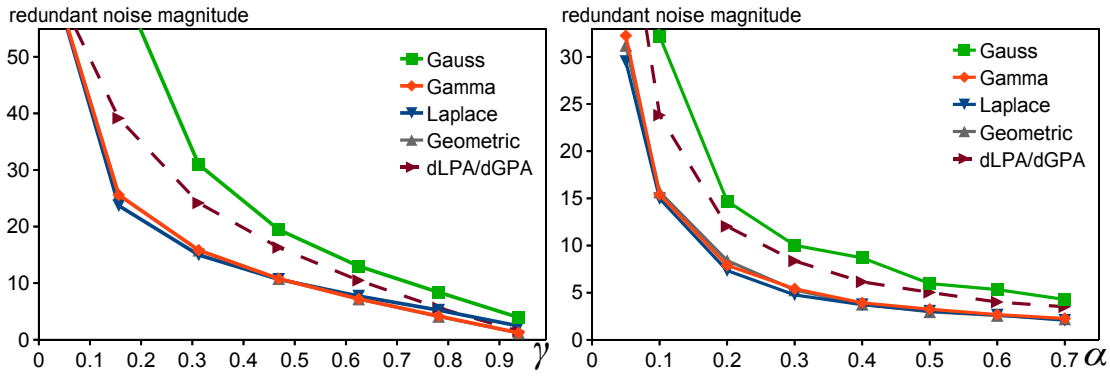


Figure 6.4: The average magnitude of redundant noise for different rate of required noise shares γ ($\alpha = 0.1$), and different privacy budgets α ($\gamma = 10/32$).

For given settings redundant noise magnitudes of dLPA and dGPA are almost the same, therefore we represent them as a single dashed line in Figure 6.4. DLPA and DGPA mechanisms cannot be compared with diluted mechanisms, in which redundant noise depends on $\beta(\gamma, \delta)$ (Definition 5.1). However, we can compare characteristics of their redundant noise. Since β is independent of α, redundant noise for diluted and other mechanisms will decrease at the same rate as α is increasing. In diluted mechanisms γ impacts redundant noise differently than other mechanisms. Requiring participation of more non-colluding parties to achieve privacy (increasing γ) decreases redundant noise for diluted mechanisms slower than for DLPA (except Gauss) and DGPA.

6.4.3 Security

In this group of experiments we evaluate performance of distributed aggregation protocols for different security schemes. Security levels guaranteed by each scheme have been already discussed (Section 6.3).

Performance of Homomorphic Encryption Schemes. In this set of experiments we evaluate homomorphic encryption schemes. In the setup phase encryption keys are generated and distributed. To ensure maximal security of the Paillier and the Ács schemes in each round new keys are used. If we lower security requirements, the same encryption key could be reused a few times. Therefore, we set up the Paillier scheme to be run in two settings named *new key* (maximal security) and *reuse key* (lower security).

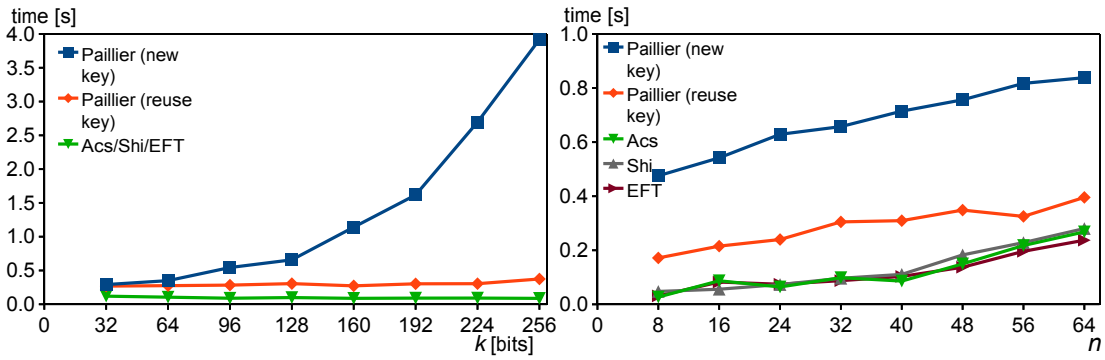


Figure 6.5: The average runtimes of a protocol for different encryption key sizes k ($n = 32$) and different number of participants n ($k = 128$).

Figure 6.5 shows the average runtime of a single round for encryption keys of different sizes and different amounts of participants. Since results of Ács, Shi, and EFT schemes are very similar, we represent them as a single line, when evaluating schemes against different encryption key sizes. Generating and distributing a set of encryption keys in the Paillier scheme is a very time consuming process. Increasing the key size k, significantly increases computation time for the *new key* scenario, and have a negligible overhead when one key is used all the time in the *reuse key* scenario.

Despite the encryption overhead, the homomorphic encryption schemes scale well. Adding new nodes, while keeping the same encryption key size, increases the average runtime of all homomorphic encryption schemes linearly.

Shamir's Secret Sharing

In this experiment we evaluate the impact of the threshold t in the Shamir's secret sharing scheme. The threshold t represents the number of shares that are necessary to reconstruct the secret. Thus, its value defines the minimal number of colluding nodes that can break security of the scheme. By modifying the value of t, one modifies also the fault tolerance level of the protocol (Table 6.1), but with a negligible impact on its performance.

Figure 6.6 shows the average runtimes of Shamir scheme protocols for different threshold t. Increasing t increases the runtime minimally. The results seem to be surprising, but are explainable. Each participant while running the protocol sends messages to all remaining

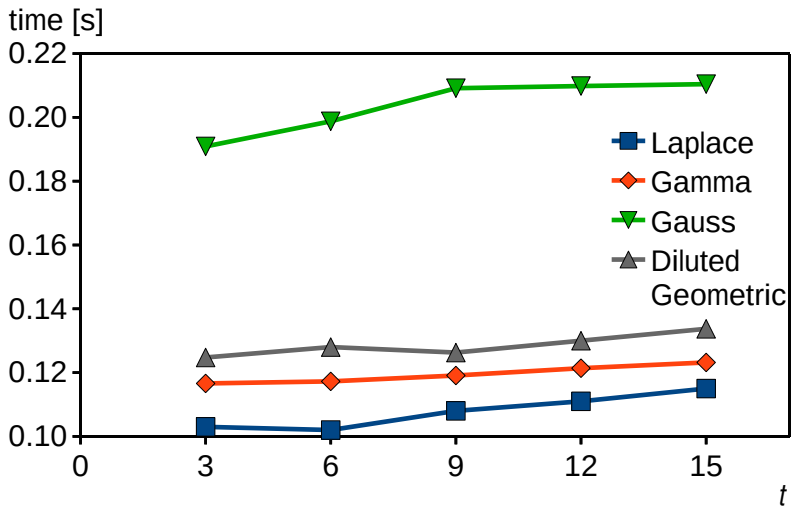


Figure 6.6: The average runtimes for different Shamir's scheme threshold t ($n = 32$) and privacy mechanisms.

nodes, i.e., to $(n-1)$ nodes, regardless of t. The only time that is gained for smaller t comes from reconstructing the result, which starts as soon as t shares are collected by a node without waiting for remaining shares to arrive. However, since our network is fast and reliable, the difference in runtimes are small.

Differences among privacy mechanisms are results of their implementation in the secret sharing scheme. Gauss mechanism requires running additional subprotocols that securely compute the sum of squared numbers.

Performance of Fault Tolerance Schemes. The goal of this experiment is to compare performance of fault tolerant schemes, i.e., Shamir, Paillier, Ács, and our EFT, with one data contributor faulting. Notice that we extended the original Ács scheme with the same recovery subprotocol as used in the EFT scheme. Remaining schemes will rerun the protocol, if any participant drops. We set $r = s = \gamma n = 3$.

Figure 6.7 shows the average runtimes for fault tolerant protocols when a single data contributor faulted. In such scenario the runtime of a protocol implemented in either Paillier or Shamir schemes is reduced due to less communication and computation that is performed. At the same time Ács and our scheme needed more time to run a recovery protocol and retrieve the final result. Despite additional computations our scheme is as efficient as Shamir scheme for fewer contributors ($n \leqslant 32$), but scales better when we increase n. Runtimes

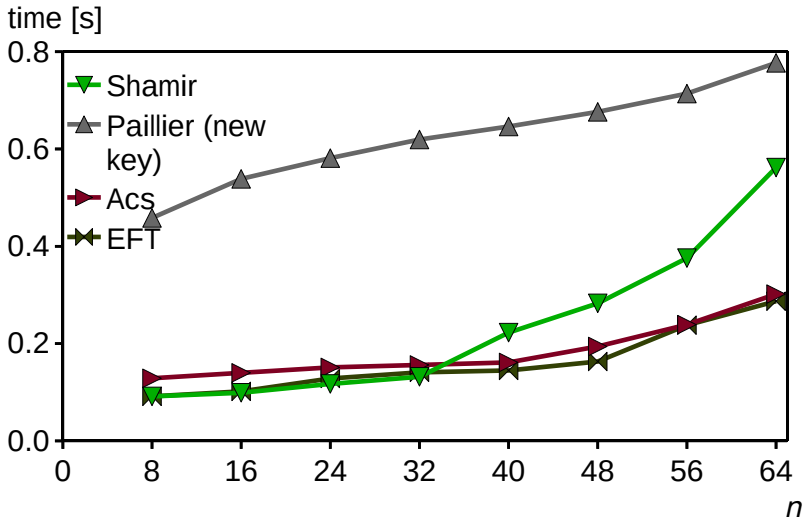


Figure 6.7: The average runtimes for different numbers of participants and fault tolerant security schemes.

for the Ács scheme are slightly longer than for our scheme, which is caused by the need of reestablishing encryption keys before each run.

Data Preparation Overhead. For all protocols majority of their computations are local and prior to any communication with other parties. Therefore, before comparing protocols for different scenarios, we run an experiment to evaluate time needed by each node to prepare its data before sending them to other nodes.

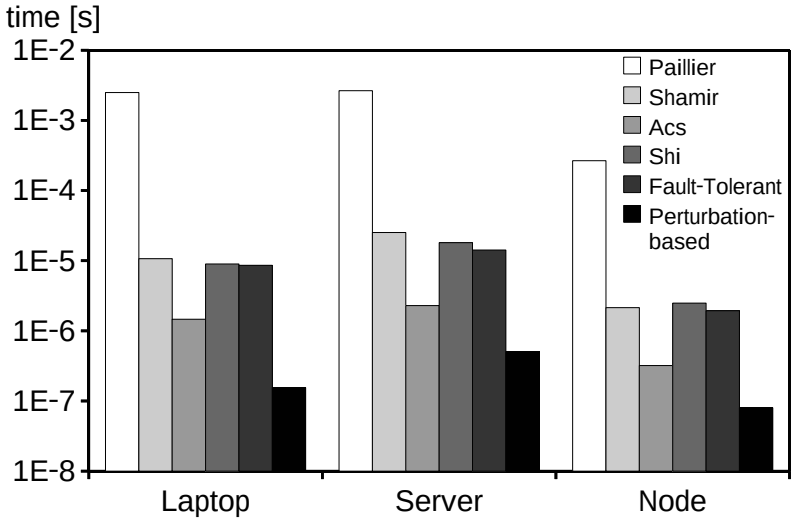


Figure 6.8: The average local computation times (logarithmic scale) for data preparations in different security schemes on different platforms.

Figure 6.8 shows the average local computation time (logarithmic scale) for data preparation of security schemes on different platforms. The runtime includes all random number generations, encryptions, and any other necessary computations. The Perturbation-based scheme outperforms others by at least two orders of magnitude. In this scheme each node generates at most one random number, which is a relatively easy task. Each node running Shamir, or Ács, or Shi, or Fault-Tolerant scheme spends more time before communicating with others. However, it is still two orders of magnitude faster than for the Paillier scheme with 128-bit key. Notice that, we have measured only the encryption time, and skipped the time spent on keys generation.

Overall Protocol Performance. In this experiment we compare the overall performance for all security schemes and different numbers of nodes. Since different privacy mechanisms have little impact on the overall performance, in all runs we use Laplace LDPA. Figure 6.9 shows runtimes of the distributed aggregation protocols implemented in different security schemes.

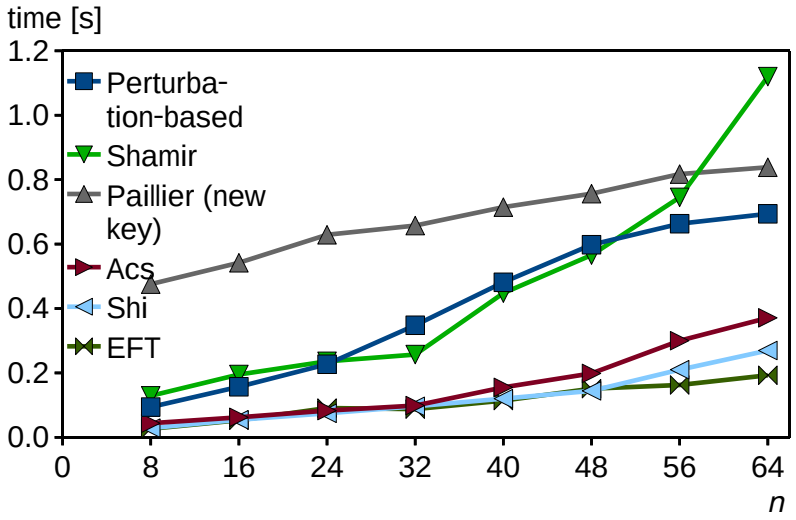


Figure 6.9: The average runtimes for different numbers of nodes and security schemes.

Notice that for security, all parties in both Paillier and Ács schemes reestablish their encryption keys in each round. As the number of nodes increases, both Perturbation-based and Shamir schemes do not scale well as the increasing communication cost becomes the dominant overhead. Communication in the Perturbation-based scheme grows linearly with

number of participants (Table 6.1), but is synchronized, i.e., each participant (except the one initiating computations) send a message after receiving it from the previous participant in the ring. Such communication is not asynchronous, which impacts scalability. Communication in the Shamir scheme grows quadratically with number of nodes (Table 6.1), which is confirmed in our experiments.

When the number of nodes is between 32 and 48, the secret sharing scheme outperforms the Perturbation-based protocol. This may seem counter-intuitive at the first glance, especially when we consider the large amount of communication required for exchanging the secret shares in the secret sharing scheme. However, we note that nodes are connected pair-wise in the secret sharing scheme, while they are connected in a ring topology in the Perturbation-based protocol. Thus, passing a message to all other nodes takes less time in the former scheme. When the number of nodes increase further, the amount of communication in the secret sharing scheme outweighs the benefit of all-to-all connections topology and hence it is outperformed by the Perturbation-based protocol again.

On the other hand, all homomorphic encryption schemes scale well due to their low communication costs. However, the Paillier scheme has a significant computation overhead comparing to others, which limits its scalability. Among encryption schemes our scheme is the fastest one, because in each round encryption keys do not have to be regenerated and the encryption function has low time complexity.

Chapter 7

Conclusions and Future Work

7.1 Summary

In this dissertation we presented and addressed challenges of privacy preserving data release, which have different distributed data settings, and cover syntactic and semantic privacy notions.

7.1.1 Syntactic Privacy Notions in Distributed Environments

For syntactic privacy notions we described and addressed challenges introduced by a set of m colluding data providers (m-adversary) in collaborative data publishing. Privacy threats introduced by m-adversaries are addressed by our new privacy notion, m-privacy, defined with respect to a privacy constraint C. We also proved that both verification of m-privacy fulfillment and anonymization datasets in order to achieve m-privacy are, in general, computationally hard.

To verify m-privacy w.r.t. any syntactic privacy notion C (Chapter 3) we presented new heuristics and SMC protocols. A few of them check m-privacy for EG monotonic privacy constraints, and use adaptive ordering techniques to improve computations efficiency. For non-EG monotonic constraints we introduced an algorithm, with minimal required number of privacy checks. We also presented a *provider-aware* anonymization algorithm with an adaptive verification strategy to ensure high utility and m-privacy of anonymized data. Experimental results confirmed that our heuristics perform better or comparable with

existing algorithms in terms of efficiency and utility.

We implemented all verification and anonymization algorithms in two distributed data settings, with and without a trusted third party (TTP). All secure multiparty computation protocols have been presented in details with their security and complexity carefully analyzed (Chapter 4). In addition, we extensively tested their performance and quality of anonymized datasets. Implementations of algorithms for the TTP setting is available on-line for further development [102].

7.1.2 Semantic Privacy Notions in Distributed Environments

In this dissertation we also studied settings with semantic privacy notions and new challenges, which they introduce. One of them is customization of privacy settings for differential privacy, i.e., we allow data providers to set their privacy budgets independently (Chapter 5). Customized privacy budget can be also an outcome of a query workload that covers only a subset of records by its queries. For such settings, we proposed a two-phase approach to generate data histograms, which differential preserve privacy and maximizes data utility. First, data are partitioned into v-optimal partitions based on records' privacy budget. Then, each partition is used to create a data histogram using any state-of-the-art algorithm with bucket counts perturbed to ensure differential privacy. Separating privacy partitioning from data-driven methods of building histograms gave us flexibility in choosing algorithms for each step independently. All presented algorithms have been evaluated extensively in experiments.

Another challenge, which we addressed is collusion of data providers in distributed settings with semantic privacy notions. We introduced a new privacy-aware method of computing data statistics, in which each data provider generates only partial noise from the same data distribution. After combining all partial noise values, the final result is perturbed enough to achieve differential privacy, even when we remove records from a few providers. In order to ensure that level of collusion resistance some redundant noise needs to be generated. Our method is designed in a such way that it generates very little redundant noise comparing to naïve and existing approaches.

To choose an efficient security scheme and a privacy mechanism for distributed and

privacy-preserving computations a few privacy and security challenges needs to be addressed. In this dissertations we introduced and evaluated a new, efficient, and fault tolerant scheme (EFT) to ensure reliability and collusion resistant of computations. Our security scheme guarantees computational security and minimal communication together with a high reliability of the system (Chapter 6). To address privacy challenges for distributed settings, we proposed a new and efficient distributed perturbation mechanism (Laplace DLPA), which introduces only small amount of redundant noise. The choice of a privacy mechanism is crucial to preserve utility of final results, and impacts performance significantly for devices with limited power and computation resources, e.g., mobile devices.

7.2 Future Work

The notion of m-privacy has ben introduced for horizontally partitioned datasets. Adapting it to vertically distributed or ad-hoc created datasets are two interesting directions for future research. Also, generalizing our approach to other kinds of data, such as set-valued data, defines another interesting path to explore.

In times of shifting from stand-alone to mobile devices with limited amount of resources, adapting both security schemes and privacy mechanisms to such new environments introduces many new challenges. Existing approaches to perform privacy-preserving computations have been designed mostly for computers with relatively amount of computation resources and with reliable power source. Therefore, running such algorithms and protocols without adapting them to mobile devices is suboptimal. Reducing complexity of such computations and minimizing their communication overhead is also a very interesting dimension of research.

Ensuring security and privacy of complex tasks is barely explored nowadays. Even for simple tasks (e.g. creating a histogram) there are still some room for improvement their performance.

An edge case of customized privacy budget setting is allowing each data owner to set its own budget. That way privacy would be personalized for each data owner. Such variety of different privacy budget values define a new environment for future studies. For example, to

improve utilization of privacy budget by each record, a new techniques of budget saturation should be proposed and analyzed.

Bibliography: Books and Journals

[1] Gergely Ács and Claude Castelluccia. "I have a DREAM!: differentially private smart metering". In: *Proceedings of the 13th International Conference on Information Hiding*. IH'11. 2011, pp. 118–132. ISBN: 978-3-642-24177-2.

[2] Gagan Aggarwal, Nina Mishra, and Benny Pinkas. "Secure Computation of the kth-Ranked Element". In: *Proceedings of the 23rd International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*. EUROCRYPT'04. 2004, pp. 40–55. ISBN: 978-3-540-21935-4.

[3] Joachim H. Ahrens and Ulrich Dieter. "Computer methods for sampling from gamma, beta, Poisson and bionomial distributions". In: *Computing* 12 (3 1974), pp. 223–246. ISSN: 0010-485X.

[4] Dima Alhadidi, Noman Mohammed, Benjamin C. M. Fung, and Mourad Debbabi. "Secure distributed framework for achieving ϵ-differential privacy". In: *Proceedings of the 12th International Privacy Enhancing Technologies Symposium*. Vol. 7384. Lecture Notes in Computer Science. 2012, pp. 120–139. ISBN: 978-3-642-31679-1.

[5] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank D. McSherry, and Kunal Talwar. "Privacy, Accuracy, and Consistency Too: A Holistic Solution to Contingency Table Release". In: *Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS'07. Beijing, China, 2007, pp. 273–282. ISBN: 978-1-59593-685-1.

[6] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. "Completeness theorems for non-cryptographic fault-tolerant distributed computation". In: *Proceedings of the*

20th Annual ACM Symposium on Theory of Computing. STOC'88. Chicago, Illinois, United States, 1988, pp. 1–10. ISBN: 0-89791-264-0.

[7] George Boros and Victor Moll. *Irresistible Integrals: Symbolics, Analysis and Experiments in the Evaluation of Integrals*. Cambridge University Press, 2004. ISBN: 978-0521796361.

[8] George Edward Pelham Box and Mervin Edgar Muller. "A Note on the Generation of Random Normal Deviates". In: *The Annals of Mathematical Statistics* 29.2 (1958), pp. 610–611.

[9] Yuriy Brun and Nenad Medvidović. "Entrusting Private Computation and Data to Untrusted Networks". In: *IEEE Transactions on the Dependable and Secure Computing* 10.4 (July 2013), pp. 225–238.

[10] Cristian Bucilă, Johannes Gehrke, Daniel Kifer, and Walker White. "DualMiner: A Dual-Pruning Algorithm for Itemsets with Constraints". In: *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD'02. Edmonton, Alberta, Canada, 2002, pp. 42–51. ISBN: 1-58113-567-X.

[11] Jesse Burke, Deborah Estrin, Mark Hansen, Andrew Parker, Nithya A. Ramanathan, Sasank Reddy, and Mani B. Srivastava. "Participatory Sensing". In: *Workshop on World-Sensor-Web (WSW): Mobile Device Centric Sensor Networks and Applications*. 2006.

[12] Robin Burke, Bamshad Mobasher, Roman Zabicki, and Runa Bhaumik. "Identifying attack models for secure recommendation". In: *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems at the International Conferece on Intelligent User Interfaces*. San Diego, California, USA, 2005.

[13] Martin Burkhart and Xenofontas A. Dimitropoulos. "Fast Privacy-Preserving Top-k Queries Using Secret Sharing". In: *Proceedings of 19th International Conference on the Computer Communications and Networks*. ICCCN'10. 2010, pp. 1–7.

[14] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. "SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics". In: *USENIX Security Symposiun*. USENIX, 2010.

[15] Baki Cakici, Kenneth Hebing, Maria Grünewald, Paul Saretok, and Anette Hulth. "CASE: A framework for computer supported outbreak detection". In: *BMC Medical Informatics and Decision Making* 10.1 (2010), p. 14.

[16] Claude Castelluccia, Aldar C-F. Chan, Einar Mykletun, and Gene Tsudik. "Efficient and provably secure aggregation of encrypted data in wireless sensor networks". In: *ACM Transactions on Sensor Networks (TOSN)* 5.3 (June 2009), 20:1–20:36. ISSN: 1550-4859.

[17] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. "Efficient Aggregation of encrypted data in Wireless Sensor Networks". In: *Proc. of the 2nd Annual International Conference on on Mobile and Ubiquitous Systems: Networking and Services. MobiQuitous 2005*. MOBIQUITOUS'05. 2005, pp. 109–117. ISBN: 0-7695-2375-7.

[18] Octavian Catrina and Amitabh Saxena. "Secure computation with fixed-point numbers". In: *Proceedings of the 14th International Conference on Financial Cryptography and Data Security*. FC'10. Tenerife, Spain, 2010, pp. 35–50. ISBN: 3-642-14576-0, 978-3-642-14576-6.

[19] Cheng-Kang Chu, Wen Tao Zhu, Sherman S. M. Chow, Jianying Zhou, and Robert H. Deng. "Secure mobile subscription of sensor-encrypted data". In: *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*. ASIACCS'11. 2011, pp. 228–237. ISBN: 978-1-4503-0564-8.

[20] Chris Clifton, Murat Kantarcıoğlu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. "Tools for privacy preserving distributed data mining". In: *ACM SIGKDD Explorations Newsletter* 4 (2 2002), pp. 28–34. ISSN: 1931-0145.

[21] Graham Cormode, Cecilia Procopiuc, Divesh Srivastava, Entong Shen, and Ting Yu. "Differentially Private Spatial Decompositions". In: *Proceedings of the 2012 IEEE*

28th International Conference on Data Engineering. ICDE'12. 2012, pp. 20–31. ISBN: 978-0-7695-4747-3.

[22] Graham Cormode, Divesh Srivastava, Ninghui Li, and Tiancheng Li. "Minimizing Minimality and Maximizing Utility: Analyzing Method-based Attacks on Anonymized Data". In: *Proceedings of the VLDB Endowment* 3 (1–2 Sept. 2010), pp. 1045–1056. ISSN: 2150-8097.

[23] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. "Multiparty Computation from Threshold Homomorphic Encryption". In: *Proceedings of the 20th International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*. EUROCRYPT'01. 2001, pp. 280–299. ISBN: 3-540-42070-3.

[24] Ivan Damgård and Mats Jurik. "A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System". In: *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*. PKC'01. 2001, pp. 119–136. ISBN: 3-540-41658-7.

[25] Ivan Damgård and Gert Læssøe Mikkelsen. "Efficient, Robust and Constant-Round Distributed RSA Key Generation". In: *Theory of Cryptography*. Ed. by Daniele Micciancio. Vol. 5978. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 183–200. ISBN: 978-3-642-11798-5.

[26] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. "Fully Homomorphic Encryption over the Integers". In: *Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques*. EUROCRYPT'10. French Riviera, France, 2010, pp. 24–43. ISBN: 3-642-13189-1, 978-3-642-13189-9.

[27] Cynthia Dwork. "A firm foundation for private data analysis". In: *Communications of the ACM* 54.1 (2011), pp. 86–95. ISSN: 0001-0782.

[28] Cynthia Dwork. "Differential privacy". In: *Proceedings of the 33rd International Conference on Automata, Languages and Programming — Volume Part II*. ICALP'06. 2006, pp. 1–12. ISBN: 3-540-35907-9, 978-3-540-35907-4.

[29] Cynthia Dwork. "Differential privacy: a survey of results". In: *Proceedings of the 5th International Conference on Theory and Applications of Models of Computation*. TAMC'08. Xi'an, China, 2008, pp. 1–19. ISBN: 3-540-79227-9, 978-3-540-79227-7.

[30] Cynthia Dwork, Krishnaram Kenthapadi, Frank D. McSherry, Ilya Mironov, and Moni Naor. "Our data, ourselves: privacy via distributed noise generation". In: *Proceedings of the 25th Annual International Conference on The Theory and Applications of Cryptographic Techniques*. EUROCRYPT'06. 2006, pp. 486–503. ISBN: 3-540-34546-9, 978-3-540-34546-6.

[31] Cynthia Dwork, Frank D. McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis". In: *Proceedings of the 3rd conference on Theory of Cryptography*. TCC'06. 2006, pp. 265–284. ISBN: 3-540-32731-2, 978-3-540-32731-8.

[32] Taher El Gamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *Proceedings of CRYPTO 84 on Advances in Cryptology*. 1985, pp. 10–18. ISBN: 0-387-15658-5.

[33] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. "Privacy-preserving data publishing: A survey of recent developments". In: *ACM Computing Surveys (CSUR)* 42.4 (2010), 14:1–14:53. ISSN: 0360-0300.

[34] Tamas S. Gal, Zhiyuan Chen, and Aryya Gangopadhyay. "A Privacy Protection Model for Patient Data with Multiple Sensitive Attributes". In: *International Journal of Information Security and Privacy* 2.3 (2008), pp. 28–44.

[35] Simson L. Garfinkel and Michael D. Smith. "Guest Editors' Introduction: Data Surveillance". In: *IEEE Security & Privacy* 4.6 (2006).

[36] Craig Gentry. "Fully homomorphic encryption using ideal lattices". In: *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*. STOC'09. Bethesda, MD, USA, 2009, pp. 169–178. ISBN: 978-1-60558-506-2.

[37] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. "Universally utility-maximizing privacy mechanisms". In: *Proceedings of the 41st annual ACM symposium on Theory of Computing*. STOC'09. 2009, pp. 351–360. ISBN: 978-1-60558-506-2.

[38] Oded Goldreich. *Foundations of Cryptography: Volume 1. Basic Tools*. Cambridge University Press, 2007. ISBN: 9780521035361.

[39] Oded Goldreich. *Foundations of Cryptography: Volume 2. Basic Applications*. Cambridge University Press, 2004. ISBN: 9780521830843.

[40] Oded Goldreich, Silvio M. Micali, and Avi Wigderson. "How to play ANY mental game". In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*. STOC'87. 1987, pp. 218–229. ISBN: 0-89791-221-7.

[41] Slawomir Goryczka, Li Xiong, and Benjamin C. M. Fung. "m-Privacy for collaborative data publishing". In: *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. CollaborateCom'11. 2011, pp. 1–10.

[42] Slawomir Goryczka, Li Xiong, and Benjamin C. M. Fung. "m-Privacy for Collaborative Data Publishing". In: *IEEE Transactions on Knowledge and Data Engineering* 99.PrePrints (2013), p. 1. ISSN: 1041-4347.

[43] Slawomir Goryczka, Li Xiong, and Vaidy S. Sunderam. "Secure multiparty aggregation with differential privacy: a comparative study". In: *EDBT/ICDT Workshops*. 2013, pp. 155–163. ISBN: 978-1-4503-1599-9.

[44] Hamed Haddadi, Richard Mortier, and Steven Hand. "Privacy analytics". In: *ACM SIGCOMM Computer Communication Review* 42.2 (Mar. 2012), pp. 94–98. ISSN: 0146-4833.

[45] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. "Boosting the Accuracy of Differentially Private Histograms Through Consistency". In: *Proceedings of the VLDB Endowment* 3.1-2 (Sept. 2010), pp. 1021–1032. ISSN: 2150-8097.

[46] Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, and Tomas Toft. "Efficient RSA key generation and threshold Paillier in the two-party setting". In: *Proceedings of the 12th Conference on Topics in Cryptology*. CT-RSA'12. San Francisco, CA, 2012, pp. 313–331. ISBN: 978-3-642-27953-9.

[47] Hosagrahar Visvesvaraya Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, and Torsten Suel. "Optimal Histograms with Quality Guarantees". In: *Proceedings of the 24rd International Conference on Very Large Data Bases*. VLDB'98. 1998, pp. 275–286. ISBN: 1-55860-566-5.

[48] Wei Jiang and Chris Clifton. "A secure distributed framework for achieving k-anonymity". In: *The VLDB Journal — The International Journal on Very Large Data Bases* 15.4 (2006), pp. 316–333.

[49] Wei Jiang and Chris Clifton. "Privacy-Preserving Distributed k-Anonymity". In: *Proceedings of the 19th annual IFIP WG 11.3 working conference on Data and Applications Security*. Vol. 3654. DBSec'05. Storrs, CT, 2005, pp. 166–177. ISBN: 3-540-28138-X, 978-3-540-28138-2.

[50] Norman Lloyd Johnson, Adrienne W. Kemp, and Samuel Kotz. *Univariate Discrete Distributions*. Wiley Series in Probability and Statistics. Wiley, 2005. ISBN: 9780471715801.

[51] Pawel Jurczyk and Li Xiong. "Distributed Anonymization: Achieving Privacy for Both Data Subjects and Data Providers". In: *Proceedings of the 23rd Annual IFIP WG 11.3 Working Conference: Data and Applications Security XXIII*. Vol. 5645. Lecture Notes in Computer Science. 2009, pp. 191–207. ISBN: 978-3-642-03006-2.

[52] Jerry Kang, Katie Shilton, Deborah Estrin, Jeff Burke, and Mark Hansen. "Self-Surveillance Privacy". In: *Iowa Law Review* 97 (2012), pp. 809–847.

[53] Daniel Kifer. "Attacks on privacy and deFinetti's theorem". In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. Providence, Rhode Island, USA: ACM, 2009, pp. 127–138. ISBN: 978-1-60558-551-2.

[54] Daniel Kifer and Ashwin Machanavajjhala. "No free lunch in data privacy". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. SIGMOD'11. 2011, pp. 193–204. ISBN: 978-1-4503-0661-4.

[55] Samuel Kotz, Tomasz J. Kozubowski, and Krzysztof Podgórski. *The Laplace Distribution and Generalizations: A Revisit with Applications to Communications, Economics, Engineering, and Finance*. Progress in Mathematics Series. Springer, 2001. ISBN: 9780817641665.

[56] Kristen Lefevre, David J. Dewitt, and Raghu Ramakrishnan. "Mondrian multidimensional k-anonymity". In: *ICDE*. 2006.

[57] Ninghui Li and Tiancheng Li. "t-Closeness: Privacy Beyond k-Anonymity and l-Diversity". In: *Proceedings of the 23rd International Conference on Data Engineering*. ICDE'07. 2007.

[58] Yehuda Lindell and Benny Pinkas. "An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries". In: *Proceedings of the 26th Annual International Conference on The Theory and Applications of Cryptographic Techniques*. EUROCRYPT'07. Barcelona, Spain, 2007, pp. 52–78. ISBN: 978-3-540-72539-8.

[59] Yehuda Lindell and Benny Pinkas. "Secure Multiparty Computation for Privacy-Preserving Data Mining". In: *The Journal of Privacy and Confidentiality* 1.1 (2009), pp. 59–98.

[60] Ashwin Machanavajjhala, Johannes Gehrke, Daniel Kifer, and Muthuramakrishnan Venkitasubramaniam. "l-Diversity: Privacy Beyond k-Anonymity". In: *Proceedings of the 22nd International Conference on Data Engineering*. ICDE'06. 2006, p. 24. ISBN: 0-7695-2570-9.

[61] George Marsaglia and Wai Wan Tsang. "The Ziggurat Method for Generating Random Variables". In: *Journal of Statistical Software* 5.8 (Oct. 2000), pp. 1–7. ISSN: 1548-7660.

[62] Frank D. McSherry. "Privacy integrated queries: an extensible platform for privacy-preserving data analysis". In: *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*. SIGMOD'09. Providence, Rhode Island, USA, 2009, pp. 19–30. ISBN: 978-1-60558-551-2.

[63] Frank D. McSherry and Kunal Talwar. "Mechanism Design via Differential Privacy". In: *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*. FOCS'07. 2007, pp. 94–103.

[64] Noman Mohammed, Dima Alhadidi, Benjamin C. M. Fung, and Mourad Debbabi. "Secure Two-Party Differentially Private Data Release for Vertically Partitioned Data". In: *IEEE Transactions on Dependable and Secure Computing* 11.1 (Jan. 2014), pp. 59–71. ISSN: 1545-5971.

[65] Noman Mohammed, Rui Chen, Benjamin C. M. Fung, and Philip S. Yu. "Differentially private data release for data mining". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD'11. San Diego, California, USA, 2011, pp. 493–501. ISBN: 978-1-4503-0813-7.

[66] Noman Mohammed, Benjamin C. M. Fung, Patrick C. K. Hung, and Cheuk-Kwong Lee. "Centralized and distributed anonymization for high-dimensional healthcare data". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4.4 (2010), 18:1–18:33.

[67] Noman Mohammed, Benjamin C. M. Fung, Ke Wang, and Patrick C. K. Hung. "Privacy-Preserving Data Mashup". In: *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. EDBT'09. Saint Petersburg, Russia, 2009, pp. 228–239. ISBN: 978-1-60558-422-5.

[68] Min Mun, Sasank Reddy, Katie Shilton, Nathan Yau, Jeff Burke, Deborah Estrin, Mark Hansen, Eric Howard, Ruth West, and Péter Boda. "PEIR, the Personal Environmental Impact Report, As a Platform for Participatory Sensing Systems Research". In: *Proceedings of the 7th International Conference on Mobile Systems, Applications, and Services*. MobiSys'09. Kraków, Poland, 2009, pp. 55–68. ISBN: 978-1-60558-566-6.

[69] Mehmet Ercan Nergiz, Abdullah Ercüment Çiçek, Thomas B. Pedersen, and Yücel Saygın. "A Look-Ahead Approach to Secure Multiparty Protocols". In: *IEEE Transactions on Knowledge and Data Engineering* 24 (2012), pp. 1170–1185. ISSN: 1041-4347.

[70] Takashi Nishide and Kouichi Sakurai. "Distributed Paillier Cryptosystem Without Trusted Dealer". In: *Proceedings of the 11th International Conference on Information Security Applications*. WISA'10. Jeju Island, Korea, 2011, pp. 44–60. ISBN: 3-642-17954-1, 978-3-642-17954-9.

[71] Pascal Paillier. "Public-key cryptosystems based on composite degree residuosity classes". In: *Proceedings of the 18th International Conference on Theory and Application of Cryptographic Techniques*. EUROCRYPT'99. 1999, pp. 223–238. ISBN: 3-540-65889-0.

[72] Thomas B. Pedersen, Yücel Saygın, and Erkay Savaş. "Secret Sharing vs. Encryption-based Techniques For Privacy Preserving Data Mining". In: *Joint UNECE/Eurostat work session on Statistical Data Confidentiality*. 2007.

[73] Stephen C. Pohlig and Martin E. Hellman. "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance (Corresp.)" In: *IEEE Transactions on Information Theory* 24.1 (2006), pp. 106–110. ISSN: 0018-9448.

[74] Viswanath Poosala, Peter J. Haas, Yannis E. Ioannidis, and Eugene J. Shekita. "Improved Histograms for Selectivity Estimation of Range Predicates". In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*. SIGMOD'96. Montreal, Quebec, Canada, 1996, pp. 294–305. ISBN: 0-89791-794-4.

[75] Layla Pournajaf, Li Xiong, Vaidy Sunderam, and Slawomir Goryczka. "Spatial Task Assignment for Crowd Sensing with Cloaked Locations". In: *15th IEEE International Conference on Mobile Data Management (MDM)*. July 2014.

[76] Vibhor Rastogi and Suman Nath. "Differentially private aggregation of distributed time-series with transformation and encryption". In: *Proceedings of the 2010 ACM*

SIGMOD International Conference on Management of Data. SIGMOD'10. 2010, pp. 735–746. ISBN: 978-1-4503-0032-2.

[77] *Report of the August 2010 Multi-Agency Workshop on InfoSymbiotics/DDDAS, The Power of Dynamic Data Driven Applications Systems*. Workshop sponsored by: Air Force Office of Scientific Research and National Science Foundation.

[78] Ronald L. Rivest, Adi Shamir, and Len Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[79] Alexander Russell and David Zuckerman. "Perfect Information Leader Election in $\log^* n+O(1)$ Rounds". In: *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*. Nov. 1998, pp. 576–583. ISBN: 0-8186-9172-7.

[80] Pierangela Samarati. "Protecting Respondents' Identities in Microdata Release". In: *IEEE Transactions on Knowledge and Data Engineering* 13.6 (2001), pp. 1010–1027.

[81] Adi Shamir. "How to share a secret". In: *Communications of the ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782.

[82] Rashid Sheikh, Beerendra Kumar, and Durgesh Kumar Mishra. "A Distributed k-Secure Sum Protocol for Secure Multi-Party Computations". In: *Journal of Computing* 2 (3 Mar. 2010), pp. 68–72. ISSN: 2151-9617.

[83] Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. "Privacy-Preserving Aggregation of Time-Series Data". In: *Proceedings of the 18th Annual Network and Distributed System Security Symposium*. NDSS'11. 2011.

[84] Katie Shilton. "Four Billion Little Brothers?: Privacy, mobile phones, and ubiquitous data collection". In: *Communications of the ACM* 52 (11 2009), pp. 48–53. ISSN: 0001-0782.

[85] Latanya Sweeney. "k-Anonymity: A Model for Protecting Privacy". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.5 (2002), pp. 557–570. ISSN: 0218-4885.

[86] Latanya Sweeney. *Uniqueness of Simple Demographics in the U.S. Population*. Tech. rep. Carnegie Mellon University, 2000.

[87] Yufei Tao, Xiaokui Xiao, Jiexing Li, and Donghui Zhang. "On Anti-Corruption Privacy Preserving Publication". In: *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*. ICDE'08. 2008, pp. 725–734. ISBN: 978-1-4244-1836-7.

[88] Jaideep Vaidya and Chris Clifton. "Secure set intersection cardinality with application to association rule mining". In: *Journal of Computer Security* 13 (4 July 2005), pp. 593–622. ISSN: 0926-227X.

[89] Michael M. Wagner, Andrew W. Moore, and Ron M. Aryel, eds. *Handbook of Biosurveillance*. Academic Press, June 2006. ISBN: 978-0123693785.

[90] Xiaokui Xiao, Gabriel Bender, Michael Hay, and Johannes Gehrke. "iReduct: Differential Privacy with Reduced Relative Errors". In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. SIGMOD'11. Athens, Greece, 2011, pp. 229–240. ISBN: 978-1-4503-0661-4.

[91] Xiaokui Xiao and Yufei Tao. "Personalized privacy preservation". In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*. SIGMOD'06. Chicago, IL, USA, 2006, pp. 229–240. ISBN: 1-59593-434-0.

[92] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. "Differential Privacy via Wavelet Transforms". In: *IEEE Transactions on Knowledge and Data Engineering* 23.8 (Aug. 2011), pp. 1200–1214. ISSN: 1041-4347.

[93] Yonghui Xiao, Li Xiong, Liyue Fan, and Slawomir Goryczka. "DPCube: Differentially Private Histogram Release through Multidimensional Partitioning". In: *ArXiv e-prints* (Feb. 2012). eprint: 1202.5358.

[94] Li Xiong, Vaidy S. Sunderam, Liyue Fan, Slawomir Goryczka, and Layla Pournajaf. "PREDICT: Privacy and Security Enhancing Dynamic Information Collection and Monitoring". In: *Proceedings of the International Conference on Computational Science*. Vol. 18. ICCS'13. June 2013, pp. 1979–1988.

[95] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. "Differentially private histogram publication". In: *The VLDB Journal — The International Journal on Very Large Data Bases* 22.6 (2013), pp. 797–822. ISSN: 1066-8888.

[96] Andrew Chi-Chih Yao. "How to generate and exchange secrets". In: *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*. SFCS'86. IEEE, 1986, pp. 162–167.

[97] W. Katherine Yih, Swati Deshpande, Candace Fuller, Dawn Heisey-Grove, John Hsu, Benjamin A. Kruskal, Martin Kulldorff, Michael Leach, James Nordin, Jessie Patton-Levine, Ella Puga, Edward Sherwood, Irene Shui, and Richard Platt. "Evaluating real-time syndromic surveillance signals from ambulatory care data in four states". In: *Public Health Reports* 125.1 (2010).

[98] Sheng Zhong, Zhiqiang Yang, and Rebecca N. Wright. "Privacy-enhancing k-anonymization of customer data". In: *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. PODS'05. Baltimore, Maryland, 2005, pp. 139–147. ISBN: 1-59593-062-0.

Bibliography: Electronic Resources

[99] *2009 H1N1 Flu*. `http://www.cdc.gov/h1n1flu/`. 2009.

[100] *Colt: Open Source Libraries for High Performance Scientific and Technical Computing in Java*. `http://acs.lbl.gov/software/colt`.

[101] *Investigation Update: Outbreak of Shiga toxin-producing E. coli O104 (STEC O104:H4) Infections Associated with Travel to Germany*. `http://www.cdc.gov/ecoli/2011/ecoliO104/`. 2011.

[102] *m-Anonymizer: Collaborative Distributed Anonymization Library with m-Privacy*. `http://www.mathcs.emory.edu/aims/m-anonymizer/`. 2011.

[103] *SEPIA: Security through Private Information Aggregation*. `http://sepia.ee.ethz.ch`.

[104] *UTD Paillier Threshold Encryption Toolbox: an Open Source Library*. `http://www.utdallas.edu/~mxk093120/paillier`.