OUTSOURCING COMPUTATION ON NON-ENCRYPTED SENSITIVE DATA TO UNTRUSTED CLOUDS

Sara Ricci

UNIVERSITAT
ROVIRA i VIRGILI

Outsourcing Computation on Non-Encrypted Sensitive Data to Untrusted Clouds

---

SARA RICCI

DOCTORAL THESIS
2018

UNIVERSITAT ROVIRA I VIRGILI
OUTSOURCING COMPUTATION ON NON-ENCRYPTED SENSITIVE DATA TO UNTRUSTED CLOUDS
Sara Ricci

UNIVERSITAT ROVIRA I VIRGILI
OUTSOURCING COMPUTATION ON NON-ENCRYPTED SENSITIVE DATA TO UNTRUSTED CLOUDS
Sara Ricci

UNIVERSITAT ROVIRA I VIRGILI
OUTSOURCING COMPUTATION ON NON-ENCRYPTED SENSITIVE DATA TO UNTRUSTED CLOUDS
Sara Ricci

Sara Ricci

Outsourcing Computation on Non-Encrypted Sensitive Data to Untrusted Clouds

DOCTORAL THESIS

Supervised by Prof. Josep Domingo-Ferrer and
Dr. Jordi Soria-Comas

Department of Computer Engineering
and Mathematics

UNIVERSITAT ROVIRA i VIRGILI

March, 2018

UNIVERSITAT ROVIRA i VIRGILI

FAIG CONSTAR que aquest treball, titulat “Outsourcing Computation on Non-Encrypted Sensitive Data to Untrusted Clouds”, que presenta Sara Ricci per a l’obtenció del títol de Doctor, ha estat realitzat sota la meva direcció al Departament d’Enginyeria Informàtica i Matemàtiques d’aquesta universitat.

HAGO CONSTAR que el presente trabajo, titulado “Outsourcing Computation on Non-Encrypted Sensitive Data to Untrusted Clouds”, que presenta Sara Ricci para la obtención del título de Doctor, ha sido realizado bajo mi dirección en el Departamento de Ingeniería Informática y Matemáticas de esta universidad.

I STATE that the present study, entitled “Outsourcing Computation on Non-Encrypted Sensitive Data to Untrusted Clouds”, presented by Sara Ricci for the award of the degree of Doctor, has been carried out under my supervision at the Department of Computer Engineering and Mathematics of this university.

Tarragona, 09-03-2018

El/s director/s de la tesi doctoral
El/los director/es de la tesis doctoral
Doctoral Thesis Supervisor/s

DOMINGO FERRER JOSEP - 33890313C

Digitally signed by DOMINGO FERRER JOSEP - 33890313C
DN: c=ES, serialNumber=33890313C, sn=DOMINGO FERRER, givenName=JOSEP, cn=DOMINGO FERRER JOSEP - 33890313C
Date: 2018.03.10 10:21:05 +01'00'

[signature] / [firma] / [signature]

[signature] / [firma] / [signature]

Acknowledgments

I would like to thank my advisors Prof. Josep Domingo-Ferrer and Dr. Jordi Soria-Comas for their guidance in every step of my doctorate. I am very grateful to Dr. Oriol Farràs who patiently introduced me to secret sharing. The guidance of Dr. David Sánchez as the patience of Mónica Muñoz-Batista are also gratefully acknowledged. I am also indebted to all CRISES group members, especially to Jesús Manjón and Romina Russo.

Thanks to Augusto Fuschini and Prof. Carlo Traverso who have both believed in my abilities and taught me the beauty of doing research. In the end but not least, I am very grateful to my family, Petr and all my friends for their unconditional support and love.

This work was partly funded by the European Commission (projects H2020 644024 "CLARUS" and H2020 700540 "CANVAS"), the Government of Catalonia (grant 2014 SGR 537), and the Spanish Government (projects TIN2014-57364-C2-R "SmartGlacis" and TIN2016-80250-R "Sec-MCloud", and predoctoral FPI grant BES-2015-071402).

Contents

CONTENTS

List of Figures

List of Tables

Abstract

Data controllers accumulate more and more data on people, of which a substantial proportion are personally identifiable and hence sensitive data. Storing and processing these data in local premises is increasingly inconvenient, but resorting to cloud storage and processing raises security and privacy issues. We tackle here the problem of outsourcing to untrusted clouds in a practical and privacy-preserving manner two basic operations on non-encrypted sensitive data: scalar products and matrix products. These operations are useful to perform data analyses such as correlations between attributes or contingency tables, among others. Specifically, we propose several secure protocols to outsource to multiple clouds the computation of a variety of multivariate analyses on nominal data (frequency-based and semantic-based). These analyses are challenging, and they are even harder when data are nominal (i.e., textual, non-ordinal), because the standard arithmetic operators cannot be used.

Our protocols allow using the cloud not only to store sensitive non-encrypted data, but also to process them. We consider two variants of honest-but-curious clouds: clouds that do not share information with each other and clouds that may collude by sharing information with each other. In addition to analyzing the security of the proposed protocols, we also evaluate their performance against a baseline consisting of downloading plus local computation. Our protocols have been designed to outsource as much workload as possible to the clouds, in order to retain the cost-saving benefits of cloud computing while ensuring that the outsourced data stay split and, hence, they are privacy-protected versus the clouds. In addition to analyzing the security of the proposed protocols, we also evaluate their performance against a baseline consisting of downloading plus local computation The experiments on categorical data that we report on the Amazon cloud service show that, with our protocols, the data controller can save more than 99.999% runtime for the most demanding computations.

We also present here a methodology to compare statistical disclosure con-

trol (SDC) methods for microdata in terms of how they perform regarding the risk-utility trade-off. Previous comparative studies (e.g. [24]) usually start by selecting some parameter values for a set of SDC methods and evaluate the disclosure risk and the information loss yielded by the methods for those parameterizations. In contrast, here we start by setting a certain risk level (resp. utility preservation level) and then we find which parameter values are needed to attain that risk (resp. utility) under different SDC methods. Finally, once we have achieved an equivalent risk (resp. utility) level across methods, we evaluate the utility (resp. the risk) provided by each method, in order to rank methods according to their utility preservation (resp. disclosure protection). This ranking depends on a certain level of risk (resp. utility) and a certain original data set. The novelty of this comparison is not limited to the above-described methodology: we also justify and use general utility and risk measures that differ from those used in previous comparisons. Furthermore, we present experimental results of our methodology to compare the utility preservation of several methods given an equivalent level of risk for all of them.

Resum

La quantitat de dades sobre la gent que s'emmagatzema creix sense aturador. Moltes d'aquestes dades fan referència a individus concrets i, per tant, poden revelar informació sensible. L'emmagatzematge i el processament d'aquestes dades en entorns locals presenta certs inconvenients, però l'alternativa de fer servir el núvol per emmagatzemar-les i per processar-les amenaça la privadesa i la seguretat. En aquesta tesi afrontem els problemes derivats de l'ús de núvols no confiables. En particular, volem que el núvol pugui fer dues operacions bàsiques amb dades sensibles no xifrades de manera pràctica i amb privadesa: productes escalars i productes de matrius. Aquestes operacions són útils per fer diferents anàlisis de dades com ara el càlcul de correlacions entre atributs i el càlcul de taules de contingència. Específicament, proposem diversos protocols segurs per externalitzar a múltiples núvols el càlcul de diverses anàlisis multivariants sobre dades nominals (basades en freqüències i basades en semàntica). En general, aquestes anàlisis són complexes i ho són encara més amb dades nominals (per exemple, dades textuals, dades no ordinals) perquè les operacions aritmètiques estàndards no es poden fer servir.

Els nostres protocols ens permeten d'utilitzar el núvol no només per emmagatzemar dades sensibles no xifrades sinó també per processar-les. Considerem dues variants de núvols semihonrats: núvols que no comparteixen informació entre ells i núvols que poden col·laborar compartint informació. A banda d'analitzar la seguretat dels protocols proposats, també n'avaluem el rendiment respecte d'un protocol de referència que consisteix a descarregar les dades i processar-les localment. Els nostres protocols han estat dissenyats per externalitzar als núvols la major quantitat possible de càrrega de treball. D'aquesta manera podem conservar la rendibilitat de la computació al núvol alhora que ens assegurem que les dades externalizades romanen fragmentades i, per tant, preserven la privadesa envers el núvol. Les avaluacions experimentals amb dades categòriques que hem fet sobre el serveis de núvol d'Amazon mostren

que, amb els nostres protocols, l'administrador de les dades pot estalviar-se més del 99.999% de temps d'execució per als càlculs més exigents.

També presentem una metodologia per comparar mètodes de control de la revelació estadística (CRE) per a microdades en termes del compromís entre risc i utilitat. Els estudis comparatius anteriors comencen habitualment seleccionant alguns valors dels paràmetres per a un conjunt de mètodes CRE i avaluen el risc de revelació i la pèrdua d'informació que produeixen aquests paràmetres. Aquí comencem seleccionant un nivell de risc (resp. utilitat) i cerquem els valors dels paràmetres que calen per obtenir aquest nivell de risc (resp. utilitat) amb cadascun dels mètodes CRE avaluats. Finalment, un cop hem aconseguit un nivell de risc (resp. utilitat) equivalent en els diferents mètodes CRE, n'avaluem la utilitat (resp. risc), cosa que permet d'ordenar-los segons la utilitat (resp. risc). Aquesta ordenació depèn del nivell de risc (resp. utilitat) i del fitxer de dades original. La novetat d'aquesta comparació no es limita a la metodologia descrita prèviament. També proposem mesures d'utilitat i de risc diferents de les habituals. Addicionalment, presentem resultats experimentals de la nostra metodologia en l'avaluació de la preservació de la utilitat de diversos mètodes per a un cert nivell de risc.

Resumen

La cantidad de datos sobre la gente que se almacena continua creciendo. Muchos de estos datos hacen referencia a individuos concretos y, por tanto, pueden revelar información sensible. El almacenamiento y el procesado de estos datos en entornos locales presenta algunos inconvenientes, pero el uso de la nube para almacenarlos y procesarlos supone una amenaza para la privacidad y la seguridad. En esta tesis afrontamos los problemas derivados del uso de nubes no confiables. En particular, queremos que la nube pueda hacer dos operaciones básicas con datos sensibles no cifrados de forma práctica y que preserve la privacidad: productos escalares y productos de matrices. Estas operaciones son útiles para hacer diferentes tipos de análisis de datos como son el cálculo de correlaciones entre atributos y el cálculo de tablas de contingencia. Específicamente, proponemos diferentes protocolos seguros para externalizar a múltiples nubes el cálculo de varios análisis multivariantes sobre datos nominales (basados en frecuencias y basados en semántica). En general, estos análisis son complejos y lo son aún más sobre datos nominales (por ejemplo, datos textuales, datos no ordinales) porque las operaciones aritméticas estándar no pueden usarse.

Nuestros protocolos nos permiten utilizar la nube no sólo para almacenar datos sensibles no cifrados sino también para procesarlos. Consideramos dos tipos de nubes semihonradas: nubes que no comparten información entre ellas y nubes que pueden colaborar compartiendo información. Aparte de analizar la seguridad de los protocolos propuestos, también evaluamos su rendimiento respecto a un método de referencia consistente en descargar los datos y hacer el procesado en local. Nuestros protocolos han sido diseñados para externalizar a las nubes la mayor cantidad posible de carga de trabajo. De esta manera podemos conservar la rentabilidad del cálculo en la nube y, al mismo tiempo, asegurarnos de que los datos externalizados permanecen fragmentados y, por tanto, mantienen la privacidad frente a la nube. Las evaluaciones experimentales con datos categóricos que hemos realizado sobre los servicios de nube de Amazon

muestran que, con nuestros protocolos, el administrador de los datos puede ahorrarse más del 99.999% del tiempo de ejecución en los cálculos más exigentes.

Presentamos asimismo una metodología para comparar métodos de control de la revelación estadística (CRE) para microdatos en términos del compromiso entre riesgo y utilidad. Los estudios comparativos anteriores empiezan habitualmente seleccionando algunos valores de los parámetros para un conjunto de métodos CRE y luego evalúan el riesgo de revelación y la pérdida de información que ocasionan estos parámetros. Aquí comenzamos por seleccionar un nivel de riesgo (resp. utilidad) y buscamos los valores de los parámetros que hacen falta para obtener dicho nivel de riesgo (resp. utilidad) con cada uno de los métodos CRE que se evalúan. Finalmente, una vez conseguido un nivel de riesgo (resp. utilidad) equivalente en los diferentes métodos CRE, evaluamos su utilidad (resp. riesgo), lo que permite ordenarlos con respecto a la utilidad (resp. riesgo). Esta ordenación depende del nivel de riesgo (resp. utilidad) y del fichero de datos original. La novedad de esta comparación no se limita a la metodología descrita previamente. También proponemos medidas de utilidad y riesgo diferentes de las habituales. Adicionalmente, presentamos resultados experimentales de nuestra metodología en la evaluación de la preservación de la utilidad de varios métodos para un cierto nivel de riesgo.

Chapter 1

Introduction

1.1 Motivation

In 2017, DOMO estimated that 90 percent of the world's data had been created in the previous two years [26]. Moreover, many data analysts expect that the digital universe will be 40 times bigger by 2020 [43]. This big amount of person-specific and sensitive data arrives from disparate sources such as social networking sites, mobile phone applications and electronic medical record systems. The use of big data offers remarkable opportunities. For example, in a healthcare context, big data can be used to refine health policies, which is beneficial for individuals and for the society as a whole. However, at the same time, the privacy of the subjects to whom the data refers to needs to be guaranteed. The data must be protected against attacks and data leakages (data protection), e.g. unauthorized people cannot have access to sensitive data. At the same time, it should not be possible to re-identify any individual in the published data even when other external or publicly available data are integrated (data anonymity). Additionally, the access to the released data should not allow an attacker to increase his knowledge about confidential information related to any specific individual (data condentiality). For instance, the Big Data Value Association [8] recognizes data protection and anonymization as one of the main priorities for research and innovation. Moreover, they also identify the need of efficient mechanisms for data storage and processing, and they suggest the joint development of hardware and software for cloud data platforms.

The cloud is fast becoming a new normal and suitable strategy in the big

data context. In fact, the cloud is often the only possible strategy due to the costs (software, hardware, energy, maintenance) associated to the storage and processing of big data. The 2017 State of the Cloud Survey [56] estimated that 95 percent of enterprises had a cloud strategy. Nevertheless, even if companies run most of their workload in the cloud, around the 79 percent of their total workload, a remaining 21 percent, runs locally. This local workload may be traced back to the reluctance of data controllers to entrust their sensitive data to the cloud due to security and privacy concerns [4]. For instance, the 2017 State of the Cloud Survey shows that 25 percent of respondents cite security as a major concern. The problem is not only that cloud service providers (CSPs) may read, use or even sell the data outsourced by their customers; but also that they may suffer attacks and data leakages that can compromise data confidentiality. For instance, Ristenpart et al. [57] show that co-resident virtual machines (VM) can give rise to certain security vulnerabilities: if the attacker becomes a customer of the cloud and obtains a VM, he can use different information leakage attacks on the shared physical resources to gain access to the victim's (sensitive) information.

Although data collection has become easier and more affordable than ever before, releasing data for secondary use (that is, for a purpose other than the one that triggered the data collection) remains very important: in most cases, researchers cannot afford collecting themselves the data they need. However, when the data released for secondary use refer to individuals, households or companies, the privacy of the data subjects must be taken into account.

A great variety of statistical disclosure control (SDC) methods, which aim at releasing data that preserve their statistical validity while protecting the privacy of each data subject, are now available [40]. Since sensitive information can be inferred in many ways from the data releases, these masking methods are compulsory. Homer et al. [39] show that participants in genomic research studies may be identified from the publication of aggregated research results, including where an individual contributes in a mixture less than 0.1% of the total genomic DNA. Greveler et al. [37] show that the high-resolution energy consumption data which are transmitted by some smart meters to the utility company can be used to identify the TV shows and movies being watched in a target household. Coull et al. [19] show that certain types of web pages viewed by users can be deduced from metadata about network flows, even when server IP addresses are replaced with pseudonyms. Finally, Goljan and Fridrich [36] show how cameras can be identified from noise in the images they produce.

While there is a great diversity of SDC methods for microdata protection, all of them imply some level of data masking. The greater the amount of

masking, the greater are both privacy protection and information loss. Different SDC methods tackle the trade-off between privacy and utility in different ways. For example, in global recoding the level of information loss is set beforehand (the amount of coarsening of the categories of each attribute), whereas the disclosure risk is evaluated afterwards on the protected data set. In contrast, in k-anonymity [61] the risk of disclosure (the risk of record re-identification, in particular) is set beforehand, whereas the actual information loss results from the masking needed to attain the desired level of disclosure risk.

Although some general assertions about specific SDC methods/models can be made, comparing the latter regarding the privacy-utility trade-off is not straightforward. Let us illustrate this point with two well-known privacy models: differential privacy [31] and k-anonymity [61]. In terms of privacy protection, ϵ-differential privacy is regarded as stronger than k-anonymity. On the contrary, k-anonymity is regarded as more utility-preserving than ϵ-differential privacy. The practical value of these general statements is dubious. After all, by increasing ϵ we reduce the protection of differential privacy, and by increasing k we reduce the utility of k-anonymous data. An accurate comparison between SDC methods has to take into consideration both aspects of the privacy-utility trade-off.

1.2 Contributions

The main objective of this thesis is the design of protocols to outsource and process sensitive data in an untrusted cloud. That is, we aim at obtaining a privacy-preserving version of the original data which allows fast analyses of its contents while preserving the privacy of the subjects to whom the data refer to. We focus on the honest-but-curious security model. That is, the cloud runs the protocols as expected but tries to learn as much as possible about the underlying data. In particular, different clouds may even collude to reconstruct the entire data set. In case of collusions, anonymization techniques are required and, therefore, we need to assess the risk-utility trade-off among several masking methods.
Specifically, for numerical data:

1. We focus on secure scalar products on vertically partitioned data when CSPs are honest-but-curious and do not share information. We present two new non-cryptographic protocols and two variants of a known cryptographic protocol to compute the secure scalar product. After that, we

show how secure scalar products between pairs of clouds can be combined with computations involving a single cloud to perform data analyses such as correlations.

2. We present a new protocol that can resist information sharing between CSPs but assumes the CSPs have no side knowledge about the original data set; rather than computing individual scalar products, this protocol computes a matrix product $\mathbf{X^T X}$ of an original data set $\mathbf{X}$ of which the cloud only knows a masked version $\mathbf{Y}$. Just like the scalar products allowed computing the data set correlation matrix, so does the above matrix product; furthermore, the clouds can also be used to compute the means and the standard deviations of attributes in $\mathbf{X}$.

3. We propose another new sharing-resistant protocol to compute the matrix product $\mathbf{X^T X}$ that involves heavier computations but stays safe even if the CSPs have information on the statistical structure of the original data set $\mathbf{X}$.

For categorical data, we propose efficient protocols to securely compute statistical dependence analyses on split outsourced data for:

1. *Frequency-based tests.* They rely on the frequencies of attribute values. The contingency table associated to categorical attributes is the input of these tests. Therefore, the secure computation of the contingency table allows the calculation of all these tests, which are not computationally demanding. We adapt the two best protocols of the honest-but-curious non-sharing CSPs model to the computation of that table.

2. *Semantic-based tests.* They are the costliest ones and the ones that would benefit the most from outsourcing the computation to the cloud. We adapt the two best protocols of the honest-but-curious non-sharing CSPs model to the sample covariance matrix computation. Specifically, several semantic measures to quantify distances between nominal values are discussed and evaluated, both theoretically and in practice.

Finally, we present a new methodology to compare statistical disclosure control methods for microdata in terms of how they perform regarding the risk-utility trade-off. Previous comparative studies usually start by selecting some parameter values for a set of SDC methods and evaluate the disclosure risk and the information loss yielded by the methods for those parameterizations. In contrast, we start by setting a certain risk level (resp. utility preservation level)

and then we find which parameter values are needed to attain that risk (resp. utility) under different SDC methods. Finally, once we have achieved an equivalent risk (resp. utility) level across methods, we evaluate the utility (resp. the risk) provided by each method, in order to rank methods according to their utility preservation (resp. disclosure protection), given a certain level of risk (resp. utility) and a certain original data set.

1.3 Document structure

The remaining of this document is organized in the following chapters:

Chapter 2 provides the background on cryptographic and non-cryptographic techniques that will be used for the secure processing of outsourced data in an untrusted cloud. Moreover, masking methods and risk and utility measures are also discussed.

Chapter 3 explores new non-cryptographic protocols for the scalar product on split data and also consider cryptographic protocols to compute on split data in honest-but-curious non-sharing clouds.

Chapter 4 introduces two non-cryptographic protocols for the matrix product in a honest-but-curious sharing clouds. With respect to the previous chapter, the non-sharing assumption is relaxed. As a result, we need to introduce the data anonymization techniques. Moreover, the procotols become more suitable for matrix products rather than for scalar products.

Chapter 5 presents efficient protocols to securely compute statistical dependence analyses on split outsourced data for a variety of methods, encompassing frequency-based and semantic-based tests.

Chapter 6 proposes a new framework based on general empirical measures of utility and risk to compare the risk-utility trade-off of several statistical disclosure control methods.

Chapter 7 summarizes the main contributions of this thesis and presents some lines of future research.

Chapter 2

Background

In this chapter, we provide the background on cryptographic and non-cryptographic techniques that will be used for the secure processing outsourced data in an untrusted cloud. Moreover, masking methods and risk and utility measures are also discussed. The chapter is organized as follows. Section 2.1 introduces cloud computing and its main models. Section 2.2 presents the CLARUS architecture and the security assumptions considered in the design of our protocols. Section 2.3 reviews the literature on data splitting. Sections 2.4 surveys multivariate analyses on nominal data. Frequency-based tests and semantic-based tests are reviewed. Section 2.5 recalls the main semantic distance measures. Section 2.6 reviews several statistical disclosure control methods which will be used in Chapter 6. In particular, we focus our attention on correlated noise addition, multiplicative noise, multivariate microaggregation and rank swapping. Section 2.7 introduces the permutation paradigm and the permutation distance (a measure used in risk assessment). Section 2.8 provides an overview on multivariate utility measures.

2.1 Cloud computing

Cloud computing, often referred to as simply "the cloud", is the delivery of on-demand computing resources over the internet on a pay-for-use basis [41]. Depending on the needs, the cloud can offer to its users different models which guarantee scalable flexibility on services that it can provide. Usually, three models are identified: Software as a Service (SaaS), Platform as a Service (PaaS)

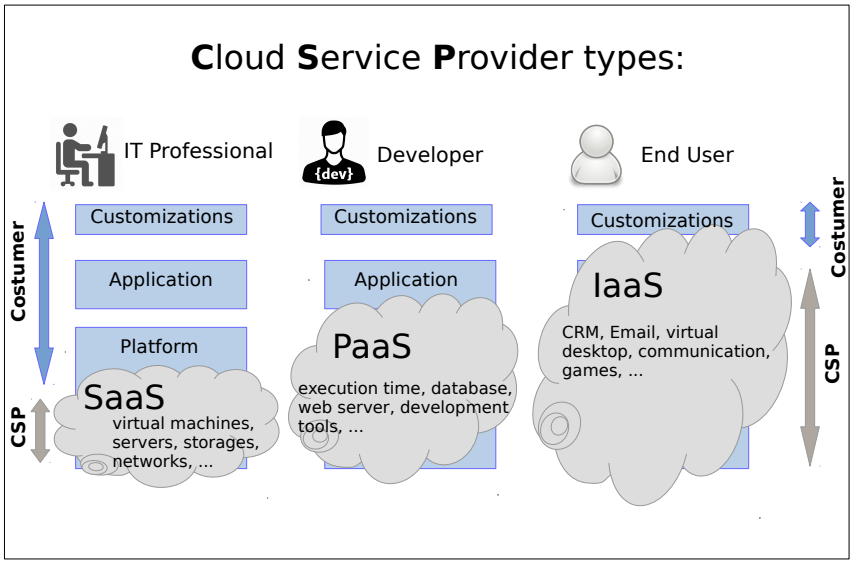

Figure 2.1: Cloud service provider types: SaaS, PaaS, IaaS.

and Infrastructure as a Service (IaaS). Figure 2.1 depicts the three models and the costumers' accessibility to the platform depending on the model.

In SaaS model, the cloud allows users to run applications on distant computers via internet and a web browser. In other words, users can log in the application software and databases owned by the cloud but they cannot manage the cloud platform where the application runs. This allows users to access the prefixed application from any computers and, therefore, this application does not need to be installed on the users' computers. Examples of SaaS are when users update their Facebook status or check their bank balance.

PaaS provides a development environment where users can develop and run their software solutions on a cloud platform, but they cannot manage or control the underlying cloud infrastructure, such as network, servers, operating systems and storage. In this way, users do not have to buy and manage the hardware, software, provisioning, and hosting needed for their development environment. Moreover, the service can be dynamically scaled to usage needs.

In IaaS model, the cloud provides users with computing resources as servers,

networking or storage. Users do not manage or control the underlying cloud infrastructure but have control over operating systems, storage, and deploying and running arbitrary software. This latter model presents all the advantages of the previous ones and, additionally, can cover all those situations where users build the infrastructure required to computation-intensive workloads and applications to big data and analytics.

In this thesis, we are particularly interested in IaaS due to the design of protocols to outsource and process sensitive *big* data on CSPs. In fact, local storage and processing of such big data is often unfeasible for the data controllers because of the associated costs (software, hardware, energy, maintenance). In statistical analyses, for instance, measuring the correlation between (just) two categorical attributes in a data set containing one million records may require computing and storing matrices of size one million times one million [55]. If matrix values are as short as 4-byte integers (real numbers would take more), then storing one such matrix takes nearly 4 terabytes.

2.2 System architecture, assumptions and security models

The scenario we consider involves the three entities in Figure 2.2: the data controller, the CLARUS proxy and the CSPs. The data controller owns the data that need to be outsourced to the CSP. CLARUS is a proxy located in a domain trusted by the data controller that implements security and privacy-enabling features towards the cloud service provider so that i) the CSP only receives privacy-protected versions of the controller's (or the controller's users') data, ii) CLARUS makes transparent the access to such data to the controller's users (by adapting their queries and reconstructing the results retrieved from the cloud) and iii) it remains possible for the users to leverage the cloud to perform accurate computations on the outsourced data without downloading them.

The *raison d'être* of such an architecture is to outsource as much storage and computation as possible to the cloud in a privacy-preserving manner, and keep the computational load of the CLARUS proxy (sitting in the controller's premises) as low as possible. The privacy-preserving calculation protocols implemented by CLARUS should, thus, follow this principle. The underlying assumption is that computing in the cloud is cheaper and/or otherwise more convenient than computing in the controller's local facilities. In order to eval-

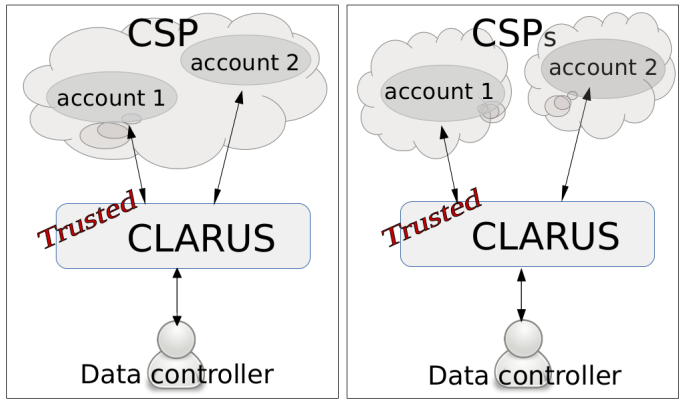

Figure 2.2: System architecture: CLARUS sits in the controller's trusted premises while the CSPs are untrusted.

uate the performance of a protocol, we will focus on how much it reduces the work to be done by the CLARUS proxy compared to the trivial alternative of CLARUS downloading the entire data set, unprotecting it and computing locally on the downloaded unprotected data.

CLARUS may outsource data either in separate cloud accounts within the same CSP (see left side of Figure 2.2) or to different clouds, each one run by a different CSP (see right side of Figure 2.2). The CSPs that receive privacy-protected versions of the controller's data are not trusted and, hence, they should not be given access to the entire original data set. Therefore, the CSP just sees fragments of the original data set or an anonymized version of it. More specifically, we consider three different security models for the CSPs, depending on their information sharing and their level of background knowledge:

Honest-but-curious non-sharing CSPs. The CSPs honestly fulfill their role in the protocols, and they do not share information with each other (perhaps because they do not even know each other). In particular, they do not pool together the data fragments they hold. However, each CSP may be curious to infer and analyze the data it stores and the message flows received during the protocol, in order to acquire additional information. This model is common in the cloud computing literature, e.g. see [13].

Honest-but-curious sharing CSPs without background knowledge. We relax here

the previous assumption about non-sharing: while honestly following the protocol, CSPs may collude by sharing information with each other. However, we assume they do not have any initial side knowledge about the original data set.

Honest-but-curious sharing CSPs with side knowledge. This is the most demanding model we consider. In addition to sharing information with each other, the CSPs have background knowledge on the statistical structure of the original data set. Nevertheless, they honestly perform their roles in the protocols.

We do not consider malicious CSPs that may deviate from the protocols because they would not be very useful for computation outsourcing in our context. The data controller/owner using the CLARUS proxy is assumed to rent the CSPs to get help from them. Hence, it would be pointless if it took CLARUS more effort to check that malicious CSPs carry out the computations correctly than to download the data and do the computations locally.

2.3 Data splitting

Data splitting (or data partitioning or fragmentation) means dividing an original sensitive data set into fragments and storing each fragment in a different site, in such a way that the fragment in any site considered in isolation is no longer sensitive. Data splitting has long been used as a privacy-preserving technique [71, 73, 18].

Queries on split data can often be answered much more efficiently than queries on encrypted data (see [1]). In data splitting, the most challenging step is usually to efficiently compute on the fragmented data when the computations involve more than one fragment: in this case, the clouds may need to exchange (part of) their respective fragments, but none of them ought to reveal its own private information. Specifically, challenging tasks in computing on split/distributed data are data mining [75] and data correlation [79]. The literature on parallel processing for statistical computation has partly treated this topic: the way to combine partial results obtained from independent processors may provide guidance on how to treat distributed data. On the other hand, privacy-preserving data mining on partitioned data can be of use too, as its main objective is to mine data owned by different parties who are willing to collaborate in order to get better results, but who do not want or cannot share their raw original data.

Splitting can be horizontal (each fragment contains the values of all attributes but only for a subset of subjects), vertical (each fragment contains the values of a subset of attributes for all subjects), or mixed (each fragment contains the values of a subset of attributes for a subset of subjects). Whereas vertical and mixed splitting preserve privacy by decomposition, horizontal splitting does not do so because all the information on the same individual subject is stored together [1]. Hence, if preserving anonymity and preventing attribute disclosure is a key concern, horizontal partitioning is not suitable.

Vertical splitting methods were proposed in [1] and [32] to ensure confidentiality. Both methods rely on predefined constraints describing risky attribute combinations. An example of a risky pair is passport number and disease, whereas blood pressure and disease is generally a safe pair. The goal of both methods is to partition the original data set into two vertical fragments in such a way that, if some risky attributes are stored together in a fragment, some of them need to be encoded/encrypted. In [15] the authors illustrate a similar approach to [1] and [32] but using an arbitrary number of non-linkable data fragments, which can be stored at an arbitrary number of providers. Also, [16] presents a solution for splitting data into two vertical fragments without requiring the use of encryption, but rather using a trusted party (the owner) to store a portion of the data and perform part of the computation.

Regarding mixed splitting, in general, it does not improve on vertical splitting in terms of privacy and it complicates distributed computation (no local computation on entire attributes is possible any more at the locations holding fragments).

In summary, vertical splitting is the most suitable splitting method for privacy, since in statistical databases it is the *joint* distribution of several attributes that is sensitive (because it may lead to re-identification of the subject behind a tuple of values). Attributes in isolation (or even groups of attributes whose value combinations are very common) are not sensitive. For example, if a fragment consists of the values of an attribute "Diagnosis", then just knowing a list of diagnoses is clearly useless to an intruder, because he cannot associate it to the corresponding subjects (it is nearly as useless as seeing a list of diseases and their frequencies in a manual of medicine).

2.3.1 Computing on vertically partitioned data

A privacy breach occurs when individuals are re-identified in a data set containing confidential attributes. Re-identification may be enabled by single attributes (like SS numbers) or by sets of attributes each of which does not uniquely iden-

tify the individual to whom the record corresponds to but whose combination may (e.g., job+age+place of birth). The former type of attributes are known as *identifiers*, whereas the latter are called *quasi-identifiers*.

If a data set is to be released without splitting, identifiers should be removed or at the very least encrypted or pseudonymized, whereas quasi-identifiers should be masked (by either perturbing them or reducing their detail); see [40] for details. Thus, information is lost in this process (removed identifiers, reduction of detail or perturbation of quasi-identifiers).

However, if the data set is protected by splitting it among several untrusted clouds, no information loss needs to be incurred. Under *vertical data splitting*, identifiers may be fragmented (e.g. locations can be split into longitude and latitude, SS numbers may be fragmented into several subgroups of digits, etc.) and the set of quasi-identifier attributes may also be fragmented into several disjoint subsets; this fragmentation of identifiers and quasi-identifiers should be such that no re-identification is feasible from a single fragment [62]. Then data fragments (either split values or attribute subsets) are stored in separate locations (i.e., different CSPs or cloud accounts).

2.3.2 Additions, updates and processing in vertically split data vs other protection options

Since re-identification is not possible from single fragments, these can be stored in clear form. Thanks to this feature, data splitting allows fast additions and updates of the outsourced data, provided that the local proxy in charge of orchestrating the splitting process stores the criteria employed to fragment the data and the locations at which fragments were stored [62].

In contrast, when protecting data by masking instead of splitting, adding or updating a value typically requires re-encrypting or re-masking the entire data set or in any case larger chunks than just the value that has been added/updated. For example, if the data are protected according to the k-anonymity privacy model by generalizing quasi-identifiers [60] or microaggregating them [25], adding or updating an original record may require k-anonymizing again the entire data set. Furthermore, the CSP storing the anonymized data set might be able to infer the value of some original records by comparing the successive anonymized versions of the data set; thus, splitting may offer more protection than masking as long as the various CSPs holding fragments do not collude.

Similar issues arise if protecting data uses functionality-preserving encryption. In order to update a single value, the data controller has to (1) retrieve the entire data set from the cloud, (2) decrypt, update and re-encrypt it and,

(3) send back the entire encrypted data to the cloud. Therefore, data splitting is significantly more efficient for additions and updates than encryption methods, such as searchable or homomorphic encryption [62]. Regarding data processing on the cloud, even though searchable and homomorphic encryption allow performing some operations on ciphertext [29], computing on encrypted data is extremely limited and costly [53], and it requires careful management of encryption keys. Outsourced data processing can be performed much more efficiently on split data: although each CSP only holds partial data, these are in the clear. Admittedly, both the orchestration of the split calculations to be done and the aggregation of the partial results retrieved from each CSPs should be done by the local proxy; therefore, computation protocols should be designed to minimize both the data that need to be locally stored and the amount of local calculations needed to obtain the final result.

In vertical splitting, analyses that involve single attributes (e.g., mean, variance) or attributes stored within a single data fragment are fast and straightforward: the cloud storing the fragment can independently compute and send the output of the analysis to the local proxy. However, analyses assessing the relationship (e.g., correlation) between attributes may involve data fragments stored in different locations, and thus, communication between several clouds. As shown in [12, 21], performing calculations on data split among multiple clouds can be decomposed into several secure scalar products to be conducted between pairs of clouds. Scalar products can be made secure with or without cryptography. Cryptographic approaches use a variety of techniques; for instance, the protocol in [35] involves homomorphic encryption. Non-cryptographic approaches are rather based on modifying the data before sharing them, in such a way that the original data cannot be inferred from the shared data but the final results are preserved (e.g., [18], [45]).

2.4 Multivariate analyses on categorical data

When data are numerical, multivariate statistical analyses such as correlations, covariances, regressions and classifications are easy to perform and can be computed using standard arithmetic operators. In contrast, analyzing categorical data is more difficult. Especially challenging are nominal categorical attributes, whose values are noun phrases describing jobs, interests or conditions, etc., because they are textual and non-ordinal and, therefore, require specific analytical methods.

Table 2.1: Source: "Preliminary report: Finding from the aspirin component of the ongoing Physicians' Health Study." *New Engl. J. Med.* 318: 262-264 (1988).

	Myocardial Infarction		
	Fatal Attack	Nonfatal Attack	No attack
Placebo	18	171	10845
Aspirin	5	99	10933

2.4.1 Frequency-based tests

The simplest methods rely on the frequencies of attribute values. Well-known frequency-based procedures to measure the statistical dependence between two categorical attributes are the χ^2-test of independence [2], ANOVA [33] and Cramer's V [2]. These tests use the contingency tables associated to categorical attributes as input for a linear regression analysis.

In particular, a *contingency table* (or cross-classification table) is a table containing the (multivariate) frequency distributions of the nominal attributes. Let $\mathbf{a}$ and $\mathbf{b}$ denote two nominal attributes, $\mathbf{a}$ with h categories $c_1(\mathbf{a}), \ldots, c_h(\mathbf{a})$ and $\mathbf{b}$ with k categories $c_1(\mathbf{b}), \ldots, c_k(\mathbf{b})$. The contingency table has h rows and k columns displaying the sample frequency counts of the $h \times k$ category combinations.

Table 2.1 shows a 2×3 contingency table from a report on the relationship between aspirin use and heart attacks by the Physicians' Health Study Research Group at Harvard Medical School. In this case, $\mathbf{a}$ represents the pharmaceutical drugs (aspirin or placebo) given to each patient and $\mathbf{b}$ represents the kind of myocardial infarction attacks got from each patient. The length of $\mathbf{a}$ and $\mathbf{b}$ is the number of participants in the research.

2.4.2 Semantic-based tests

Even though frequency-based methods can measure some degree of statistical dependence, they only consider the similarities between the distributions of categorical labels; therefore, they fail to capture the *semantic* similarity among the categories themselves, which is the means by which human beings create, understand and manage nominal data.

To tackle this issue, semantically grounded methods have been recently proposed. In [23], nominal values are associated to numbers that capture both their semantic and distributional features; from these, semantically coherent

variances [65] and correlations can be computed based on standard numerical methods. In [69], a more accurate way to measure the dependence between categorical attributes is proposed. Specifically, the *distance covariance* and the *distance correlation* measures are proposed as alternatives to the numerical covariance and correlation, respectively. The numerical covariance requires the values of attributes to be totally ordered, and it measures dependence by checking whether greater values of one attribute correspond to greater values of the other attribute, and smaller values to smaller values. This assumption does not work for non-ordinal (i.e., nominal) categorical attributes, which lack total order. In contrast, the distance covariance quantifies to what extent the two attributes are independently dispersed, where dispersion is measured according to the pairwise distances between all pairs of values of each attribute. Unlike frequency-based approaches, pairwise distances can capture the *semantics* inherent to categorical values. To do so, the pairwise distance can be calculated using similarity/distance measures [59], that quantify how similar are the meanings of the concepts associated to the categorical values, based on the semantic evidences gathered from one or several knowledge sources (e.g., ontologies, corpora).

Formally, let $\mathbf{x^1} = (x_1^1, \ldots, x_n^1)^T$ and $\mathbf{x^2} = (x_1^2, \ldots, x_n^2)^T$ be vectors of values of two nominal attributes. The calculation of the distance covariance requires measuring the pairwise semantic distance between the nominal values of each attribute. The *semantic-distance matrix* of $\mathbf{x^1}$ is given by

$$\mathbf{X^1} = [x_{ij}^1]_{i,j \leq n}, \tag{2.1}$$

where $x_{ij}^1 = |x_i^1 - x_j^1|$ are the semantic distances between two nominal values of the same attribute $\mathbf{x^j}$ (see Section 2.5 for more details). Similarly, we define $\mathbf{X^2} = [x_{ij}^2]_{i,j \leq n}$, where $x_{ij}^2 = |x_i^2 - x_j^2|$. Then, the *double-centered matrix* $\hat{\mathbf{X}}^\mathbf{1}$ is computed, whose elements are obtained as

$$\hat{X}_{kl}^1 = x_{kl}^1 - \bar{x}_{k\cdot}^1 - \bar{x}_{\cdot l}^1 + \bar{x}_{\cdot\cdot}^1 \quad \text{for } k, l = 1, \ldots, n, \tag{2.2}$$

and where

$$\bar{x}_{k\cdot}^1 = \frac{1}{n}\sum_{l=1}^{n} x_{kl}^1, \quad \bar{x}_{\cdot l}^1 = \frac{1}{n}\sum_{k=1}^{n} x_{kl}^1, \quad \bar{x}_{\cdot\cdot}^1 = \frac{1}{n^2}\sum_{k,l=1}^{n} x_{kl}^1. \tag{2.3}$$

Similarly, let us define $\hat{X}_{kl}^2 = x_{kl}^2 - \bar{x}_{k\cdot}^2 - \bar{x}_{\cdot l}^2 + \bar{x}_{\cdot\cdot}^2$ for $k, l = 1, \ldots, n$.

Definition 1. *The squared* semantic-distance covariance *is obtained as the arithmetic average of the products* $X^1_{kl}X^2_{kl}$*, that is,*

$$d\mathcal{V}_n^2(\mathbf{x^1}, \mathbf{x^2}) = \frac{1}{n^2}\sum_{k,l=1}^{n} X^1_{kl}X^2_{kl}, \tag{2.4}$$

and the squared semantic-distance variance *is obtained as*

$$d\mathcal{V}_n^2(\mathbf{x^1}) = d\mathcal{V}_n^2(\mathbf{x^1}, \mathbf{x^1}) = \frac{1}{n^2}\sum_{k,l=1}^{n} X^1_{kl}X^1_{kl}. \tag{2.5}$$

See [69] and [59] for details and justification of the above definition.

In general, if $\mathbf{X} = (\mathbf{x}^1, \ldots, \mathbf{x}^m)$ is a data set with m attributes $\mathbf{x}^j$, $j = 1, \ldots, m$, the distance covariance matrix $\hat{\mathbf{\Sigma}}$ of $\mathbf{X}$ is

$$\hat{\mathbf{\Sigma}} = \left(\begin{array}{c|c|c|c} d\mathcal{V}_n(\mathbf{x}^1) & d\mathcal{V}_n(\mathbf{x}^1, \mathbf{x}^2) & \cdots & d\mathcal{V}_n(\mathbf{x}^1, \mathbf{x}^m) \\ \hline d\mathcal{V}_n(\mathbf{x}^2, \mathbf{x}^1) & d\mathcal{V}_n(\mathbf{x}^2) & \cdots & d\mathcal{V}_n(\mathbf{x}^2, \mathbf{x}^m) \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline d\mathcal{V}_n(\mathbf{x}^m, \mathbf{x}^1) & d\mathcal{V}_n(\mathbf{x}^m, \mathbf{x}^2) & \cdots & d\mathcal{V}_n(\mathbf{x}^m) \end{array}\right).$$

The method we have just recalled accurately captures the (semantic) dependence between nominal attributes; however, its main drawback is the cost. Due to the need to compute pairwise distance matrices, the calculation of the distance covariance between two nominal attributes has quadratic cost, both in time and storage. Moreover, as we discuss in Section 2.5 below, evaluating the semantic distance between each pair of nominal attribute values adds a significant burden.

Therefore, for large data sets, the only feasible alternative for data practitioners may be to outsource the calculation to the cloud. However, when the data are sensitive (which is often the case because most of the personal attributes gathered on individuals are nominal [72]), outsourcing storage and calculation should be performed in a privacy-preserving way. This is precisely the main goal of this thesis.

2.5 Semantic distance calculation

The *semantic distance* quantifies the difference between the meaning of two nominal values. Semantic similarity/distance measures rely on the semantic

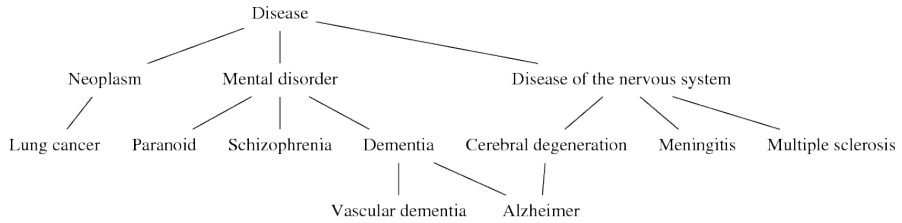

Figure 2.3: Ontology extract for the "Diagnosis" concept

evidences gathered from knowledge bases, such as ontologies, which taxonomically structure the concepts of a domain of knowledge [7]. Formally, an *ontology* $\mathcal{O}$ is composed, at least, of a set of concepts or classes C organized in a directed acyclic graph (due to multiple inheritance) by means of is-a ($c_i < c_j$) relationships [14], as shown in Figure 2.3.

Measuring the semantic distance in large ontologies can be costly. In this section we discuss the computational cost of some well-known measures by relying on the concepts introduced in the following definition.

Definition 2. *Let $S(\mathbf{X^a})$ be the set of subsumers (i.e., taxonomic ancestors) of the nominal values of attribute $\mathbf{X^a}$ mapped in an ontology $\mathcal{O}$. The* least common subsumer *of $\mathbf{X^a}$, denoted by $LCS(\mathbf{X^a})$, is the most specific concept in $S(\mathbf{X^a})$. Formally,*

$$S(\mathbf{X^a}) = \{c_i \in \mathcal{O} | \forall c_j \in \mathbf{X^a} : c_j \leq c_i\};$$

$$LCS(\mathbf{X^a}) = \{c \in S(\mathbf{X^a}) | \forall c_i \in S(\mathbf{X^a}) : c \leq c_i\}.$$

The *semantic distance* is defined as a function $d_s : \mathcal{O} \times \mathcal{O} \to \mathbb{R}$ mapping a pair of concepts (corresponding to nominal values) to a real number that quantifies the difference between their meanings. According to the calculation principle employed, ontology-based measures can be divided in three families:

1. Edge-counting measures.

2. Feature-based measures.

3. Measures based on information content.

Edge-counting measures
They estimate the semantic distance between concept pairs as a function of the length of the *taxonomic path* connecting the two concepts in the ontology [52].

A well-known edge-counting measure was proposed by Wu and Palmer [78]:

$$d_{\mathrm{WP}}(c_1, c_2) = 1 - \frac{2 \times \mathrm{depth}(LCS(c_1, c_2))}{2 \times \mathrm{depth}(LCS(c_1, c_2)) + \mathrm{path}(c_1, LCS(c_1, c_2)) + \mathrm{path}(c_2, LCS(c_1, c_2))}, \tag{2.6}$$

where $LCS(c_1, c_2)$ is the most specific subsumer of c_1 and c_2 in the ontology; $\mathrm{depth}(LCS(c_1, c_2))$ is the number of nodes in the longest taxonomic path between the $LCS(c_1, c_2)$ and the node root of the taxonomy; and $\mathrm{path}(c_1, LCS(c_1, c_2))$ is the number of taxonomic edges in the shortest taxonomic path between the two concepts.

Simplicity is the main advantage of edge-counting measures. However, they present some shortcomings: 1) if they are applied to ontologies incorporating multiple taxonomical inheritance, several taxonomical paths are not taken into account, and 2) by considering only the paths (i.e., subsumers) between the concepts, much of the taxonomical knowledge explicitly modeled in the ontology is ignored.

Assuming that concepts in the ontology are linked with their ancestors through pointers, in the worst case (comparing the two most specific concepts in the ontology that have the root node as LCS), obtaining the $LCS(c_1, c_2)$ requires running through the longest path in the taxonomy, i.e., twice the taxonomy depth D. Therefore, it takes $O(D)$ cost to compute Equation (2.6).

Feature-based measures
They consider the degree of overlap between the sets of ontological features of the concepts to be compared. In [63], the authors suggested measuring the semantic distance as a function of taxonomic features, i.e., as the ratio between the number of non-common taxonomic ancestors and the total number of ancestors of the two concepts:

$$d_{log\mathrm{SC}}(c_1, c_2) = \log_2 \left(1 + \frac{|S(c_1) \cup S(c_2)| - |S(c_1) \cap S(c_2)|}{|S(c_1) \cup S(c_2)|} \right), \tag{2.7}$$

where $S(c_i)$ is the set of taxonomic subsumers of the concept c_i, for $i = 1, 2$. Due to the additional knowledge feature-based measures take into account (i.e.,

multiple direct ancestors in case of multiple inheritance), they tend to be more accurate than edge-counting measures [63].

If S is the maximum number of ancestors that a concept can have in the ontology, computing Equation (2.7) takes $O(S)$ cost. Notice that, for ontologies without multiple inheritance, this cost is the same as the one of edge-counting measures.

Measures based on information content

They measure the semantic distance between two concepts as the inverse of the amount of information they share in the ontology, which is represented by their LCS [54]. In particular, Lin [48] proposed as a measure the inverse of the ratio between the information content of the LCS of the concepts and the sum of the information content of each concept.

$$d_{\text{lin}}(c_1, c_2) = 1 - \frac{IC(LCS(c_1, c_2))}{IC(c_1) + IC(c_2)}. \tag{2.8}$$

In [64], $IC(c)$ is intrinsically estimated within the ontology as the normalized ratio between the number of leaves (i.e., terminal hyponyms) under concept c in the taxonomy and the number of subsumers of c:

$$IC(c) = -\log\left(\frac{\frac{|\text{leaves}(c)|}{|S(c)|} + 1}{|\text{max\_leaves} + 1|}\right). \tag{2.9}$$

Thanks to IC-based measures exploiting the largest amount of ontological evidence (i.e., ancestors and leaves), they achieve better accuracy than edge-counting and feature-based measures [6].

Equation (2.8) requires computing the LCS of the two concepts, plus the ICs of the LCS and the concepts. Like in edge-counting measures, computing the LCS has a worst-case complexity $O(D)$. On the other hand, Equation (2.9) requires obtaining all the possible concepts connected to c, either subsumers of hyponyms; hence, in the worst case (i.e., when c is the root node, which subsumes all the concepts in the ontology), the IC computation takes $O(C)$ cost, where C is the total number of concepts in the taxonomy. In conclusion, Equation (2.8) has $O(C + D)$ computational cost. Thus, IC-based measures are not only the most accurate but also the costliest.

Table 2.2: Example of correlated noise addition method with $\gamma = 0.2$.

Original data (+ noise)			Masked data		
Age	Height	Income	Age	Height	Income
55 -1	1.80 -4	2000 -381	54	1.76	1619
44 -3	1.60 -9	1100 +252	41	1.51	1352
32 +0	1.83 +0	1500 -410	32	1.83	1090
56 -11	1.78 -11	500 +59	56	1.67	959
40 +4	1.56 +6	750 +76	40	1.62	826
77 +5	1.70 -2	1350 +258	77	1.68	1608
41 -4	1.80 -5	600 -10	41	1.80	590
13 -10	1.71 +1	400 -269	13	1.72	131

2.6 Statistical disclosure control

Statistical disclosure control techniques reduce the risk of disclosing information on individuals, enterprises or other organisations [40]. These methods mask the original data, reducing the risk of disclosure, but, as a side effect, they reduce the utility of the published data. The real issue is to find a satisfactory balance in the risk-utility trade-off of the protected released data.

Masking methods can be: (i) *non-pertubative* (they do not distort data but instead produce partial suppressions or reductions of details in the original data set) and (ii) *perturbative* (they distort the data before publication and they ensure that the statistics computed on the perturbed data set do not differ significantly from the statistics that could be computed on the original data set). As an alternative to masking methods, we can generate *synthetic data* that preserve some statistical properties of the original data set. The latter approach has issues related to data utility. In fact, only the statistical properties explicitly selected by the data protector will be preserved.

In the following sections, we describe those methods that will be used in Chapter 6, namely correlated noise addition, multiplicative noise, multivariate microaggregation and rank swapping. We will call $\mathbf{x} = (x^1, \ldots, x^m)$ a record of an original data set $\mathbf{X}$ with attributes $X^1, \ldots, X^m$ and $\mathbf{y} = (y^1, \ldots, y^m)$ a record from a related anonymized data set $\mathbf{Y}$ with attributes $Y^1, \ldots, Y^m$.

Table 2.3: Example of multiplicative noise method with $s = 0.1$.

Original data			Masked data		
Age	Height	Income	Age	Height	Income
55 * 1.1	1.80 * 1.07	2000 * 1.16	63	1.88	2166
44 * 0.87	1.60 * 0.95	1100 * 0.95	39	1.60	1040
32 * 1.03	1.83 * 0.92	1500 * 0.97	33	1.71	1395
56 * 0.91	1.78 * 0.95	500 * 0.97	63	1.72	891
40 * 0.99	1.56 * 1.1	750 * 1.09	36	1.74	839
77 * 0.91	1.70 * 0.92	1350 * 0.85	68	1.64	1135
41 * 1.08	1.80 * 1.04	600 * 1.07	50	1.88	685
13 * 0.99	1.71 * 0.93	400 * 0.94	22	1.65	500

2.6.1 Correlated noise addition

Correlated noise addition [40] preserves means and correlation coefficients, i.e. the masked version has the same means and correlation coefficients as the original data set $\mathbf{X}$. This is achieved by adding multivariate normally distributed noise to the original records in the collected data set, that is

$$\mathbf{Y} = \mathbf{X} + N(\mathbf{0}, \gamma\mathbf{\Sigma}),$$

where $\mathbf{\Sigma}$ is the covariance matrix of $\mathbf{X}$, γ is an input parameter and $N(\mathbf{0}, \gamma\mathbf{\Sigma})$ is the multivariate normal distribution with null matrix as mean and Σ as covariance matrix. Note that the covariance matrix of $\mathbf{Y}$ is proportional to the covariance matrix of $\mathbf{X}$. Table 2.2 shows correlated noise addition method applied to a data set of 3 attributes and 8 records. The parameter γ takes value 0.2 in the masking.

The most prominent shortcomings of this method are that it does not preserve the univariate distributions of the original data and that it cannot be applied to discrete variables due to the structure of the transformations.

2.6.2 Multiplicative noise

Multiplicative noise has advantages with respect to noise addition. The use of a noise with constant variance leads to different levels of protection. On one side, small values are strongly perturbed and, on other side, large values are weakly perturbed. For instance, if we consider a business data set, then the

Table 2.4: Example of multivariate microaggregation method with $k = 3$.

Original data ($*$ noise)			Masked data		
Age	Height	Income	Age	Height	Income
55	1.80	2000	53	1.78	1617
44	1.60	1100	53	1.78	1617
32	1.83	1500	53	1.78	1617
67	1.78	900	43	1.70	750
36	1.56	750	43	1.70	750
72	1.70	1350	43	1.70	750
45	1.85	600	43	1.70	750
23	1.71	400	43	1.70	750

large enterprises, which are much easier to re-identify than the smaller ones, have their confidential data only slightly perturbed.

We present here Höhne's variant ([38] and Ch. 3 of [40]) which preserves the first- and second-order moments. In a first step, each attribute value $x_j^i \in \mathbf{X}$ is multiplied by $1 \pm N(0, s)$, where s is an input parameter. In particular, the formula is as follows,

$$y_j^i = x_j^i(1 \pm N(0, s)),$$

where $N(0, s)$ is the normal distribution with mean 0 and standard deviation s.

Then, the following transformation is applied to preserve the first and second-order moments,

$$\hat{Y}^i = \frac{\sigma_{X^i}}{\sigma_{Y^i}}(Y^i - \mu_{Y^i}) + \mu_{X^i},$$

where μ_{X^i} and μ_{Y^i} are the average of the original and masked variables, and σ_{X^i} and σ_{Y^i} are the corresponding standard deviations, respectively.

Table 2.3 shows Höhne's method applied to a data set of 3 attributes and 8 records. The parameter s takes the value 0.1 in the masking. The noise added to the elements of the original data set is depicted in red.

2.6.3 Multivariate microaggregation

Univariate microaggregation proceeds attribute by attribute. For each attribute, we partition the values into groups (clusters) of cardinality equal or greater than

k. To minimize the information loss, the partition groups should be as homogeneous as possible. Then, we compute a group representative (*e.g.* the group centroid) and replace each of the values by the corresponding group representative.

Here, we follow the MDAV heuristic [25], which is a multivariate microaggregation method. MDAV proceeds in two steps: i) it generates clusters of k (or more) records that are distant to the center of the data set, and ii) it replaces each of the original records by the corresponding cluster centroid. The clustering step proceeds in the following way. First we compute $\bar{\mathbf{X}}$, the average record. Then we compute x_r, the farthest record to $\bar{\mathbf{X}}$, and x_s, the farthest record to x_r. To generate the clusters, we check the number of remaining records to cluster:

- If this number is greater than or equal to $3k$, we generate two clusters: one containing x_r and the $k-1$ records closest to it, and the other containing x_s and the $k-1$ records closest to it.
- If this number is between $2k$ and $3k-1$, we generate two clusters: one containing x_r and the $k-1$ records closest to it, and the other containing all the remaining records.
- If this number is below $2k$, we generate a single cluster with all the remaining records.

Finally, the clustered records are removed from X, and the process is repeated meanwhile there are remaining records.

Table 2.3 shows MDAV method applied to a data set of 3 attributes and 8 records. The parameter k takes value 3 in the masking. Records with the same color belong to the same cluster.

2.6.4 Rank swapping

Among the presented masking methods, rank swapping is the only univariate one. It works as follows. Independently for each attribute, this method swaps the attribute's values within a restricted range: the ranks of two swapped values cannot differ by more than $p\%$ of the total number of records, where p is an input parameter. More details about these methods can be found in [40].

Table 2.3 shows the rank swapping method applied to a data set of 3 attributes and 8 records. The parameter p takes value 20% in the masking. For instance, the elements with the same color are swapped between the two data sets.

Table 2.5: Example of rank swapping method with $p = 20\%$.

Original data		
Age	Height	Income
55	1.80	2000
44	1.60	1100
32	1.83	1500
67	1.78	900
36	1.56	750
72	1.70	1350
45	1.85	600
23	1.71	400

Masked data		
Age	Height	Income
55	1.78	1500
36	1.60	900
32	1.85	2000
72	1.80	1100
44	1.56	600
67	1.71	1350
45	1.83	750
23	1.70	400

2.7 Permutation paradigm and permutation distance

In [20], a permutation paradigm to model anonymization was proposed. Let $X = \{x_1, \ldots, x_n\}$ be the values taken by attribute X in the original data set. Let $Y = \{y_1, \ldots, y_n\}$ represent the anonymized version of X. Consider the attribute Z obtained using the following reverse-mapping procedure

For $i = 1$ to n

Compute $j = rank(y_i)$

Set $z_i = x_{(j)}$ (where $x_{(j)}$ is the value of X of rank j)

Endfor

We can now view the anonymization of X into Y as a permutation step to turn X into Z, plus a small noise addition to turn Z into Y. Note the noise addition must be necessarily small, because it cannot alter ranks: by construction the ranks of Y and Z are the same. If we perform the above procedure independently for all attributes of an original data set $\mathbf{X}$ and corresponding attributes of an anonymized data set $\mathbf{Y}$, we can say that anonymization can be decomposed into a permutation step to obtain a data set $\mathbf{Z}$ plus a (small) noise addition to obtain $\mathbf{Y}$ from $\mathbf{Z}$.

The permutation distance measures the dissimilarity between two records in terms of the ranks of the values of their attributes. Assume the original data set

$\mathbf{X}$ consists of m attributes $X^1, \ldots, X^m$ and the anonymized data set consists of corresponding attributes $Y^1, \ldots, Y^m$. Let $\mathbf{x} = (x^1, \ldots, x^m)$ be a record in $\mathbf{X}$ and $\mathbf{y} = (y^1, \ldots, y^m)$ be a record in $\mathbf{Y}$. The permutation distance between $\mathbf{x}$ and $\mathbf{y}$ is the maximum of the rank distances of the attributes:

$$d(\mathbf{x}, \mathbf{y}) = \max_{1 \leq i \leq m} |rank(x^i) - rank(y^i)|.$$

The permutation distance between records is used in [20] to conduct a record linkage between the original data set $\mathbf{X}$ and the anonymized data set $\mathbf{Y}$. In particular, records with minimal permutation distance are linked.

2.8 Utility measures

Utility measures are a key component to compare SDC methods. We introduce two utility measures that will be used in the empirical evaluation of the proposed methodology: the propensity scores [77] and the earth mover's distance [58].

Algorithm 1 shows a way to use the propensity scores as a utility measure.

Algorithm 1.

1. *Merge the original data set* $\mathbf{X}$ *and the anonymized data set* $\mathbf{Y}$ *and add a binary attribute* T *with value 1 for the anonymized records and 0 for the original records.*

2. *Regress* T *on the rest of attributes of the merged data set and call the adjusted attribute* $\hat{T}$*. Let the* propensity score $\hat{p}_i$ *of record* i *of the merged data set be the value of* $\hat{T}$ *for record* i*.*

3. *The utility can be considered high if the propensity scores of the anonymized and original records are similar. Hence, if the original and the anonymized data sets have the same number* n *of records, the following is a utility measure*

$$\mathcal{U}_{ps}(\mathbf{X}, \mathbf{Y}) = \frac{1}{2n} \sum_{i=1}^{2n} [\hat{p}_i - \frac{1}{2}]^2. \tag{2.10}$$

The value $\mathcal{U}_{ps}$ resulting from Equation (2.10) is close to zero if the propensity scores computed with the regression model for all records are similar (in which case they will be neither 0 nor 1, but close to 1/2). This situation means that the

original and the anonymized records cannot be distinguished by the regression model, and hence the utility of the anonymized data set is high (its records "look" like the original records). In contrast, if the adjusted propensity scores were exactly the original values of T, it would mean that the regression model can exactly tell the original from the anonymized records, so the utility of the latter is low; in this case, we would have n propensity scores 0 and n propensity scores 1, which would yield a large $\mathcal{U}_{ps}$. Obviously, propensity scores as a utility measure are very dependent on the accuracy of the regression model adjusted to the data: the more accurate the model, the more discriminating it is and the less likely are values of $\mathcal{U}_{ps}$ indicating good utility (close to 0).

Earth mover's distance (EMD) is a natural extension of the notion of distance between single elements to distance between sets, or distributions, of elements. Given two distributions, one can be seen as a mass of earth in the space and the other as a collection of holes in that same space. Then, the EMD measures the least amount of work needed to fill the holes with earth, i.e. the minimal cost needed to transform one distribution into another by moving distribution mass. Thus, the EMD distance can be used to evaluate the similarity between the distribution of the original data set and the distribution of the anonymized data set. Note here that measuring similarity amounts to measuring utility, because, the more similar the distribution of the anonymized data to the distribution of the original data, the more useful are the anonymized data.

Formally, we can group records in clusters and represent each cluster j by its mean and the fraction ω_j of records that belong to that cluster. Let the original data set $\mathbf{X}$ be clustered as $\{(t_1, \omega_{t_1}), \ldots, (t_h, \omega_{t_h})\}$, and the anonymized data set $\mathbf{Y}$ as $\{(q_1, \omega_{q_1}), \ldots, (q_k, \omega_{q_k})\}$. Let $D = (d_{ij})$ be the matrix of the distance between the h clusters of $\mathbf{X}$ and the k clusters of $\mathbf{Y}$, i.e. $d_{ij} = t_i - q_j$ (in the multivariate case, we take the Euclidean distance between cluster means). The problem is to find a flow $F = (f_{ij})$, with f_{ij} being the flow between t_i and q_j, that minimizes the overall cost under some constraints (see [58] for more details). Once the optimal flow F is found, the earth mover's distance is defined as the resulting work normalized by the total flow:

$$\mathcal{U}_{emd}(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i=1}^{h} \sum_{j=1}^{k} d_{ij} f_{ij}}{\sum_{i=1}^{h} \sum_{j=1}^{k} f_{ij}} \tag{2.11}$$

The greater $\mathcal{U}_{emd}$ is, the more different the distributions of $\mathbf{X}$ and $\mathbf{Y}$ are and hence the more utility has been lost in the anonymization process.

Chapter 3

Secure Scalar Product in Honest-but-curious Cloud

3.1 Introduction

With the advancement and spread of computation and communication technologies, the amount of data collected and stored by private and public sectors is constantly increasing. Storing and processing such huge amounts of information in local premises has become very problematic, due to soaring costs of software, hardware, energy and maintenance. In this context, the need to find a fast and cost-effective alternative emerges as a necessity. An attractive possibility for a data controller is to outsource storage and processing to a cloud [4]. Such outsourcing gives the data controller several benefits like elimination of infrastructure costs (no software/hardware investments needed), flexibility (storage and computing power can scale depending on business growth) and energy savings.

Unfortunately, storing and processing data in the cloud has also downsides related to security and privacy. A lot of the information collected is personally identifiable and therefore sensitive. Neither the data controller nor the subjects to whom the data refer to want the cloud service provider (CSP) to read, use or sell their data.

In this thesis we discuss several procedures to store *and process* sensitive data in a privacy-preserving way in untrusted clouds, where processing consists of two basic operations: scalar products, which will be discussed in this chapter,

and matrix products (Chapter 4). These operations are useful in statistics and data analysis (to compute correlations between attributes, contingency tables, etc.), and also in engineering (image encryption, 3D graphics simulation, etc.); see Section 1.2 of [47] and references therein.

For scalar products, *the privacy-protected sensitive data stored in the clouds are not encrypted* and *fully preserve the utility (that is, the statistical features) of the original data*. In fact, the data are neither modified nor encrypted. This allows making the most of the outsourced data, while ensuring that no original records can be re-created from the outsourced records. The outsourced data can be used for purposes other than computing scalar products or matrix products. This is a relevant difference with respect to related work on algebraic computation outsourcing (see Section 3.2).

In our protocols, we use vertical splitting, so that each cloud stores a cleartext fragment on which any statistical analysis can be directly performed by the cloud. The goal is that the outsourced version achieves the same utility of the original data for exploratory analysis by any user with direct access to the cloud-stored data (who should, however, be unable to reconstruct the original data). Note that there are many organizations interested in releasing privacy-protected data for secondary analysis, including but not limited to official statistics [40].

Following the architecture defined in the "CLARUS" European H2020 project [17] (within which this work has been carried out), we will assume a proxy located in a domain trusted by the data controller (e.g., a server in her company's intranet or a plug-in in her device) that implements security and privacy-enabling features towards the cloud service providers. We will call this trusted proxy CLARUS, see Section 2.2 for more details.

Contributions

In this chapter, we explore new non-cryptographic protocols for the scalar product on split data and also consider cryptographic protocols to compute on split data. In the cryptographic protocols we use encryption only in the communication between clouds. However, the sensitive data stored in the clouds are protected by splitting, not by encryption. Regarding the sharing assumptions, we assume that the CSPs do not pool their fragments to reconstruct the original data set. We start from two existing protocols for this setting, one use cryptography and the other does not, and we develop two new non-cryptographic protocols and two variants of the cryptographic protocol.

The contributions in this chapter have been published in [21]. A preliminary version appeared in [12], where already existent non-cryptographic proposals for

secure scalar product and secure matrix product on vertically split data were evaluated. Moreover, we proposed two new non-cryptographic protocols and enhanced already existent cryptographic protocols adapted to the CLARUS scenario. The CSPs were assumed to be honest-but-curious and not to share information with each other.

This chapter is organized as follows: Section 3.2 depicts the related work on algebraic computation outsourcing and points out how our contributions differ from the previous proposals. Section 3.3 focuses on secure scalar products on vertically partitioned data when CSPs are honest-but-curious and do not share information: a non-cryptographic protocol and a cryptographic protocol are reviewed, and then two new non-cryptographic protocols and two variants of the cryptographic protocol are presented. After that, we show how secure scalar products between pairs of clouds can be combined with computations involving a single cloud to perform data analyses such as correlations and contingency tables. This section also contains mathematical technicalities related to the presented cryptographic protocols. In Section 3.4, the computation and communication costs of all protocols described in Sections 3.3 are assessed and compared against a benchmark protocol consisting of the CLARUS proxy downloading the entire data set and locally computing on the downloaded data set. Section 3.5 presents the experimental results obtained by implementing the proposed protocols in a multi-cloud scenario. Finally, Section 3.6 lists some conclusions.

3.2 Related work

Secure scalar products can be based on cryptography or not. Cryptographic approaches may use a variety of techniques. For instance, the protocol in [35] involves homomorphic encryption. Similarly, [49] presents a protocol in which all users encrypt their private vector using fully homomorphic encryption and upload it to a server. A user (initiator) sends a scalar product query to the server, which returns the final result after a series of computations and collaborative operations with the other user.

Non-cryptographic approaches are based on modifying the data before sharing them in such a way that the original data cannot be deduced from the shared data but the final results are preserved (e.g. [18], [27], [28], [45]).

The vast majority of protocols proposed for computing on vertically split data do without a trusted third party (with the exception of the commodity server solution of [27, 28]). While avoiding a trusted third party is technically

elegant, it normally takes more computation and communication. See [74] for a performance comparison of several protocols for the secure scalar product.

As explained in Section 2.2, our scenario involves a trusted party, the CLARUS proxy; in particular, the clouds holding the private vectors do not see the result of the scalar product (that is only seen by the CLARUS proxy). Hence, our setting is simpler than the one usually assumed in secure two-party computation, in which there is no third party and at least one of the parties learns the computation output.

3.3 Secure scalar product for data analysis on vertically partitioned data

Initially, the data are stored by the data controller vertically split across several clouds, either directly or via the CLARUS proxy. After that, users want to use the CLARUS proxy to compute scalar products between attributes stored in different clouds. Our goal is to minimize the computation at the CLARUS proxy, which is a resource in the controller's premises. In contrast, we assume that the CSPs have unlimited storage and computational power, and, therefore, we want to shift as much of the computational and storage load as possible to the CSPs' side. As to security, *no cloud should learn the data stored by the other clouds, but the CLARUS proxy is trusted and is entitled to know everything.*

In this context, we assume that the clouds are honest-but-curious, i.e. they do not deviate from the protocols. If properly performed, splitting does not allow linking confidential information to specific individuals. For example, if a fragment consists of the values of an attribute "Diagnosis", then just knowing a list of diagnoses is useless to an intruder, because he cannot associate them to the corresponding subjects (it is nearly as useless as seeing a list of diseases and their frequencies in a manual of medicine).

In Section 3.3.1, we start from the most efficient protocol for computing on vertically split data identified in [74] and adapted for use with the CLARUS proxy in [12]. We identify several shortcomings of this protocol in the CLARUS setting and we modify it by replacing or complementing random noise additions with permutations. In Section 3.3.2, we revise and adapt to the CLARUS scenario a cryptographic protocol for the secure distributed scalar product. Section 3.3.4 illustrates how secure scalar products can be used to compute correlations and contingency tables on vertically split data.

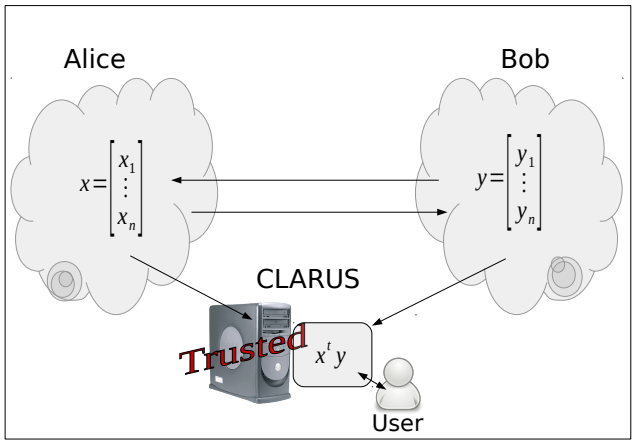

Figure 3.1: Honest-but-curious, non-sharing CSPs

3.3.1 Secure scalar product without cryptography

In this section, we work under the honest-but-curious and non-sharing model: the CSPs neither deviate from the protocols nor pool the data fragments they hold. Let $\mathbf{x}$ and $\mathbf{y}$ be two vectors with n components owned by Alice and Bob (who can be two CSPs), respectively. The goal is to securely compute the product $\mathbf{x^T y}$, see Figure 3.1. The privacy of the following protocols relies on the fact that the original vectors $\mathbf{x}$ and $\mathbf{y}$ are not shared at any time by the respective CSPs owning them; only linear transformations of them are shared, such that the number of unknowns (randomness) added by the tranformations is greater than or equal to the number of private unknowns. In the following protocols, CLARUS obtains the desired result; note than any disclosure by Alice or Bob to CLARUS does not entail any privacy leak, because (unlike Alice and Bob) CLARUS is trusted by the data controller/owner.

We take as starting point the protocol proposed in [27, 28], that is based on what they call a commodity server. This protocol is identified as the most efficient one in [74]. Let Alice and Bob be as previously defined and let a third, non-sharing cloud Charlie play the role of the commodity server. In [12], we suggested the following variant adapted for use with the CLARUS proxy:

Protocol 1.

1. *Charlie generates a random n-vector $\mathbf{r_x}$ and another n-vector $\mathbf{r_y}$ and computes $p = \mathbf{r_x^T r_y}$ (note that p is a number).*

2. *Charlie sends Alice the seed for a common random generator of $\mathbf{r_x}$, and sends Bob the seed for a common random generator of $\mathbf{r_y}$ (or equivalently Charlie sends $\mathbf{r_x}$ and $\mathbf{r_y}$ to Alice and Bob, respectively, if sending these vectors is faster than Alice and Bob generating them). Also, Charlie sends p to CLARUS.*

3. *Alice sends $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{r_x}$ to Bob.*

4. *Bob sends $t = \hat{\mathbf{x}}^T\mathbf{y}$ to CLARUS and sends $\hat{\mathbf{y}} = \mathbf{y} + \mathbf{r_y}$ to Alice.*

5. *Alice sends $s_x = \mathbf{r_x^T}\hat{\mathbf{y}}$ to CLARUS.*

6. *CLARUS computes $t - s_x + p$ $[= (\mathbf{x}+\mathbf{r_x})^\mathbf{T}\mathbf{y} - \mathbf{r_x^T}(\mathbf{y}+\mathbf{r_y}) + \mathbf{r_x^T r_y}] = \mathbf{x^T y}$.*

The authors of [27] show that their basic protocol allows neither Alice to learn $\mathbf{y}$ nor Bob to learn $\mathbf{x}$. In [12], is it is shown that the above variant still offers the same security regarding Alice and Bob as the basic protocol [27].

It is important to note that in Protocol 1 (as well as in the basic protocol [27]), the generated random vectors $\mathbf{r_x}$ and/or $\mathbf{r_y}$ should be reused in successive instances of the protocol with the same original data vectors $\mathbf{x}$ and/or $\mathbf{y}$, in order to avoid leaking new equations that would facilitate the reconstruction of a player's original data vector by the other player. It is easy for Alice to store $\mathbf{r_x}$ along with $\mathbf{x}$ for subsequent potential reuse, and the same holds for Bob with respect to $\mathbf{r_y}$ and $\mathbf{y}$. However, computing p requires knowledge of *both* $\mathbf{r_x}$ and $\mathbf{r_y}$, which neither Alice nor Bob have. On the other hand, Charlie knew both random vectors when he generated them, but he is unaware of any reuse unless told. So we propose to add the following two preliminary steps to Protocol 1:

pi. If Alice wants to reuse a previous private vector $\mathbf{x}$, she sends Charlie the corresponding seed of the random vector $\mathbf{r_x}$ (or equivalently sends the vector if doing so is faster than Charlie generating it).

pii. If Bob wants to reuse a previous private vector $\mathbf{y}$, he sends Charlie the seed of $\mathbf{r_y}$ (or equivalently sends the vector if doing so is faster than Charlie generating it).

and modify the first two steps of Protocol 1 as

1. If Charlie has not received $\mathbf{r_x}$, he generates it randomly; if Charlie has not received $\mathbf{r_y}$, he generates it randomly. Charlie computes $p = \mathbf{r_x^T r_y}$.

2. If Charlie generated $\mathbf{r_x}$, he sends the seed used to generate this vector to Alice; if Charlie generated $\mathbf{r_y}$, he sends the seed used to generate this vector to Bob (equivalently, instead of sending seeds, Charlie may send the actual vectors if doing so is faster). Also, Charlie sends p to CLARUS.

Beyond the need to manage reuse as specified above, using random vectors can result in the following potential weaknesses:

- Weak choices of $\mathbf{r_x}$ and $\mathbf{r_y}$ could also leak information, and should therefore be avoided: for example, an unsafe choice is when only one component of $\mathbf{r_x}$ (or $\mathbf{r_y}$) is different from zero.
- While the basic protocol in [27] and Protocol 1 do not leak the *exact* values of $\mathbf{y}$ to Alice or the exact values of $\mathbf{x}$ to Bob, it may be possible to infer the range of some elements in $\mathbf{x}$ and $\mathbf{y}$ from the range of the random vectors, $\mathbf{r_x}$ and $\mathbf{r_y}$. For example, imagine the elements of $\mathbf{x}$ are known to lie in the domain $[0, 100]$ (the domain of an attribute may sometimes be estimated from its semantics, e.g., the domain of *Age* can be estimated at $[0, 100]$). On the other hand, assume Charlie generates the elements of $\mathbf{r_x}$ by randomly sampling the $[0, 200]$ domain (the parameters of (pseudo)random number generation are normally public). Then, if an element of vector $\mathbf{x}+\mathbf{r_x}$ is 250, Bob learns that the corresponding original element in vector $\mathbf{x}$ is greater than 50.

A way to avoid the previous shortcomings of noise addition is to resort to the other main principle of non-cryptographic data protection, namely permutation [20]. Independent random permutation of the values of each attribute affords suitable protection if: a) the values taken by the attribute to be permuted are diverse enough; b) breaking the joint occurrence of attribute values in original records is deemed sufficient to protect the subjects' privacy, but the values of each attribute can be released as long as they cannot be linked to the corresponding subjects (in fact, using vertical data splitting for privacy is also predicated on this assumption). Specifically, we propose an alternative new permutation-based protocol, which is graphically depicted in Figure 3.2 and whose steps are as follows:

Protocol 1.1.

1. *Alice randomly permutes the values in her private vector to obtain* $\hat{\mathbf{x}} = \mathcal{P}_x(\mathbf{x})$.
2. *Alice sends* $\hat{\mathbf{x}}$ *to Bob and* $\mathbf{r_x} = \hat{\mathbf{x}} - \mathbf{x}$ *to Charlie.*

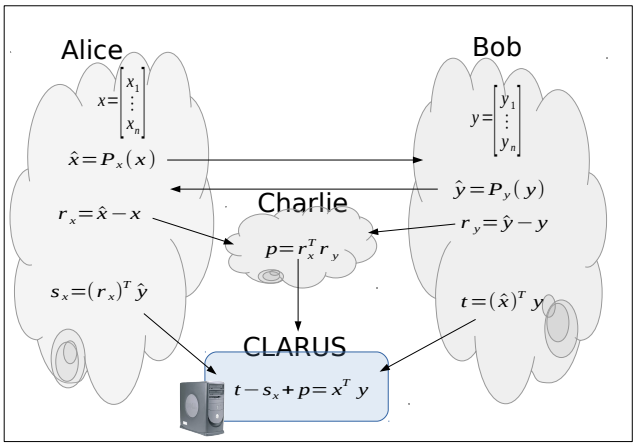

Figure 3.2: In Protocol 1.1, the CSPs permute their private vectors

3. *Bob randomly permutes the values in his private vector to obtain* $\hat{\mathbf{y}} = \mathcal{P}_y(\mathbf{y})$.
4. *Bob sends* $\hat{\mathbf{y}}$ *to Alice and* $\mathbf{r_y} = \hat{\mathbf{y}} - \mathbf{y}$ *to Charlie.*
5. *Charlie sends* $p = \mathbf{r_x^T r_y}$ *(note that* p *is a number) to CLARUS.*
6. *Bob sends* $t = \hat{\mathbf{x}}^T \mathbf{y}$ *to CLARUS.*
7. *Alice sends* $s_x = \mathbf{r_x^T} \hat{\mathbf{y}}$ *to CLARUS.*
8. *CLARUS computes* $t - s_x + p \ [= (\mathbf{x}+\mathbf{r_x})^\mathbf{T}\mathbf{y} - \mathbf{r_x^T}(\mathbf{y}+\mathbf{r_y}) + \mathbf{r_x^T r_y}] = \mathbf{x^T y}$.

Unlike in Protocol 1, randomness in Protocol 1.1 does not consist in adding random numbers and it is not generated by Charlie: Alice and Bob are the ones generating random permutations for their own vectors. Hence, the shortcomings identified above for Protocol 1 (need to handle random vector reuse, possible weak choices of random values, possible partial inferences) do not apply to Protocol 1.1. Regarding security, we have the following result.

Proposition 1 (Security)**.** *After participating in Protocol 1.1, Charlie does not learn* $\mathbf{x}$ *or* $\mathbf{y}$*; the probability of Bob's guessing* $\mathbf{x}$ *is at most*

$$\frac{n_1^x! n_2^x! \ldots n_{d_x}^x!}{n!},$$

where d_x is the number of different values among the n values of $\mathbf{x}$, and n_i^x is the number of repetitions of the i-different value; analogously, the probability of Alice's guessing $\mathbf{y}$ is at most

$$\frac{n_1^y! n_2^y! \dots n_{d_y}^y!}{n!}.$$

Further, the probability of Bob's guessing one particular component in $\mathbf{x}$ is at most $\max(n_1^x, \dots, n_{d_x}^x)/n$ and the probability of Alice's guessing one particular component in $\mathbf{y}$ is at most $\max(n_1^y, \dots, n_{d_y}^y)/n$.

Proof. Charlie receives $\mathbf{r_x}$ from Alice. Note that $\mathbf{r_x}$ can be obtained as the difference between $\hat{\mathbf{x}}+\mathbf{k}$ and $\mathbf{x}+\mathbf{k}$, where $\mathbf{k}$ is an n-vector with all its components set to k, where k is any real number. Hence, Charlie learns nothing about $\mathbf{x}$. A similar argument shows that Charlie learns nothing about $\mathbf{y}$.

On the other hand, Bob receives $\hat{\mathbf{x}}$ from Alice, which is a random permutation of $\mathbf{x}$. The number of different permutations of $\mathbf{x}$ is $\frac{n!}{n_1^x! n_2^x! \dots n_{d_x}^x!}$; hence, the probability of Bob's guessing the correct one is 1 divided by this number. The argument for the probability of Alice's guessing $\mathbf{y}$ is analogous.

Finally, if Bob wants to guess a particular component of $\mathbf{x}$ given $\hat{\mathbf{x}}$, his best guess is the most frequent value in $\hat{\mathbf{x}}$, which is also the most frequent value in $\mathbf{x}$. The probability of the target component coinciding with the most frequent value is the relative frequency of the most frequent value. The argument when Alice wants to guess a component of $\mathbf{y}$ is analogous. □

If the probabilities given by Proposition 1 are not considered low enough (this may be the case if data are not diverse enough), then Protocol 1.1 should not be used. In this case, another option can be to combine permutation and noise addition into a hybrid of Protocol 1 and Protocol 1.1, as follows:

Protocol 1.2.

1. *Charlie sends Alice the seed for a common random generator of a random n-vector $\mathbf{r_x}$, and sends Bob the seed for a common random generator of a random n-vector $\mathbf{r_y}$ (or equivalently generates and sends the vectors if doing so is faster than Alice and Bob generating them).*

2. *Alice computes $\hat{\mathbf{x}} = \mathbf{x} + \mathbf{r_x}$ and randomly permutes the values in $\hat{\mathbf{x}}$ to obtain $\hat{\mathbf{x}}' = \mathcal{P}_x(\hat{\mathbf{x}})$.*

3. *Alice sends $\hat{\mathbf{x}}'$ to Bob and $\mathbf{r}'_\mathbf{x} = \hat{\mathbf{x}}' - \mathbf{x}$ to Charlie.*

4. *Bob computes* $\hat{\mathbf{y}} = \mathbf{y} + \mathbf{r_y}$ *and randomly permutes the values in* $\hat{\mathbf{y}}$ *to obtain* $\hat{\mathbf{y}}' = \mathcal{P}_y(\hat{\mathbf{y}})$.

5. *Bob sends* $\hat{\mathbf{y}}'$ *to Alice and* $\mathbf{r}'_\mathbf{y} = \hat{\mathbf{y}}' - \mathbf{y}$ *to Charlie.*

6. *Charlie sends* $p = (\mathbf{r}'_\mathbf{x})^\mathbf{T}\mathbf{r}'_\mathbf{y}$ *(note that* p *is a number) to CLARUS.*

7. *Bob sends* $t = (\hat{\mathbf{x}}')^T\mathbf{y}$ *to CLARUS.*

8. *Alice sends* $s_x = (\mathbf{r}'_\mathbf{x})^\mathbf{T}\hat{\mathbf{y}}'$ *to CLARUS.*

9. *CLARUS computes*

$$t - s_x + p \ [= (\mathbf{x} + \mathbf{r}'_\mathbf{x})^\mathbf{T}\mathbf{y} - (\mathbf{r}'_\mathbf{x})^\mathbf{T}(\mathbf{y} + \mathbf{r}'_\mathbf{y}) + (\mathbf{r}'_\mathbf{x})^\mathbf{T}(\mathbf{r}'_\mathbf{y})] = \mathbf{x}^\mathbf{T}\mathbf{y}.$$

In the above protocol, reusing the random vectors in successive instances is not needed, because the components are randomly permuted each time, so that an attacker cannot link the successive noise-added versions of the same original component of $\mathbf{x}$ (resp. $\mathbf{y}$). Regarding security, Protocol 1.2 is at least as secure as Protocol 1. Specifically:

Proposition 2 (Security)**.** *Protocol 1.2 does not allow Charlie to learn* $\mathbf{x}$ *or* $\mathbf{y}$*, it does not allow Alice to learn* $\mathbf{y}$*, and it does not allow Bob to learn* $\mathbf{x}$*.*

Proof. Charlie receives $\mathbf{r}'_\mathbf{x}$ from Alice. Note that $\mathbf{r}'_\mathbf{x}$ can be obtained as the difference between $\hat{\mathbf{x}}'+\mathbf{k}$ and $\mathbf{x}+\mathbf{k}$, where $\mathbf{k}$ is an n-vector with all its components set to k, where k is any real number. Hence, Charlie learns nothing about $\mathbf{x}$. A similar argument shows that Charlie learns nothing about $\mathbf{y}$.

Regarding Alice and Bob, Protocol 1.2 clearly offers at least the same security as Protocol 1. In fact, due to the additional random permutation step and provided that there is some diversity in the original attribute values, it offers better security: it is free from the shortcomings of Protocol 1 related to poor choice of random vectors and to range inference. At best, the attacker can make inferences on permuted values. □

3.3.2 Secure scalar product with cryptography

Using cryptography to compute the scalar product of two vectors $\mathbf{x}$ and $\mathbf{y}$ privately owned by Alice and Bob, respectively, can be expected to increase the computational complexity with respect to non-cryptographic protocols. However, it is attractive in terms of security if it can be shown that for Alice to

learn $\mathbf{y}$ or Bob to learn $\mathbf{x}$ they should break a cryptosystem that is known to be secure.

In [35], the authors proposed a cryptographic protocol based on the Paillier homomorphic cryptosystem [51]. Let $\mathbf{x} = (\mathbf{x_1}, \ldots, \mathbf{x_n})^\mathbf{T}$ and $\mathbf{y} = (\mathbf{y_1}, \ldots, \mathbf{y_n})^\mathbf{T}$. We remain under the honest-but-curious, non-sharing model: the CSPs neither deviate from the protocols nor pool the data fragments they hold. The protocol is depicted in Figure 3.1 and consists of the following steps:

Protocol 2.

Set-up phase:

1. *Alice generates a private and public key pair* (s_k, p_k) *and sends* p_k *to Bob.*

Scalar product:

2. *Alice generates the ciphertexts* $c_i = Enc_{p_k}(x_i; r_i)$*, where* r_i *is a random number in* $\mathbb{F}_N$*, for every* $i = 1, \ldots, n$*, and sends them to Bob.*
3. *Bob computes* $\omega = \prod_{i=1}^{n} c_i^{y_i}$.
4. *Bob generates a random plaintext* s_B*, a random number* r' *and sends* $\omega' = \omega Enc_{p_k}(-s_B; r')$ *to Alice.*
5. *Alice computes* $s_A = Dec_{s_k}(\omega') = \mathbf{x^T y} - \mathbf{s_B}$.
6. *Alice and Bob simultaneously exchange the values* s_A *and* s_B*, respectively, so that both can compute* $s_A + s_B = \mathbf{x^T y}$.

Protocol 2 works in a finite field $\mathbb{F}_N$, where the order N must be large enough (as explained in 3.3.3) and it is the product of two primes p and q of the same length and such that $\gcd(pq, (p-1)(q-1)) = 1$. In case Alice and Bob need to execute this protocol several times, they can reuse public and private keys and thus the set-up step (first step) needs to be executed only once. The number of computations required is: Bob must perform n exponentiations and one encryption, and Alice has to perform n encryptions and one decryption. Encryption involves computing two exponentiations and multiplying them, but one of the exponentiations can be precomputed. Decryption needs one exponentiation as its most expensive operation. The complexity of all these operations depends on N: the larger N, the more computationally demanding they are.

In Protocol 2, both Alice and Bob obtain the result. If we want only the proxy to learn it, we propose the following variation of the last steps:

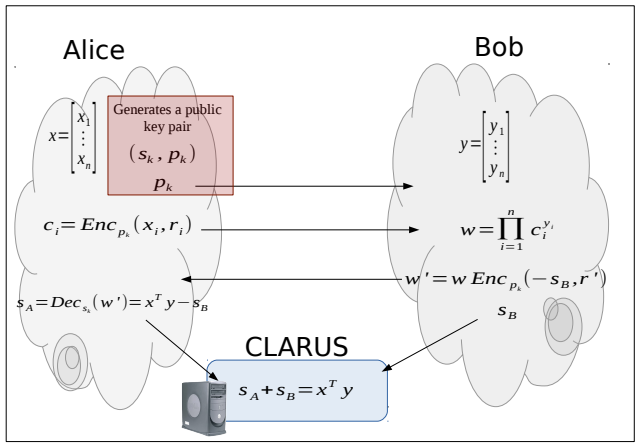

Figure 3.3: Protocol 2.1 is based on the Paillier homomorphic cryptosystem. In this case, the generation of the public key pair is left to Alice.

Protocol 2.1.

4. *Bob generates a random plaintext s_B, a random number r', sends $\omega' = \omega Enc_{p_k}(-s_B; r')$ to Alice and sends s_B to CLARUS.*

5. *Alice sends $s_A = Dec_{s_k}(\omega') = \mathbf{x^T y} - \mathbf{s_B}$ to CLARUS.*

6. *CLARUS computes $s_A + s_B = \mathbf{x^T y}$.*

A sketch of Protocol 2.1 is given in Figure 3.3.

If it is possible for CLARUS to use an auxiliary cryptographic module, the number of computations can be reduced, and the protocol can be simplified as follows:

Protocol 2.2.

Set-up phase:

1. *CLARUS generates a private and public key pair (s_k, p_k) and sends p_k to Alice and Bob.*

Scalar product:

2. *Alice generates the ciphertexts* $c_i = Enc_{p_k}(x_i; r_i)$*, where* r_i *is a random number belonging to* $\mathbb{F}_N$*, for every* $i = 1, \ldots, n$ *and sends them to Bob.*

3. *Bob computes* $\omega = \prod_{i=1}^{n} c_i^{y_i}$ *and sends it to CLARUS.*

4. *CLARUS computes* $Dec_{s_k}(\omega) = \mathbf{x^T y}$.

We have the following security result regarding the previous protocols.

Proposition 3 (Security)**.** *If Paillier's cryptosystem is secure, then Protocols 2.1 and 2.2 are secure, in the sense that Alice cannot learn* $\mathbf{y}$ *and Bob cannot learn* $\mathbf{x}$.

Proof. It is proven in [35] that Protocol 2 is secure in the above sense if the Paillier cryptosystem is secure (see [51] about the security of this cryptosystem).

Regarding Protocol 2.1, the only modification with respect to Protocol 2 is that Alice and Bob do not share their results s_A and s_B, but they send these values to CLARUS. Since neither Alice nor Bob have more information than in Protocol 2, the security of Protocol 2 is preserved by Protocol 2.1.

Regarding Protocol 2.2, the only differences with Protocol 2 are that: Alice neither generates the key pair in the set-up phase nor decrypts ω' later; Bob does not generate nor encrypt s_B; Alice and Bob do not share their results s_A and s_B, but they send these values to CLARUS. Neither Alice nor Bob have more information than in Protocol 2. Therefore, the security of Protocol 2 is preserved by Protocol 2.2. □

3.3.3 Finite field for the computations

Protocol 2 and its variants work in a finite field $\mathbb{F}_N$, i.e. given $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{y} = (y_1, \ldots, y_n)$ two private n-vectors, we are computing $\boldsymbol{x}^T\boldsymbol{y} \mod (N)$. If we do not want the result to be modified by the modulus, it must hold that $N > \boldsymbol{x}^T\boldsymbol{y}$. Let $M_x = \max_{x_i \in x} x_i$ be the maximum value belonging to $\boldsymbol{x}$, $M_y = \max_{y_i \in y} y_i$ the maximum value belonging to $\boldsymbol{y}$ and $M = \max\{M_x, M_y\}$. It is sufficient to choose $N > nM_xM_y$. To chose such an N, we suggest that, before the protocol, Bob send $M'_y > M_y$ to Alice who chooses $M'_x > M_x$, and then picks $N > nM'_xM'_y$.

Alternatively, Alice can one-sidedly choose a very large N without Bob's input (a 1024-bit N is a common choice with Paillier's cryptosystem). In Protocol 2.2 the public key generation and hence this one-sided choice would be done by CLARUS.

The choice of a very large N allows decreasing the computational cost of the set-up phase of Protocols 2, 2.1 and 2.2 (reading the vectors is not necessary anymore and so the set-up phase has $O(1)$ cost). On the other hand, the computational cost of the scalar product in all three protocols is considerably increased, because a larger N means larger keys and ciphertexts, which make cryptographic operations slower.

3.3.4 Example data analyses based on scalar products

In vertical splitting, analyses that involve only attributes in a single fragment are really fast and easy to compute: the cloud storing the fragment can compute and send the outputs of the analyses to the CLARUS proxy. Unfortunately, many statistical analyses, such as regression, classification, principal component analysis, etc., are likely to involve attributes stored in different fragments, and thus communication between clouds.

The sample correlation matrix $\hat{\zeta}$ is fundamental for many statistical analyses. Let $\mathbf{X}$ be the original data set with n rows (records) and m columns (attributes $\mathbf{X_1}, \cdots, \mathbf{X_m}$). The sample correlation matrix of $\mathbf{X}$ can be computed as $\hat{\zeta} = (\hat{\rho}_{ij})$ for $1 \leq i, j \leq m$, where

$$\hat{\rho}_{ij} = \frac{1}{n} \frac{\mathbf{X_i^T X_j} - \mathbf{n}\hat{\mu}_\mathbf{i}\hat{\mu}_\mathbf{j}}{\hat{\sigma}_i \hat{\sigma}_j} \tag{3.1}$$

with $\hat{\mu}^T = (\hat{\mu}_1, \ldots, \hat{\mu}_m)$ being the vector of sample means and $\hat{\sigma}^T = (\hat{\sigma}_1, \ldots, \hat{\sigma}_m)$ the vector of sample standard deviations of the attributes of $\mathbf{X}$. Regarding the scalar product $\mathbf{X_i^T X_j}$ in the numerator of Equation (3.1), it can be viewed as a component of the following matrix

$$\begin{aligned} \mathbf{X^T X} &= (\mathbf{X_1}|\cdots|\mathbf{X_m})^T (\mathbf{X_1}|\cdots|\mathbf{X_m}) \\ &= \left(\begin{array}{c|c|c|c} \mathbf{X_1^T X_1} & \mathbf{X_1^T X_2} & \cdots & \mathbf{X_1^T X_m} \\ \hline \mathbf{X_2^T X_1} & \mathbf{X_2^T X_2} & \cdots & \mathbf{X_2^T X_m} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \mathbf{X_m^T X_1} & \mathbf{X_m^T X_2} & \cdots & \mathbf{X_m^T X_m} \end{array}\right). \end{aligned}$$

Each element of $\hat{\mu}$, $\hat{\sigma}$ and each diagonal element $\mathbf{X_i^T X_i}$ can be separately computed by the respective cloud and sent to the CLARUS proxy, who calculates $\hat{\zeta}$. Off-diagonal elements are also easy to compute if the involved attributes are stored in the same cloud. The most challenging task is therefore calculating the off-diagonal elements of $\mathbf{X^T X}$ when the involved attributes are *not* in the same cloud. Note that it is not expensive for CLARUS to subsequently reassemble

and store $\mathbf{X^T X}$ and to derive $\hat{\zeta}$ according to Equation (3.1): on the one hand, if $\mathbf{X}$ has n records and m attributes with $m \ll n$, $\mathbf{X^T X}$ is an $m \times m$ matrix, and thus it is very small compared to the size of $\mathbf{X}$; on the other hand, the required computations to obtain $\hat{\zeta}$ are simple operations between short vectors (of m components) or numbers.

Computing $\mathbf{X_i^T X_j}$, for any $\mathbf{X_i}$ and $\mathbf{X_j}$ stored in different clouds, amounts to performing a secure scalar product of two vectors each held by a different party (where "secure" means without any party disclosing her vector to the other party). Therefore, obtaining the sample correlation matrix in vertical splitting among several clouds can be decomposed into several secure scalar products to be conducted between pairs of clouds.

Furthermore, being able to securely compute scalar products permits obtaining, in addition to sample correlation matrices, contingency tables (that is, cross-tabulations) in a very simple way. To help computing a cross-tabulation cell between value x of attribute $\mathbf{X_i}$ and value x' of attribute $\mathbf{X_j}$, the cloud holding $\mathbf{X_i}$ computes an auxiliary binary attribute as follows:

$$\mathbf{aux_{ix}} = \begin{cases} 1 \text{ for records with } \mathbf{X_i} = \mathbf{x}; \\ 0 \text{ for records with } \mathbf{X_i} \neq \mathbf{x}; \end{cases}$$

similarly, the cloud holding $\mathbf{X_j}$ computes another auxiliary attribute:

$$\mathbf{aux_{jx'}} = \begin{cases} 1 \text{ for records with } \mathbf{X_j} = \mathbf{x'}; \\ 0 \text{ for records with } \mathbf{X_j} \neq \mathbf{x'}. \end{cases}$$

Finally, to count the joint occurrences of $\mathbf{X_i} = \mathbf{x}$ and $\mathbf{X_j} = \mathbf{x'}$, the clouds holding $\mathbf{X_i}$ and $\mathbf{X_j}$, respectively, engage in a secure scalar product protocol of their attributes $\mathbf{aux_{ix}}$ and $\mathbf{aux_{jx'}}$. Note that this procedure is directly applicable to discrete numerical and categorical attributes, and can also be applied to continuous numerical attributes if discretized as intervals.

3.4 Comparison among methods

We compare here the protocols described in the previous sections against benchmark solutions that consist of CLARUS downloading the data from the cloud or clouds and computing locally.

Specifically, the protocols to compute scalar products over vertically split data described in Section 3.3 are compared to the following benchmark protocol:

Protocol 3.

1. *Alice and Bob send* $\mathbf{x}$ *and* $\mathbf{y}$ *to CLARUS, respectively.*
2. *CLARUS locally computes* $\mathbf{x^T y}$.

The case of CLARUS outsourcing encrypted data to a single CSP can also be used as benchmark. The whole data set $\mathbf{X}$ (containing in particular vectors $\mathbf{x}$ and $\mathbf{y}$) is stored in a single cloud in an encrypted form. In this setting, CLARUS first encrypts the data set, stores it in the cloud and, when it needs to compute a scalar product, it downloads and decrypts the data set, as detailed in the following protocol:

Protocol 4.

Set-up phase:

1. *CLARUS encrypts the original data set* $\mathbf{E} = \mathbf{Enc}(\mathbf{X})$
2. *CLARUS sends* $\mathbf{E}$ *to a cloud Alice for remote storage, and deletes* $\mathbf{X}$ *from local storage.*

Matrix product computation:

3. *CLARUS requests* $\mathbf{E}$ *from Alice, decrypts* $\mathbf{X} = \mathbf{Dec}(\mathbf{E})$ *and performs the computation* $(\mathbf{X})^\mathbf{T}\mathbf{X}$.

Encryption and decryption can be performed using a fast symmetric cryptosystem, such as the Advanced Encryption Standard (AES), which takes time linear in the number of records/vector components n, as well as ciphertexts similar in size to the corresponding plaintexts.

We now evaluate the computational cost for Alice, Bob, CLARUS and the total computation under each protocol. Moreover, operations that do not need to be repeated each time the protocol is executed, specifically the generation of cryptographic keys in Protocols 2.1 and 2.2, are separately counted as set-up costs.

Considering that the clouds have unlimited storage, it is reasonable to assume that they can store any random matrices or vectors that may need to be reused. In contrast, we do not assume unlimited storage at CLARUS; therefore, we assume the proxy just stores the random seeds and (re)generates random matrices or vectors when needed. Also, the cost of a communication is associated both to the sender (who needs to send the data) and to the receiver (who needs to read the data). We use a parameter γ to represent the maximum length of the numbers in the vectors and matrices used in the protocols. For the case

Table 3.1: **(Set-up computation and communication costs of Protocols 1, 1.1, 1.2, 2.1, 2.2, 3 and 4)**: n is the length of the private vectors $\mathbf{x}$ and $\mathbf{y}$. γ represents the maximum length of the numbers in the vectors and matrices used in the protocols. N is the size of the plaintext field used by Paillier (see 3.3.3). The computation cost is presented in terms of the costliest operations performed in each case. Protocols 2.1 and 2.2 have two different computation costs depending on the choice of N (they are separated by "|" in the table): if the smallest possible N is taken, then the private vectors need to be read (see 3.3.3 for details). The communication cost is the exact amount of data transmitted. The Paillier key generation cost is indicated with "PKgen", the AES keys generation with "RNDgen" and the AES encryption with "AESencr". Note: in protocols not requiring the presence of Bob or Charlie, their costs are indicated with "–".

	Set-up							
	Computation				Communication			
	Alice	Bob	Charlie	CLARUS	Alice	Bob	Charlie	CLARUS
Prot. 1	0	0	0	0	0	0	0	0
Prot. 1.1	0	0	0	0	0	0	0	0
Prot. 1.2	0	0	0	0	0	0	0	0
Prot. 2.1	n read + PKgen | PKgen	n read|0	–	0	$3\log_2 N$	$3\log_2 N$	–	0
Prot. 2.2	n read|0	n read|0	–	PKgen	$3\log_2 N$	$3\log_2 N$	–	$6\log_2 N$
Prot. 3	0	0	–	0	0	0	–	0
Prot. 4	0	–	–	n AESencr +1 RNDgen	$2n\gamma$	–	–	$2n\gamma$

of Protocols 2.1 and 2.2, lengths are a function of the size N of the field used by the Paillier cryptosystem (see Section 3.3.3): the public key is $3\log_2 N$ bits long, the secret key is $\log_2 N$ bits long, ciphertexts are $2\log_2 N$ bits long and plaintexts are $\log_2 N$ bits long. We consider that, whenever possible, the participants send the seeds of random vectors and matrices, rather than the vectors and matrices themselves. If communications are very fast and/or vectors are very short, sending the vectors rather than the seeds might be preferable (see related experimental results in Section 3.5).

3.4.1 Comparison for scalar products over split data

In this section, we compare the protocols based on data splitting (Protocols 1, 1.1, 1.2, 2.1 and 2.2) with the benchmark Protocols 3 and 4. We do not detail the costs of Protocol 2, because they are mostly equivalent to those of the two variants, Protocols 2.1 and 2.2.

Table 3.1 shows the set-up computation costs and set-up communication

costs incurred by all protocols. For the computation cost, just giving the order of magnitude of the complexity is not accurate enough (e.g. n additions are faster than n multiplications, even if we have $O(n)$ computation in both cases); therefore, we give the complexity in terms of the costliest operation performed in each case. For instance, "read" means reading the vector, "AESencr" means AES encryption of the vectors and "RNDgen" is one random number generation. When the stored data are updated (that is, records are added, changed or removed), the set-up phase (key generation) of Protocols 2.1 and 2.2 needs to be repeated only if some of the new values are greater than the order N of the finite field in use; otherwise, it is possible to reuse the same keys without losing security. Protocol 4 requires downloading, decrypting, updating and re-encrypting all the records.

If in Protocols 2.1 and 2.2 one wants to save storage by using the smallest possible N that does not result in overflow, it takes $2n$ "read" effort (to check all vector elements). If one just takes a large N, say a 1024-bit N, then the key generation cost is constant. This is why the n reads have been marked as optional in Table 3.1 (See 3.3.3 for details). At most, the key generation for both Protocol 2.1 and Protocol 2.2 requires 3 random generations, 1 modular multiplicative inversion and 1 least common multiple computation, which is represented with "PKgen" in the table. It is possible to use the same keys also for different data sets belonging to the same field $\mathbb{F}_N$; therefore, the update of elements in the private vectors does not require generating new keys. Protocol 4 also requires a set-up phase, that is, the key generation for the AES cryptosystem and the encryption of the private vectors. Compared to Protocols 2.1 and 2.2, the set-up phase of Protocol 4 needs to be repeated every time that the private vectors are changed.

Table 3.1 also shows the communication cost of the set-up phase. As said above, the cost of communicating a number of bits is incurred by *both* the sender (who must write them to the channel) and the receiver (who must read them from the channel). Only Protocol 2.1, Protocol 2.2 and Protocol 4 involve a set-up communication cost, due to the exchange of the public key for the two former protocols and the transmittal of the encrypted private vectors for the latter one.

Table 3.2 shows the long-term and temporary data storage costs (temporary storage is the one needed only to conduct a certain calculation at some point). In Protocol 1, Alice needs long-term storage for her data vector $\mathbf{x}$ and also for the random vector $\mathbf{r_x}$, which is needed for potential reuse; similarly for Bob regarding $\mathbf{y}$ and $\mathbf{r_y}$. In Protocols 1.1 and 1.2, the random vectors do not need to be reused, so less long-term storage is needed by Alice and Bob; on the

Table 3.2: **(Long-term and temporary storage costs of Protocols 1, 1.1, 1.2, 2.1, 2.2, 3 and 4)**: n is the length of the private vectors $\mathbf{x}$ and $\mathbf{y}$; γ represents the maximum length of the numbers in the vectors and matrices used in the protocols. N is the size of the plaintext field used by Paillier (see 3.3.3). Note: in protocols not requiring the presence of Bob or Charlie, their costs are indicated with "–".

	Storage							
	Long-term				Temporary			
	Alice	Bob	Charlie	CLARUS	Alice	Bob	Charlie	CLARUS
Prot. 1	$(2n+1)\gamma$	$(2n+1)\gamma$	0	0	$(2n+1)\gamma$	$(2n+1)\gamma$	$(2n+3)\gamma$	4γ
Prot. 1.1	$n\gamma$	$n\gamma$	0	0	$(2n+1)\gamma$	$(2n+1)\gamma$	$(2n+1)\gamma$	4γ
Prot. 1.2	$(2n+1)\gamma$	$(2n+1)\gamma$	0	0	$(4n+1)\gamma$	$(4n+1)\gamma$	$(2n+3)\gamma$	4γ
Prot. 2.1	$n\gamma+$ $4\log_2 N$	$n\gamma$	–	0	$2\gamma+$ $2(n+1)\log_2 N$	$3\gamma+$ $2(n+2)\log_2 N$	–	3γ
Prot. 2.2	$n\gamma$	$n\gamma$	–	$4\log_2 N$	$\gamma+$ $2(n+1)\log_2 N$	$2(n+1)\log_2 N$	–	$\gamma+2\log_2 N$
Prot. 3	$n\gamma$	$n\gamma$	–	0	0	0	–	$(2n+1)\gamma$
Prot. 4	$2n\gamma$	–	–	0	0	–	–	$(4n+1)\gamma$

other hand, computing the random permutations in those protocols takes just n random number generations and no auxiliary storage, by using Durstenfeld's algorithm [30]. Only the benchmark Protocols 3 and 4 require CLARUS to (temporarily) store a large amount of data, namely the downloaded data, plus the decrypted data if the downloaded data are encrypted.

Table 3.3 and Table 3.4 show the computational and communication costs, respectively, incurred by the execution of the above mentioned protocols (after set-up). Protocols 1, 1.1, 1.2 and 2.1 have all similar costs for CLARUS. Protocol 2.2 requires more computation and communication from CLARUS, but these additional costs can be reduced if CLARUS has a cryptographic module. It is worth noting that *all protocols have constant computation, storage and communication costs for CLARUS; hence, they clearly outperform the benchmark Protocols 3 and 4, which require CLARUS to carry out computations, storage and communications whose complexity increase linearly with the data set size n.*

If we add the costs associated to all the involved parties, Protocol 1 is the most efficient one, closely followed by Protocol 1.1 and Protocol 1.2 (note that these three protocols do not require any set-up). If we focus on the cost/security trade-off, Protocol 1.2 is probably the best choice.

Table 3.3: **(Computation costs for Protocols 1, 1.1, 1.2, 2.1, 2.2, 3 and 4)**: n is the length of the private vectors $\mathbf{x}$ and $\mathbf{y}$; Finally, Charlie only appears in Protocols 1, 1.1 and 1.2, and Bob does not appear in Protocol 4; we indicate the absence of a cloud with "−". The computation cost is presented in terms of the costliest operations performed in each case.

	Computational cost			
	Alice	Bob	CLARUS	Charlie
Prot. 1	n prod. + n RNDgen.	n prod. + n RNDgen.	2 sum.	n prod. + $2n$ RNDgen.
Prot. 1.1	n prod. + n RNDgen.	n prod. + n RNDgen.	2 sum.	n prod.
Prot. 1.2	n prod. + n RNDgen.	n prod. + n RNDgen.	2 sum.	n prod.
Prot. 2.1	n RNDgen. + n encr. +1 decr.	n prod. + 2 RNDgen +1 encr.	1 sum.	−
Prot. 2.2	n RNDgen. + n encr.	n prod.	1 decr.	−
Prot. 3	0	0	n prod.	−
Prot. 4	0	−	n prod. + n AESdecr.	−

3.5 Experimental results

This section details the experimental results obtained by implementing the proposed protocols in Java in a multi-cloud scenario. Since Protocols 1 and 2 had security and functionality issues that motivated Protocols 1.1, 1.2, 2.1, and 2.2, we focused on implementing the latter protocols.

The reported experiments were conducted using the first two attributes of the California housing data set (CADATA, [10]), a usual test data set in the statistical disclosure control literature that contains 9 numerical attributes and $20,640$ records. Let $\mathbf{x}$ and $\mathbf{y}$ represent the two selected attributes of CADATA, and $\mathbf{X}$ be the matrix containing $\mathbf{x}$ and $\mathbf{y}$.

First, we ran the tests on Amazon Web Services (AWS), a public CSP that offers 12 months free tier. The tests were performed on a t2.micro Amazon EC2 instance for each cloud. Since the computing power and main storage were substantially capped in this free-of-charge service, and communication with it was slow, we took it as representing the low-end scenario a user can expect from a CSP. Second, we also used a local server (CERSEI) offering more computational power and main storage, as well as faster communication, in order to mimic the service that can be expected from a for-payment CSP. In both cases, we used another local computer to run CLARUS on it, that worked as the proxy located in a trusted domain. Table 3.5 summarizes the specifications of CLARUS,

Table 3.4: **(Communication costs for Protocols 1, 1.1, 1.2, 2.1, 2.2, 3 and 4)**: n is the length of the private vectors $\mathbf{x}$ and $\mathbf{y}$; γ represents the maximum length of the numbers in the vectors and matrices used in the protocols. Finally, Charlie only appears in Protocols 1, 1.1 and 1.2, and Bob does not appear in Protocol 4; we indicate the absence of a cloud with "−". The communication cost is the exact amount of data transmitted. In Protocol 1 we have considered the most usual case in which there is no reuse; for each reused private vector, $n\gamma$ communication cost is shifted from Charlie to the reusing cloud, and the computational cost for Charlie decreases by n RNDgen.

	Communication cost				Who needs a
	Alice	Bob	CLARUS	Charlie	crypt. module
Prot. 1	$(2n+2)\gamma$	$(2n+2)\gamma$	3γ	3γ	none
Prot. 1.1	$(3n+1)\gamma$	$(3n+1)\gamma$	3γ	$(2n+1)\gamma$	none
Prot. 1.2	$(3n+2)\gamma$	$(3n+2)\gamma$	3γ	$(2n+3)\gamma$	none
Prot. 2.1	$2(n+1)\log_2 N+\gamma$	$2(n+1)\log_2 N+\gamma$	2γ	−	Alice
Prot. 2.2	$2n\log_2 N$	$(2(n+1)\log_2 N$	$2\log_2 N$	−	CLARUS
Prot. 3	$n\gamma$	$n\gamma$	$2n\gamma$	−	none
Prot. 4	$2n\gamma$	−	$2n\gamma$	−	CLARUS

Table 3.5: **(Computational specifications of CLARUS and CSPs)** CLARUS, the trusted proxy, was run on a local computer. CERSEI is a local server mimicking a for-payment CSP. AWS represents a free-of-charge t2.micro Amazon EC2 instance.

Machine	Operating System	Width(bits)	CPU(GHz)	RAM(GB)	HDD(GB)
CLARUS	Windows 7	64	2.5	8	500
CERSEI	Ubuntu 14.4 LTS	64	3.4	16	500
AWS	Ubuntu Server 16.04 LTS	64	2.4	1	30

CERSEI and the AWS instance.

The computational cost was measured as the time in seconds that each machine spent to perform the computations specified by the protocols. The communication cost was measured as an approximation of the time each cloud spent at sending data (writing to the channel) or receiving data (reading from the channel). As explained in Section 3.4, we consider in general that the

participants send the seeds of random vectors and matrices, rather than the vectors and matrices themselves. The reason is that generating a random vector takes normally less time than sending it. However, the communication time depends on many factors (user internet connection, distance between users, network load, etc.). In our experimental setting, we compared the times for generating and sending vectors of several sizes n; the results are reported in Table 3.6, where it can be seen that sending a random vector takes longer than generating it at the recipient for sizes $n > 10^4$.

3.5.1 Experimental results for scalar products over split data

In this section, we detail storage, computation and communication costs of Protocols 1.1, 1.2 and 2.1 (based on data splitting) and the benchmark Protocols 3 and 4. We do not detail the costs of Protocol 2.2, because they are basically equivalent to those of Protocol 2.1.

Table 3.7 shows the comparison of long-term and temporary storage measured in bytes. The storage does not depend on the particular CSP used. Long-term storage turns out to be similar in all protocols for all players involved; the only remarkable difference occurs in Protocol 4, where CLARUS must keep the AES keys (whereas it keeps nothing in the other protocols). Temporary storage is of the same order of magnitude for Protocols 1.1, 1.2 and 2.1 and really small for CLARUS; instead, the benchmark Protocols 3 and 4 require large temporary storage on the CLARUS side.

Table 3.8 shows the computation and communication costs incurred by the execution of the above mentioned protocols. Communication takes substantial time both at the sender and at the receiver: obviously, the receiver cannot process the data until he receives them, and the sender needs to wait for the

Table 3.6: **(Time to send a random vector vs time to generate it at the recipient)**: n is the number of elements of a random vector $\mathbf{x}$ and the times are given in seconds. Sending is faster only for short vectors.

Time (s) \ n	10^3	10^4	10^5	10^6
to generate $\mathbf{x}$	2.08×10^{-3}	4.6×10^{-3}	8.4×10^{-3}	2.07×10^{-2}
to send $\mathbf{x}$	2.8×10^{-4}	3.5×10^{-3}	1.9×10^{-2}	1.19×10^{-1}

Table 3.7: **(Long-term and temporary storage costs of Protocols 1.1, 1.2, 2.1, 3 and 4)**: Private vectors $\mathbf{x}$ and $\mathbf{y}$ have length $n = 20,640$. The representation of the numbers takes $\gamma = 8$ bytes. Elements in the field $\mathbb{F}_N$ used by the Paillier cryptosystem have 65 bytes. We indicate the absence of a cloud with "–". The values are given in bytes.

	Storage (B)							
	Long-term				Temporary			
	Alice	Bob	Charlie	CLARUS	Alice	Bob	Charlie	CLARUS
Prot. 1.1	1.7×10^5	1.7×10^5	0	0	3.3×10^5	3.3×10^5	3.3×10^5	32
Prot. 1.2	3.3×10^5	3.3×10^5	0	0	6.6×10^5	6.6×10^5	3.3×10^5	32
Prot. 2.1	1.7×10^5	1.7×10^5	–	0	2.5×10^5	2.5×10^5	–	24
Prot. 3	1.7×10^5	1.7×10^5	–	0	0	0	–	3.3×10^5
Prot. 4	3.3×10^5	–	–	0	0	–	–	6.6×10^5

receiver's acknowledgment of receipt before carrying on. The communication between AWS and CLARUS is slow, in part because the Amazon services are geographically distant and in part because we used a free-of-charge instance. Note that, although CLARUS receives only scalars in Protocols 1.1, 1.2 and 2.1, its communication cost is greater than for Alice, who sends and receives vectors. The explanation is that: i) reading/sending between the CSPs (Alice and Bob) is faster than between the CSPs and CLARUS (because Alice and Bob are located in the same cloud system, AWS in one case or CERSEI in the other case); ii) in the CLARUS-CSP communication the time to send a scalar is dominated by the time to establish the channel. Protocols 1.1, 1.2 and 2.1 have all similar costs for CLARUS. In Protocol 2.1, the computational and communication costs for Alice and Bob are greater than in Protocols 1.1 and 1.2, because in the former protocol computations are performed over a field $\mathbb{F}_N$, where N is big and the representation of numbers takes 65 bytes instead of 8.

If we aggregate costs for all parties involved, Protocol 1.1 is the most efficient one, closely followed by Protocol 1.2 (note that these two protocols do not require any set-up). These results are consistent with the analytical comparison in Section 3.4.1.

3.6 Summary

We have presented several protocols (two of them new variants of already existent protocols, two of them new, and two benchmark protocols) for outsourc-

Table 3.8: **(Execution costs for Protocols 1.1, 1.2, 2.1, 3 and 4)**: Private vectors $\mathbf{x}$ and $\mathbf{y}$ have length $n = 20,640$. The representation of the numbers takes $\gamma = 8$ bytes. Elements in the field $\mathbb{F}_N$ used by the Paillier cryptosystem have 65 bytes. We indicate the absence of a cloud with "$-$". Times are given in seconds.

AWS	Computational cost (s.)				Communication cost (s.)			
	Alice	Bob	CLARUS	Charlie	Alice	Bob	CLARUS	Charlie
Prot. 1.1	8.3×10^{-3}	8.5×10^{-3}	3.3×10^{-5}	1.4×10^{-3}	0.05	0.05	0.40	0.06
Prot. 1.2	1.7×10^{-2}	1.6×10^{-2}	4.1×10^{-5}	1.4×10^{-3}	0.06	0.06	0.40	0.06
Prot. 2.1	34.1	121.6	9.1×10^{-5}	–	0.15	0.31	0.40	–
Prot. 3	0	0	1.5×10^{-3}	–	0.1	0.1	0.65	–
Prot. 4	0	–	0.8	–	2.2	–	2.62	–
CERSEI	Computational cost (s.)				Communication cost(s.)			
	Alice	Bob	CLARUS	Charlie	Alice	Bob	CLARUS	Charlie
Prot. 1.1	8.1×10^{-3}	7.8×10^{-3}	3.1×10^{-5}	1.4×10^{-3}	0.04	0.04	0.20	0.04
Prot. 1.2	5.7×10^{-3}	5.8×10^{-3}	3.7×10^{-5}	1.4×10^{-3}	0.04	0.04	0.20	0.04
Prot. 2.1	29.3	120.9	8.4×10^{-5}	–	0.07	0.08	0.20	–
Prot. 3	0	0	1.9×10^{-3}	–	0.02	0.01	0.23	–
Prot. 4	0	–	0.7	–	0.3	–	0.45	–

ing the computation of scalar products on sensitive numerical data vectors to honest-but-curious non-sharing clouds. Based on this operation, more complex data analyses can be performed, such as correlations and contingency tables. The goal is to minimize the amount of work that needs to be performed locally by the controller, who wants to use the cloud as much as possible to compute on her outsourced sensitive data. For the sake of flexibility and efficiency, we have considered a non-cryptographic method for data protection, such as data splitting, rather than the heavier fully homomorphic encryption (e.g. [34]). A distinguishing feature of our approach is that the outsourced data on which the clouds compute fully retain the utility of the original data, which entails added value with respect to outsourcing encrypted or otherwise gibberish data.

If clouds can be assumed not to share information (perhaps because they do not know each other), data splitting is probably the best choice, due to simplicity and flexibility. We have proposed four protocols to compute on split data.

We have provided complexity analyses and benchmarking for all proposed protocols, in order to show their computational advantages for the outsourcing controller. Further, we have provided experimental evidence that the new pro-

tocols take less effort from CLARUS than the benchmark protocols consisting of downloading and locally processing.

In this way, clouds are not only used to store sensitive data, but also to perform computations on these data in a privacy-aware manner. This is especially interesting for large sensitive data sets.

Chapter 4

Secure Matrix Product in Untrusted Clouds

4.1 Introduction

As shown in the previous chapter, outsourcing split data to one or more honest-but-curious CSPs can take advantage of the cloud to do most of the computations while preserving the privacy of the subjects to whom their data refer. In fact, such outsourcing brings several benefits like elimination of infrastructure costs (no software/hardware investments needed), flexibility (storage and computing power can scale depending on business growth) and energy saving.

Security is attained by splitting the data among several CSPs and, then, using secure protocols to process them. In data splitting, the most challenging step is to efficiently compute on the fragmented data when the computations involve more than one fragment. Chapter 3 shows how to carry out secure scalar products involving two fragments, which is one of the most challenging operations among fragments and which allows performing several statistical analyses (e.g. covariance matrix and contingency table). The security of those scalar products holds on the fact that the CSPs honestly fill their role in the protocols, and they do not share information with each other (perhaps because they do not even know each other). What happens if there are collusions among CSPs and/or CSPs own some external information on those data? For instance, if only one CSP with several accounts is used to store the data, it may happen that the CSP recognizes that two or more of those accounts belong to the same entity.

In this scenario, the CSP may try to pool together the data fragments it hold. Moreover, if the CSP has side knowledge on the outsourced data, it can try to re-identify the subjects to whom those data refer.

Existing literature proceeds by outsourcing encrypted data [46, 50, 67]. This comes at the price of using complex cryptographic schemes. In all those schemes the servers only see encrypted versions or shares of the actual data: the client must create these encrypted versions for each protocol execution and decrypt the final result.

In our solutions, *the privacy-protected sensitive data stored in the clouds are not encrypted* and *preserve some of the utility (that is, some statistical features) of the original data*. This allows us to make the most of the outsourced data, while ensuring that no original records can be re-created from the outsourced records. The outsourced data can be used for purposes other than computing scalar products or matrix products. This is a relevant difference with respect to related work on the outsourcing of algebraic computations (see Section 4.2 for more details). Specifically, the outsourced data preserve the mean and the standard deviation of attributes for the entire data set and even in subsets of it.

Following the architecture defined in the "CLARUS" European H2020 project [17] (within which this work has been carried out), we will assume a proxy located in a domain trusted by the data controller (e.g., a server in her company's intranet or a plug-in in her device) that implements security and privacy-enabling features towards the cloud service providers (see Section 2.2 for more details). We will call this trusted proxy CLARUS.

Contributions

In Chapter 3, we explore two new non-cryptographic protocols for the scalar product on split data and we also consider cryptographic protocols to compute on split data when the CSPs are honest-but-curious and they do not collude.

Here, we start from the conclusions of the previous chapter and break new ground by relaxing the non-sharing assumption. We present two non-cryptographic protocols that are sharing-resistant, even though they require substantial cloud storage (because they rely on data replication rather than splitting). The relaxation of the non-sharing assumption also requires adding privacy-preserving techniques such as data anonymization and data replication. Moreover, the procotols end up being more suitable for matrix products rather than for scalar products. In fact, some data anonymization methods require us to work on the entire data set.

The contributions in this chapter have been published in [21].

This chapter is organized as follows: Section 4.2 depicts the related work on outsourcing matrix and polynomial computations, and points out how our contributions differ from previous works. Section 4.3 presents a new protocol that can resist information sharing between CSPs but assumes the CSPs have no side knowledge about the original data set. Section 4.4 proposes another sharing-resistant protocol to compute the matrix product $\mathbf{X}^T\mathbf{X}$ that stays safe even if CSPs have information on the data set $\mathbf{X}$ but involves heavier computations. In Section 4.5, the computation and communication costs of all protocols described in Sections 4.3 and 4.4 are assessed and compared against a benchmark protocol consisting of the CLARUS proxy downloading the entire data set and locally computing on the downloaded data set. Section 4.6 presents the experimental results obtained by implementing the proposed protocols in a multi-cloud scenario. Finally, Section 4.7 lists some conclusions.

4.2 Related work

There is a substantial amount of literature devoted to outsourcing matrix and polynomial computations. We next review it and then highlight the differences with our approach.

In [5], a client securely outsources algebraic computations to one or several remote servers, in such a way that the server learns nothing about the client's private input or the result of the computation, and any attempted corruption of the answer by the server is detected with high probability. This scheme is based on multiparty secure computation via secret sharing. In [46], a client outsources a matrix inversion to an untrusted cloud, so that the cloud does not learn either the original or the inverted matrices. Furthermore, the protocol is resistant against a cheating cloud. In [47], the authors present a protocol to outsource multiplication of large matrices that can detect cheating by the server. The more recent contribution [67] follows the same line (outsourcing polynomials and matrix computations), but it focuses on public verifiability of the computation (any third party can verify its correctness, not only the client as in the previous proposals). This comes at the price of using more complex cryptographic schemes. In all the schemes reviewed in this paragraph, the server(s) only see(s) encrypted versions or shares of the actual data: the client must create these encrypted versions for each protocol execution and decrypt the final result.

Outsourcing matrix computations where the server computes on additively split matrices rather than encrypted matrices is considered in [50]. Even though

no encryption is used, the split versions of the matrics seen by the server do not preserve any of the statistical features of the original data (they look gibberish), so that no direct exploratory analyses can be performed on them.

A substantial difference between our proposals in this paper and the previous literature is that we assume that the cloud(s) compute on data that have been previously outsourced and privacy-protected *not only* to compute scalar products or matrix products on them. Specifically, the outsourced data preserve some of the utility of the original data, as explained in Section 4.1 above.

A second difference is that, as mentioned at the end of Section 2.2, in the CLARUS setting we can assume that CSPs are not malicious: hence, we do not need all the cryptographic apparatus of the above cryptographic proposals to detect cheating.

4.3 A sharing-resistant protocol in case of clouds without side knowledge

If the non-sharing assumption between clouds does not hold (see Section 2.2), then vertical partitioning alone cannot guarantee the privacy of the stored sensitive data. Here we show how the owner of a sensitive data set (represented by the CLARUS proxy in our case) can still use the computing power of several honest-but-curious, potentially sharing clouds to compute the matrix product $\mathbf{X^T X}$ of her original data set $\mathbf{X}$ (which contains the scalar products of the columns of $\mathbf{X}$). With the new protocol we propose, no information sharing between the clouds can determine the original data set or its matrix product. The latter is only seen by the owner CLARUS. We assume here that the CSPs do not have any side knowledge about the original data set (such as its statistical structure).

Let $\mathbf{X}$ be an $n \times m$ matrix representing an original data set with n records and m attributes, where $n \gg m$. In this case we are interested in computing the matrix product $\mathbf{X^T X}$. The protocol is depicted in Figure 4.1 and it runs in two phases: set-up and matrix product computation. The set-up phase needs to be run only once. The steps of the protocol are detailed next.

Protocol 1.
Set-up phase (data storage):

1. *CLARUS (the owner of* $\mathbf{X}$*) does:*

   (a) *Choose a random invertible* $m \times m$ *matrix* $\mathbf{P}$.

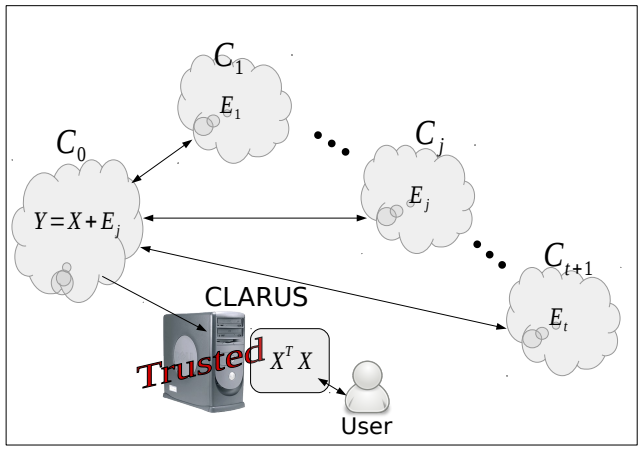

Figure 4.1: CSPs that share information but do not have side knowledge on the original data set. Despite sharing, the lack of knowledge of the CSPs on $\mathbf{X}$ prevents them from discovering with certainty which error matrix they ought to subtract from $\mathbf{Y}$ to get $\mathbf{X}$.

(b) *Send* $\mathbf{X'} = \mathbf{XP}$ *to* t *clouds* $C_1, \ldots, C_t$.

(c) *Delete* $\mathbf{X'}$ *and* $\mathbf{P}$.

2. *For* $i = 1$ *to* t, *each cloud* C_i *does:*

(a) *Compute a random* $n \times m$ *matrix* $\mathbf{E_i}$ *such that it is orthogonal to* $\mathbf{X'}$, *that is, such that*

$$(\mathbf{X'})^{\mathbf{T}}\mathbf{E_i} = (\mathbf{E_i})^{\mathbf{T}}\mathbf{X'} = \mathbf{0}.$$

and send $\mathbf{E_i}$ *to CLARUS.*

(b) *Compute* $(\mathbf{E_i})^{\mathbf{T}}\mathbf{E_i}$ *and send it to CLARUS.*

3. *CLARUS does:*

(a) *Randomly select* $j \in_R \{1, \ldots, t\}$.

(b) *Read* $\mathbf{E_j}$ *and* $(\mathbf{E_j})^{\mathbf{T}}\mathbf{E_j}$, *and discard all communications from clouds other than* C_j.

(c) *Compute* $\mathbf{Y} = \mathbf{X} + \mathbf{E_j}$.

(d) Store $\mathbf{Y}$ *at cloud* C_0.

(e) Delete $\mathbf{X}, \mathbf{E_j}$ *and* $\mathbf{Y}$.

Matrix product computation:

1. *Cloud* C_0 *computes* $(\mathbf{Y})^\mathbf{T}\mathbf{Y}$ *and returns it to CLARUS.*
2. *CLARUS just subtracts*

$$\mathbf{Y^T Y} - (\mathbf{E_j})^\mathbf{T}\mathbf{E_j} = \mathbf{X^T X}. \tag{4.1}$$

We have the following correctness and security results.

Proposition 4 (Correctness)**.** *Protocol 1 is correct.*

Proof. If $\mathbf{P}$ is not invertible, then there exists at least one vector $\mathbf{b}$ in $\text{Ker}(\mathbf{X}')$ such that $\mathbf{X}'\mathbf{b} = \mathbf{0}$ (but it is not necessary that $\mathbf{Xb} \neq \mathbf{0}$). We can always take $\mathbf{P}$ invertible without losing security (see Section 4.3.5). The correctness of Equation (4.1) follows from

$$\begin{aligned}(\mathbf{Y})^\mathbf{T}\mathbf{Y} &= (\mathbf{X})^\mathbf{T}\mathbf{X} + (\mathbf{E_j})^\mathbf{T}\mathbf{E_j} + (\mathbf{X})^\mathbf{T}\mathbf{E_j} + (\mathbf{E_j})^\mathbf{T}\mathbf{X} \\ &= (\mathbf{X})^\mathbf{T}\mathbf{X} + (\mathbf{E_j})^\mathbf{T}\mathbf{E_j},\end{aligned}$$

where in the last step we use orthogonality between $\mathbf{E_j}$ and $\mathbf{X}$, which in turn follows from orthogonality between $\mathbf{E_j}$ and $\mathbf{X}'$. □

In Section 4.3.2, we discuss how to create $\mathbf{E_j}$ so that $\mathbf{X}'$ preserves the means and the variance of $\mathbf{X}$.

Proposition 5 (Security)**.** *If the CSPs are honest-but-curious and share information, but have no side knowledge on* $\mathbf{X}$*, Protocol 1 is secure in the sense that their probability of guessing the correct* $\mathbf{X}$ *and* $\mathbf{X^T X}$ *is at most* $1/t$.

Proof. Let us examine what the clouds receive. During the matrix product computation, none of the clouds receives any further information. They only receive information during the set-up.

In Step 1 of the set-up, clouds $C_1, \ldots, C_t$ receive matrix $\mathbf{X}'$, which is the original data set multiplied by a random matrix generated by CLARUS; hence $\mathbf{X}'$ leaks no information on $\mathbf{X}$ to the clouds.

In Step 3 of the set-up, cloud C_0 receives $\mathbf{Y}$, which is the original data set plus a random matrix $\mathbf{E_j}$ generated by one of the clouds. However, the sharing clouds

cannot find out which of the noise matrices $\mathbf{E_1}, \ldots, \mathbf{E_t}$ they have generated is the one that has been used to obtain $\mathbf{Y}$. The clouds could try an exhaustive method: subtract all the possible $\mathbf{E_i}$ from $\mathbf{Y}$, i.e. $\mathbf{S_i} = \mathbf{Y} - \mathbf{E_i}$, for $i = 1, \ldots, t$. Note that, since all $\mathbf{E_i}$ are random (we assume the clouds honestly follow the protocol), all t matrices $\mathbf{S_i}$ are different with overwhelming probability. But the clouds have no way to guess which $\mathbf{S_i}$ equals the original data set $\mathbf{X}$, because by assumption they neither know $\mathbf{X}$ nor have any side information on $\mathbf{X}$. Hence, even by sharing information, the clouds have no better strategy than randomly picking one of $S_1, \ldots, S_t$; the probability that they hit the correct $\mathbf{X}$ is $1/t$.

On the other hand, to obtain $\mathbf{X^T X}$, Alice also needs to guess the correct $\mathbf{E_j}$ in Equation (4.1), which happens with probability $1/t$. □

4.3.1 Computing the attribute means and standard deviations

In addition to leveraging the clouds to compute $\mathbf{X^T X}$, CLARUS can use them to compute the attribute means and standard deviations. Note that, according to Equation (3.1), the vector of means and the vector of standard deviations are needed to compute the sample correlation matrix of the data set. The following additions need to be done to Protocol 1 to compute the means:

- In Step 2b of Protocol 1 (set-up phase), each cloud C_i computes the vector of sums of the columns of $\mathbf{E_i}$, say $\mathbf{s_i^T} = (\mathbf{s_{i1}}, \ldots, \mathbf{s_{im}})$ and the vector of the squared sums of the columns of $\mathbf{E_i}$, that is, $\mathbf{ss_i^T} = ((\mathbf{s_{i1}})^\mathbf{2}, \ldots, (\mathbf{s_{im}})^\mathbf{2})$; then C_i sends $\mathbf{s_i^T}$ and $\mathbf{ss_i^T}$ along with $\mathbf{E_i}$ and $(\mathbf{E_i})^\mathbf{T}\mathbf{E_i}$ to CLARUS.
- In Step 1 of Protocol 1 (matrix product computation), C_0 computes the vector of sums of the columns of $\mathbf{Y}$, say $\mathbf{s_Y^T} = (\mathbf{s_{Y1}}, \ldots, \mathbf{s_{Ym}})$, and the vector of squared sums of the columns of $\mathbf{Y}$, that is, $\mathbf{ss_Y^T} = ((\mathbf{s_{Y1}})^\mathbf{2}, \ldots, (\mathbf{s_{Ym}})^\mathbf{2})$ and returns both vectors to CLARUS along with $(\mathbf{Y})^\mathbf{T}\mathbf{Y}$.
- In Step 2 of Protocol 1 (matrix product computation), CLARUS computes the vector of sample means of $\mathbf{X}$, say $(\hat{\mu}_X)^T = (\hat{\mu}_{X1}, \ldots, \hat{\mu}_{Xm})$ from the partial results obtained from the clouds as

$$\hat{\mu}_X = \frac{\mathbf{s_Y^T} - \mathbf{s_j^T}}{n}.$$

  Additionally, CLARUS computes $\mathbf{ss_X^T} = ((\mathbf{s_{X1}})^\mathbf{2}, \ldots, (\mathbf{s_{Xm}})^\mathbf{2})$ from the partial results obtained from the clouds as

$$\mathbf{ss_X^T} = \mathbf{ss_Y^T} - \mathbf{2}(\mathbf{s_{j1}s_{Y1}}, \ldots, \mathbf{s_{jm}s_{Ym}}) + \mathbf{ss_j^T}.$$

Finally, CLARUS computes the vector of standard deviations of $\mathbf{X}$ as

$$\hat{\sigma}_X^T = \left(\sqrt{\frac{(s_{X1})^2}{n} - (\mu_{X1})^2}, \ldots, \sqrt{\frac{(s_{Xm})^2}{n} - (\mu_{Xm})^2} \right).$$

The additional computations to be performed by CLARUS may seem substantial, but they are only $O(m)$ and, since $m \ll n$, they are much less than the additional $O(mn)$ computations performed by the clouds. On the other hand, the correctness of those additional computations follows from direct algebraic verification.

Security The security of the extended version of Protocol 1 to compute the attribute means and standard deviations is the same as the security of the basic Protocol 1. Clearly, the proof of Proposition 5 also holds for the extended version of Protocol 1, because the computations and communications added in the extended version do not result in the clouds *receiving* any additional information (all additional communications are directed to CLARUS).

4.3.2 Preserving the attribute means in the masked data set

It may be desirable to preserve the means of attributes of $\mathbf{X}$ in the data set $\mathbf{Y}$ stored in the cloud. To that end, matrices $\mathbf{E_i}$, $i = 1, \ldots, t$ must be such that each of their columns adds to 0. We show how to obtain such matrices $\mathbf{E_i}$.

Given a random $n \times m$ matrix $\mathbf{B_i}$ such that $(\mathbf{X'})^\mathbf{T}\mathbf{B_i} = (\mathbf{B_i})^\mathbf{T}\mathbf{X'} = \mathbf{0}$, take any two of the m column vectors of $\mathbf{B_i}$, say $\mathbf{b_j}$ and $\mathbf{b_k}$, such that their respective components do not add to zero. This may be infeasible in two cases:

- If all columns of $\mathbf{B_i}$ have their respective components adding to 0, then we can take $\mathbf{E_i} = \mathbf{B_i}$ and stop.
- If all columns of $\mathbf{B_i}$ except one have their components adding to 0, then we have a problem and we must choose a new random matrix $\mathbf{B_i}$.

Divide all components of $\mathbf{b_j}$ by the sum of the components, in order to obtain a vector $\hat{\mathbf{b}}_j$ such that its components add to 1; do the same to $\mathbf{b_k}$ and get $\hat{\mathbf{b}}_k$. By construction, it holds that $(\mathbf{X'})^\mathbf{T}\hat{\mathbf{b}}_\mathbf{j} = \mathbf{0}$, $(\hat{\mathbf{b}}_j)^T\mathbf{X'} = \mathbf{0^T}$, $(\mathbf{X'})^\mathbf{T}\hat{\mathbf{b}}_\mathbf{k} = \mathbf{0}$, $(\hat{\mathbf{b}}_k)^T\mathbf{X'} = \mathbf{0^T}$. Then build

$$\mathbf{E_i} = \mathbf{B_i} - \mathbf{s_1}\mathbf{M_1} - \ldots - \mathbf{s_m}\mathbf{M_m}, \tag{4.2}$$

where $\mathbf{M_l}$ for $l \neq j$ is a matrix having all columns equal to $\mathbf{0}$ except the l-th column which is equal to $\hat{\mathbf{b}}_j$, $\mathbf{M_j}$ is a matrix having all columns equal to $\mathbf{0}$ except the j-th column which is equal to $\hat{\mathbf{b}}_k$, and, for $l = 1, \ldots, m$, s_l is the sum of the l-th column of $\mathbf{B_i}$. Clearly, each column of $\mathbf{E_i}$ adds to 0 and, by construction, $(\mathbf{X}')^{\mathbf{T}}\mathbf{E_i} = (\mathbf{E_i})^{\mathbf{T}}\mathbf{X}' = \mathbf{0}$.

Security The error matrices obtained with Equation (4.2) are still random and can be different from each other. On the other hand, these additional computations are done separately by each cloud. Thus, they do not require exchange of information and hence do not affect the security of the basic protocol, stated in Proposition 5.

4.3.3 Preserving means and correlations in subdomains

One may wish to enable the computation of sample correlations in the cloud for subdomains of $\mathbf{X}$. Also, one may wish to preserve the means of attributes for records in the subdomain. For example, consider a medical data set, in which we define the following 20 subdomains: women aged 0 to 9, women aged 10 to 19, ..., women aged 90+, men aged 0 to 9, men aged 10 to 19, ..., men aged 90+.

In the example, one can split $\mathbf{X}$ into 20 data subsets $\mathbf{X}(\mathbf{1}), \ldots, \mathbf{X}(\mathbf{20})$. Then Protocol 1 (set-up phase) with the improvements described in Sections 4.3.1 and 4.3.2 is separately run for each $\mathbf{X}(\mathbf{i})$. It holds that the corresponding $\mathbf{Y}(\mathbf{i})$ preserves attribute means and Protocol 1 (matrix product computation) can be used to compute correlations within each $\mathbf{X}(\mathbf{i})$.

The price paid for considering subdomains is that it may no longer hold that $n_i \gg m$, where n_i is the number of records of $\mathbf{X}(\mathbf{i})$. This reduces the computational gain of using Protocol 1 and also the privacy of the $\mathbf{X}(\mathbf{i})$, because the number of degrees of freedom is reduced. However, as long as all n_i are substantially larger than m, considering subdomains is acceptable.

Security We merely subdivide the problem into disjoint subproblems. In each subproblem, security is guaranteed as for Proposition 5.

4.3.4 Using a single cloud

If one assumes all clouds may share information, then from the security point of view the situation is equivalent to using a single cloud. We could then take

$C_0 = C_1 = \cdots = C_t$ in Protocol 1. While security is not affected, there are performance pros and cons in using a single cloud:

- *Pros.* Using a single cloud is simpler and CLARUS saves communication at Step 1b, because $\mathbf{X}'$ only needs to be sent to one cloud. Also, one can modify the protocol at Step 1 for the cloud to compute $\mathbf{Y^T Y} - (\mathbf{E_i})^\mathbf{T}\mathbf{E_i}$ for $i = 1, \ldots, t$, so that at Step 2, CLARUS only needs to pick $\mathbf{Y^T Y} - (\mathbf{E_j})^\mathbf{T}\mathbf{E_j}$ as $\mathbf{X^T X}$ without doing any computation. A similar modification could be done to the additional computations described in Section 4.3.1: the cloud could be asked to provide all candidate vectors of means and standard deviations under the t different error matrices, so that the only job left to CLARUS would be to pick the right vectors.

- *Cons.* When all the work needs to be done by a single cloud, the overall computation is likely to take longer (except if the cloud has a great computational power and/or is very efficient at parallelizing). Also, CLARUS can no longer discard any communication (as it did in Step 3b when communication coming from clouds other than C_j was discarded). Further, if we require the cloud to provide all $\mathbf{Y^T Y} - (\mathbf{E_i})^\mathbf{T}\mathbf{E_i}$ for $i = 1, \ldots, t$, plus all candidate vectors of means and standard deviations under all error matrices, communication increases even more.

So, all in all, there is no clear advantage in using a single cloud: while CLARUS saves computation, it incurs more communication costs.

4.3.5 Computation of the invertible matrix $\boldsymbol{P}$

The probability of obtaining an invertible matrix $\boldsymbol{P}$ (needed in the set-up phase of Protocol 1) if we randomly choose its elements depends on the cardinality of the field we are working in.

Lemma 1. *Let $\mathbb{K}$ be a field where the elements of a random $m \times m$ matrix $\boldsymbol{P}$ are chosen. If $|\mathbb{K}| = N$, then*

$$\Pr(\boldsymbol{P} \textit{ is invertible}) > 1 - \frac{1}{N-1} + \frac{1}{(N-1)N^m}. \tag{4.3}$$

Proof. $\boldsymbol{P}$ has m column vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$. In order for $\boldsymbol{P}$ to be invertible, they must be linearly independent. We have:

- $\Pr(\boldsymbol{v}_1 \text{ is the null vector}) = \frac{1}{N^m}$;

- $\Pr(\boldsymbol{v}_2 \text{ is linear combination of } \boldsymbol{v}_1) \leq \frac{N}{N^m}$;
- $\Pr(\boldsymbol{v}_3 \text{ is linear combination of } \boldsymbol{v}_1, \boldsymbol{v}_2) \leq \frac{N^2}{N^m}$;
- ...
- $\Pr(\boldsymbol{v}_t \text{ is linear combination of } \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1}) \leq \frac{N^{m-1}}{N^m}$.

Hence, $\Pr(\boldsymbol{P} \text{ is not invertible}) \leq \sum_{i=0}^{m-1} \frac{N^i}{N^m} = \frac{N^m-1}{(N-1)N^m}$ and then

$$\Pr(\boldsymbol{P} \text{ is invertible}) > 1 - \frac{N^m - 1}{(N-1)N^m} = 1 - \frac{1}{N-1} + \frac{1}{(N-1)N^m}.$$

□

We suggest to obtain $\boldsymbol{P}$ by randomly picking an $m \times m$ matrix over $\mathbb{K}$ and trying to invert it. This will work with probability lower-bounded by Equation (4.3). As N grows, the lower bound approaches 1, so if we take a sufficiently large $\mathbb{K}$, a single attempt is very likely to suffice to find $\boldsymbol{P}$. If the first attempt fails, one can always try again.

4.3.6 Orthogonal complement of a matrix

In Protocol 1 (set-up phase), each cloud C_i needs to find an $n \times m$ matrix $\boldsymbol{E}_i$ orthogonal to $\boldsymbol{X}'$. Following [45], we suggest to find $Ort(\boldsymbol{X}')$ by using the QR-decomposition of $\boldsymbol{X}'$, where $\boldsymbol{Q}$ is an $(n \times n)$ orthonormal matrix and $\boldsymbol{R}$ is an $n \times m$ upper-triangular matrix. If we split $\boldsymbol{Q} = [\boldsymbol{Q}_1, \boldsymbol{Q}_2]$, where $\boldsymbol{Q}_1$ is $(n \times m)$ and $\boldsymbol{Q}_2$ is composed of $n - m$ orthogonal vectors to $\boldsymbol{X}'$, it is possible to select m columns out of $\boldsymbol{Q}_2$ to generate $\boldsymbol{E}_i$. Note that $n \gg m$, so m columns can be selected out of $n - m$ and each cloud is likely to select a different set of m columns.

If $n - m > m$, but it does not hold that $n \gg m$, then m random linear combinations of the columns of $\boldsymbol{Q}_2$ are preferable to using the columns themselves: since Protocol 1 (set-up phase) will be used by all the clouds $C_1, C_2, \ldots, C_t$, we want to avoid the possibility that different clouds obtain the same $\boldsymbol{E}_i$.

4.4 A sharing-resistant protocol robust against cloud side knowledge

In Protocol 1, if clouds have side knowledge on the statistical structure of $\mathbf{X}$ (for example, correlations between attributes, etc.), they can discard most of

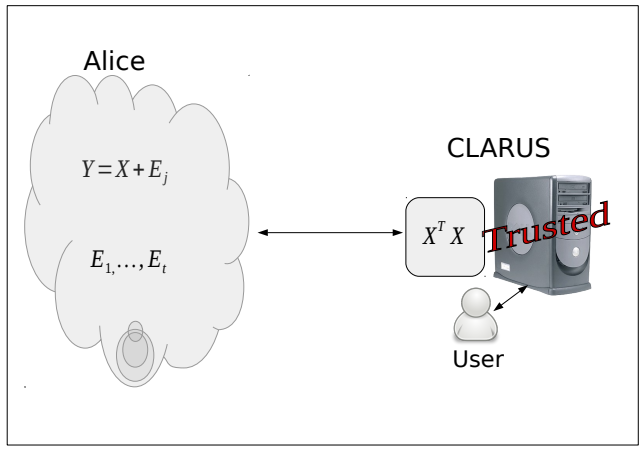

Figure 4.2: CSPs that share information and have side knowledge about the original data set are treated as if there was a single CSP. An anonymized version $\mathbf{Y} = \mathbf{X}+\mathbf{E_j}$ of the original data set $\mathbf{X}$ is stored in the cloud, together with several plausible error matrices $\mathbf{E_1}, \ldots, \mathbf{E_t}$ under the side information on $\mathbf{X}$ known to the CSP. The CSP does not know which error matrix should be subtracted from $\mathbf{Y}$ to recover $\mathbf{X}$.

the "false" error matrices and focus on the (possibly unique) error matrix $\mathbf{E_j}$ such that $\mathbf{Y} - \mathbf{E_j}$ matches their side knowledge on $\mathbf{X}$. This would allow them to recover $\mathbf{X}$. We propose an alternative solution that addresses this issue but requires more set-up computation from CLARUS.

By the argument given in Section 4.3.4, a set of clouds sharing information is equivalent to a single cloud from a security point of view. Whereas Protocol 1 could work the same way with one or several clouds, in this section we present a new protocol designed for a single cloud. The proposed protocol is depicted in Figure 4.2 and its steps are detailed next.

Protocol 2.
Set-up phase (data storage):

CLARUS does:

1. *Anonymize the whole data set* $\mathbf{X}$ *using a randomized statistical disclosure control method to obtain a safe matrix* $\mathbf{Y}$ *and send* $\mathbf{Y}$ *to cloud Alice.*

2. *Randomly select* $j \in_R \{1, \ldots, t\}$.

3. *Let* $\mathbf{E_j} = \mathbf{Y} - \mathbf{X}$ *and let* $\mathbf{E_1}, \ldots, \mathbf{E_{j-1}}, \mathbf{E_{j+1}}, \ldots, \mathbf{E_t}$ *be fake but plausible error matrices (in the sense that all* $\mathbf{Y} - \mathbf{E_i}$ *are plausible with the side information on* $\mathbf{X}$ *known to the clouds, for* $i = 1, \ldots, t$*).*

4. *Send* $\mathbf{E_1}, \ldots, \mathbf{E_t}$ *to Alice.*

5. *Delete* $\mathbf{X}$, $\mathbf{Y}$ *and* $\mathbf{E_1}, \ldots, \mathbf{E_t}$.

Matrix product computation:

1. *Alice computes* $\mathbf{Y^T Y}$.

2. *For* $i = 1, \ldots, t$*, Alice computes:*

$$(\mathbf{Y} - \mathbf{E_i})^\mathbf{T}(\mathbf{Y} - \mathbf{E_i}) = \mathbf{Y^T Y} + (\mathbf{E_i})^\mathbf{T}\mathbf{E_i} - \mathbf{Y^T}\mathbf{E_i} - (\mathbf{E_i})^\mathbf{T}\mathbf{Y}. \qquad (4.4)$$

3. *For* $i = 1, \ldots, t$*, Alice sends* $(\mathbf{Y} - \mathbf{E_i})^\mathbf{T}(\mathbf{Y} - \mathbf{E_i})$ *to CLARUS.*

4. *CLARUS picks* $(\mathbf{Y} - \mathbf{E_j})^\mathbf{T}(\mathbf{Y} - \mathbf{E_j})$ *as* $\mathbf{X^T X}$.

Note that, to compute the products in Step 2, the cloud needs to know $\mathbf{Y}$ and all error matrices. This is why this protocol only works for a single cloud (or for clouds that share all information). The randomized statistical disclosure control (SDC) methods usable to obtain $\mathbf{Y}$ from $\mathbf{X}$ should be such that the matrix $\mathbf{Y} - \mathbf{X}$ looks random. Possible options include additive noise, multiplicative noise, synthetic data, etc. (see [40] for more details).

A key issue to Protocol 2 is how to obtain fake plausible error matrices $\mathbf{E_1}, \ldots, \mathbf{E_{j-1}}, \mathbf{E_{j+1}}, \ldots, \mathbf{E_t}$. We propose to generate these fake error matrices as anonymized versions of the true $\mathbf{E_j}$. As above, the anonymization method should be such that the difference matrix between $\mathbf{E_j}$ and any anonymized version of it looks random. However, even if random-looking, the differences between $\mathbf{E_j}$ and its anonymized versions should be relatively small, so that the following two conditions are satisfied:

1. All the $\mathbf{E_i}$'s are similar enough to yield data sets $\mathbf{Y} - \mathbf{E_i}$ plausible under the side knowledge on $\mathbf{X}$ held by the clouds;

2. The matrices $\mathbf{Y} - \mathbf{E_i}$ for $i \neq j$ are not too similar to $\mathbf{Y} - \mathbf{E_j} = \mathbf{X}$.

Introducing random perturbations that achieve a good trade-off between protection and preservation of the structure of data is precisely the purpose of the aforementioned SDC methods. Note that, whereas when anonymizing $\mathbf{X}$ into $\mathbf{Y}$ the only purpose was protection (and perturbations could be large), when anonymizing $\mathbf{E_j}$ into $\mathbf{E_1}, \ldots, \mathbf{E_{j-1}}, \mathbf{E_{j+1}}, \ldots, \mathbf{E_t}$, parameters for the SDC methods yielding smaller perturbations must be chosen to attain the previously mentioned protection/preservation trade-off. We can now state the following security result:

Proposition 6 (Security)**.** *If $\mathbf{Y} - \mathbf{X}$ looks random, all $\mathbf{Y} - \mathbf{E_i}$ for $i = 1, \ldots, t$ are plausible under the CSP's side knowledge on $\mathbf{X}$, and $\mathbf{Y} - \mathbf{E_i}$ for $i = 1, \ldots, t$, $i \neq j$ are not too similar to $\mathbf{Y} - \mathbf{E_j} = \mathbf{X}$, then Protocol 2 is secure in the sense that the probability of the CSP guessing the correct $\mathbf{X}$ and $\mathbf{X^T X}$ is at most $1/t$.*

Proof. Let us examine what the cloud Alice receives. During the matrix product computation, Alice receives no further information. She only receives information during the set-up.

In Step 1 of the set-up, Alice receives an anonymized version $\mathbf{Y}$ of the original data set $\mathbf{X}$. By assumption, $\mathbf{Y} - \mathbf{X}$ looks random, so $\mathbf{Y}$ does not leak $\mathbf{X}$.

Finally, in Step 4 of the set-up, Alice receives error matrices $\mathbf{E_1}, \ldots, \mathbf{E_t}$. By assumption, Alice's side knowledge does not allow her to single out $\mathbf{Y} - \mathbf{E_j}$ (which is equal to $\mathbf{X}$) from $\mathbf{Y} - \mathbf{E_i}$ for $i = 1, \ldots, t$. On the other hand, also by assumption, $\mathbf{Y} - \mathbf{E_i}$ for $i = 1, \ldots, t$, $i \neq j$ are not too similar to $\mathbf{Y} - \mathbf{E_j}$.

Hence, Alice has no better strategy than randomly picking one of $\mathbf{Y} - \mathbf{E_i}$ for $i = 1, \ldots, t$ as $\mathbf{X}$; her probability of hitting the correct $\mathbf{X}$ is $1/t$.

On the other hand, to obtain $\mathbf{X^T X}$, Alice also needs to guess the correct $\mathbf{E_j}$, which happens with probability $1/t$. □

4.4.1 Computing the attribute means and standard deviations

If attribute means and standard deviations are to be computed (for example, for CLARUS to obtain the sample correlation matrix of the data set), then some computations need to be added to Protocol 2:

- In Step 2 of Protocol 2 (matrix product computation), for $i = 1, \ldots, t$, Alice computes in an analogous way as in Section 4.3.1:
  1. $\mathbf{s_Y^T}$ and $\mathbf{ss_Y^T}$ corresponding to $\mathbf{Y}$;
  2. $\mathbf{s_i^T}$ and $\mathbf{ss_i^T}$ corresponding to $\mathbf{E_i}$;

3. $\hat{\mu}_{Y-E_i} = (\mathbf{s_Y^T} - \mathbf{s_i^T})/\mathbf{n}$;
4. $\mathbf{ss_{Y-E_i}} = ((\mathbf{s_{Y-E_i,1}})^\mathbf{2}, \ldots, (\mathbf{s_{Y-E_i,m}})^\mathbf{2})$ as
$$\mathbf{ss_{Y-E_i}} = \mathbf{ss_Y^T} - \mathbf{2}(\mathbf{s_{i1}s_{Y1}}, \ldots, \mathbf{s_{im}s_{Ym}}) + \mathbf{ss_i^T}.$$
5. The vector of standard deviations of $\mathbf{Y - E_i}$:
$$\hat{\sigma}_{Y-E_i}^T = \left(\sqrt{\frac{(s_{Y-E_i,1})^2}{n} - (\mu_{Y-E_i,1})^2}, \ldots, \sqrt{\frac{(s_{Y-E_i,m})^2}{n} - (\mu_{Y-E_i,m})^2} \right).$$

- In Step 3 of Protocol 2, Alice sends $\hat{\mu}_{Y-E_i}$ and $\hat{\sigma}_{Y-E_i}$ to CLARUS, for $i = 1, \ldots, t$.
- In Step 4 of Protocol 2, CLARUS picks $\hat{\mu}_{Y-E_j}$ as $\hat{\mu}_X$ and $\hat{\sigma}_{Y-E_j}$ as $\hat{\sigma}_X$.

Security The security of the extended version of Protocol 2 to compute the attribute means and standard deviations is the same as the security of the basic Protocol 2. The proof of Proposition 6 also holds for the extended version of Protocol 2, because the computations and communications added in the extended version do not result in Alice *receiving* any additional information.

4.4.2 Preserving means and other statistics

In Section 4.3.2 above, we discussed how the data set $\mathbf{Y}$ stored in the cloud under Protocol 1 could exactly preserve the attribute means of $\mathbf{X}$. This is even easier for Protocol 2, where it would be sufficient for all columns of all error matrices $\mathbf{E_1}, \ldots, \mathbf{E_t}$ to add to zero. Given $\mathbf{E_j}$ whose columns add to zero, it is easy to generate anonymized versions of it (the other error matrices $\mathbf{E_1}, \ldots, \mathbf{E_{j-1}}$, $\mathbf{E_{j+1}}, \ldots, \mathbf{E_t}$) whose columns also add to zero (see [40]).

Furthermore, we can do more than preserving means. In the discussion above, we have not required the anonymized $\mathbf{Y}$ to preserve the statistical properties of $\mathbf{X}$. If the parameters of the SDC method used to transform $\mathbf{X}$ into $\mathbf{Y}$ are carefully chosen, many statistical properties of $\mathbf{X}$ can be exactly or approximately preserved by $\mathbf{Y}$. The specific preserved properties and whether preservation is only approximate depend on the particular SDC method used (see [40]).

4.5 Comparison among methods

If we relax the non-sharing assumption, then we need a benchmark which no longer relies on data splitting (note that Protocol 3 splits data among Alice and

Bob). Therefore, the protocols described in Sections 4.3 and 4.4 are compared here against a benchmark solution that consist of CLARUS downloading the data from the cloud or clouds and computing locally. In this setting, CLARUS first encrypts the data set $\mathbf{X}$ (containing in particular vectors $\mathbf{x}$ and $\mathbf{y}$), stores it in the cloud and, when it needs to compute a scalar product, it downloads and decrypts the data set, as detailed in the following protocol:

Protocol 3.

Set-up phase:

1. *CLARUS encrypts the original data set* $\mathbf{E} = \mathbf{Enc}(\mathbf{X})$
2. *CLARUS sends* $\mathbf{E}$ *to a cloud Alice for remote storage, and deletes* $\mathbf{X}$ *from local storage.*

Matrix product computation:

3. *CLARUS requests* $\mathbf{E}$ *from Alice, decrypts* $\mathbf{X} = \mathbf{Dec}(\mathbf{E})$ *and performs the computation* $(\mathbf{X})^{\mathbf{T}}\mathbf{X}$.

Encryption and decryption can be performed using a fast symmetric cryptosystem, such as the Advanced Encryption Standard (AES), which takes time linear in the number of records/vector components n, and generates cyphertexts that are similar in size to the corresponding plaintexts.

We now evaluate the computational cost for Alice, Bob and CLARUS and the total computational cost under each protocol. In particular, operations that do not need to be repeated each time the protocol is executed, specifically the generation of the random and error matrices $\mathbf{E_j}$ in Protocols 1 and 2, respectively, are separately counted as set-up costs.

Assuming that the clouds have unlimited storage, it is reasonable to assume that they can store any random matrices that may need to be reused. In contrast, we do not assume unlimited storage at CLARUS; therefore, we assume the proxy just stores the random seeds and (re)generates random matrices or vectors when needed. Also, the cost of a communication is associated both to the sender (who needs to send the data) and to the receiver (who needs to read the data). We use a parameter γ to represent the maximum length of the numbers in the vectors and matrices used in the protocols. We consider that, whenever possible, the participants send the seeds of random vectors and matrices, rather than the vectors and matrices themselves. If communications are very fast and/or vectors are very short, sending the vectors rather than the seeds might be preferable (see related experimental results in Section 4.6).

Table 4.1: **(Costs for Protocols 1 and 3)**: $\mathbf{X}$ is the $n \times m$ matrix (with $n \gg m$) containing the original data set. γ represents the maximum length of the numbers in the vectors and matrices, and t is the number of clouds involved in the protocol.

	Protocol 1			Protocol 3	
	Alice (C_0)	Bob (C_j)	CLARUS	Alice	CLARUS
Set-up computation	0	$3nm^2$ prod.	nm^2 prod. $+m^2$ RNDgen.	0	nm AESencr.
Set-up communication	nm	$(2nm + m^2)\gamma$	$((t+2)nm + m^2)\gamma$	$nm\gamma$	$nm\gamma$
Long-term storage	$nm\gamma$	$nm\gamma$	$m^2\gamma$	$nm\gamma$	$nm\gamma$
Temporary storage	0	0	$2m^2\gamma$	0	$(nm + m^2)\gamma$
Matrix product computation	nm^2 prod.	0	m^2 subtr.	0	nm AESdecr. $+nm^2$ prod.
Matrix product communication	$m^2\gamma$	0	$m^2\gamma$	$nm\gamma$	$nm\gamma$

4.5.1 Comparison for the sharing-resistant alternative without side knowledge

Table 4.1 shows all the costs for Protocols 1 and 3. Alice represents C_0, the cloud storing the anonymized data set $\mathbf{Y}$. The other clouds $C_1, \ldots, C_m$ perform similar amounts of computation and so we represent all of them by Bob. Protocol 1 needs a set-up phase, in which:

- CLARUS needs to compute the random invertible $m \times m$ matrix $\mathbf{P}$ (as described in Section 4.3.5). Then it must multiply $\mathbf{X}$ times $\mathbf{P}$ and add $\mathbf{E_j}$ to $\mathbf{X}$.
- Each cloud needs to compute the orthogonal complement of $\mathbf{Y}$ (as described in Section 4.3.6), which takes $2nm^2$ products (using the Gram-Schmidt algorithm for QR decomposition), plus $(\mathbf{E_j})^\mathbf{T}\mathbf{E_j}$, which takes nm^2 products.

The set-up phase is performed just once, unless $\mathbf{X}$ is modified (in which case it needs to be repeated).

Regarding storage, in Protocol 1 each cloud stores one $n \times m$ matrix ($\mathbf{X}'$ for $C_1, \ldots, C_t$, and $\mathbf{Y}$ for C_0), whereas CLARUS stores three $m \times m$ matrices ($(\mathbf{Y})^\mathbf{T}\mathbf{Y}$ and $(\mathbf{X})^\mathbf{T}\mathbf{X}$ as temporary storage and $(\mathbf{E_j})^\mathbf{T}\mathbf{E_j}$ as long-term storage). As to communication, at set-up CLARUS sends one $n \times m$ matrix to clouds $C_1, \ldots, C_t$ (this can be done in a single message if using broadcast or if using a single cloud to do all computations) and one $n \times m$ matrix $\mathbf{Y}$ to C_0; each of

Table 4.2: **(Costs for Protocol 2)**: γ represents the maximum length of the numbers in the vectors and matrices. "anon" stands for the anonymization cost of a value and "read" stands for the cost of reading a value.

	Protocol 2		Protocol 3	
	Alice	CLARUS	Alice	CLARUS
Set-up computation	0	$(t+1)nm$ anon.	0	nm AESencr.
Set-up communication	$(t+1)nm\gamma$	$(t+1)nm\gamma$	$nm\gamma$	$nm\gamma$
Long-term storage	$(t+1)nm\gamma$	γ	$nm\gamma$	$nm\gamma$
Temporary storage	$(3t+1)m^2$	$m^2\gamma$	0	$(nm+m^2)\gamma$
Matrix product computation	$(2t+1)nm^2$ prod. $+tm^2$ reads	0	0	nm AESdecr. $+nm^2$ prod.
Matrix product communication	$(3t+1)m^2\gamma$	$(3t+1)m^2\gamma$	$nm\gamma$	$nm\gamma$

$C_1, \ldots, C_t$ returns one $n \times m$ matrix and one $m \times m$ matrix to CLARUS; C_0 returns one $m \times m$ matrix to CLARUS.

To compute the matrix product (after set-up), in Protocol 1 C_0 needs to compute $(\mathbf{Y})^{\mathbf{T}}\mathbf{Y}$, which takes nm^2 products. CLARUS only needs to compute m^2 subtractions. As to communications during matrix product, only C_0 needs to send an $m \times m$ matrix, which makes $m^2\gamma$ bits.

With the benchmark Protocol 3, CLARUS needs less set-up communication than with Protocol 1, but more communication during the matrix product computation. The difference between the two protocols regarding storage is substantial: Protocol 1 requires CLARUS to use much less temporary and long-term storage. Another important difference refers to the computing time for the matrix product: in the benchmark protocol, CLARUS needs to perform nm decryptions (to decrypt the entire data set) plus nm^2 products, which is clearly more work than the m^2 subtractions needed under Protocol 1.

In summary, the advantage of Protocol 1 is that it permits different statistical analyses without requiring the use of the original data set $\mathbf{X}$: just using $\mathbf{Y}$ suffices and most of the computational burden falls on the C_0 cloud. In contrast, Protocol 3 requires CLARUS to download and decrypt $\mathbf{X}$ before performing any analysis.

4.5.2 Comparison for the sharing-resistant alternative with side knowledge

Table 4.2 shows all the costs for Protocols 2 and 3. Protocol 2 requires a set-up phase to anonymize the original data set, compute the true error matrix and obtain the t plausible fake error matrices by means of an SDC method.

To compute Equation (4.4) t times, Alice first computes $\mathbf{Y^T Y}$ once, and then, for $i = 1, \ldots, t$, she computes $(\mathbf{E_i})^{\mathbf{T}}\mathbf{E_i}$, $\mathbf{Y^T E_i}$ and $(\mathbf{E_i})^{\mathbf{T}}\mathbf{Y}$. This takes

Table 4.3: **(Computational specifications of CLARUS and CSPs)** CLARUS, the trusted proxy, was run on a local computer. CERSEI is a local server mimicking a for-payment CSP. AWS represents a free-of-charge t2.micro Amazon EC2 instance.

Machine	Operating System	Width(bits)	CPU(GHz)	RAM(GB)	HDD(GB)
CLARUS	Windows 7	64	2.5	8	500
CERSEI	Ubuntu 14.4 LTS	64	3.4	16	500
AWS	Ubuntu Server 16.04 LTS	64	2.4	1	30

$(2t+1)nm^2$ products, and transposing $(\mathbf{Y})^{\mathbf{T}}\mathbf{E_i}$ into $(\mathbf{E_i})^{\mathbf{T}}\mathbf{Y}$ for $i = 1, \ldots, t$ takes tm^2 reads.

When comparing Protocol 2 with Protocol 3, we can see that the latter has smaller set-up costs for CLARUS, but larger costs for the matrix computation phase. Although the set-up costs are higher for Protocol 2, they allow usefully releasing the data protected with anonymization to potential users (such as researchers), which is not possible with Protocol 3 (because protected data are encrypted in the latter protocol).

4.6 Experimental results

This section details the experimental results obtained by implementing the proposed protocols in Java in a multi-cloud scenario. The reported experiments were conducted using the first two attributes of the California housing data set (CADATA, [10]), a usual test data set in the statistical disclosure control literature that contains 9 numerical attributes and $20,640$ records. Let $\mathbf{X}$ be the matrix containing the two selected attributes of CADATA.

First, we ran the tests on Amazon Web Services (AWS), a public CSP that offers 12 months free tier. The tests were performed on a t2.micro Amazon EC2 instance for each cloud. Since the computing power and main storage were substantially capped in this free-of-charge service, and communication with it was slow, we took it as representing the low-end scenario a user can expect from a CSP. Second, we also used a local server (CERSEI) offering more computational power and main storage, as well as faster communication, in order to mimic the service that can be expected from a for-payment CSP. In both cases, we used another local computer to run CLARUS on it, that worked as the proxy located in a trusted domain. Table 4.3 summarizes the specifications of CLARUS, CERSEI and the AWS instance.

The computational cost was measured as the time in seconds that each machine spent to perform the computations specified by the protocols. The communication cost was measured as an approximation of the time each cloud spent at sending data (writing to the channel) or receiving data (reading from the channel). As explained in Section 4.5, we consider in general that the participants send the seeds of random vectors and matrices, rather than the vectors and matrices themselves. The reason is that generating a random vector takes normally less time than sending it. However, the communication time depends on many factors (user internet connection, distance between users, network load, etc.). In our experimental setting, we compared the times for generating and sending vectors of several sizes n; the results are reported in Table 4.4, where it can be seen that sending a random vector takes longer than generating it at the recipient for sizes $n > 10^4$.

4.6.1 Comparison for the sharing-resistant alternatives

In this section, Protocol 1 (sharing-resistant without background knowledge) and Protocol 2 (sharing-resistant with background knowledge) are compared with the benchmark Protocol 3.

Table 4.5 shows the storage, computation and communication costs for the aforementioned protocols. CLARUS needs significantly less storage, computation and communication resources in Protocols 1 and 2 than in the benchmark Protocol 3. In fact, once the set-up phase is completed, the sharing-resistant protocols are extremely convenient for CLARUS, because nearly all the computations are performed by the cloud and CLARUS only needs to do very little work. AWS and CERSEI give results that are consistent with the analytical comparison in Sections 4.5.1 and 4.5.2.

Table 4.4: **(Time to send a random vector vs time to generate it at the recipient)**: n is the number of elements of a random vector $\mathbf{x}$ and the times are given in seconds. Sending is faster only for short vectors.

Time (s) \ n	10^3	10^4	10^5	10^6
to generate $\mathbf{x}$	2.08×10^{-3}	4.6×10^{-3}	8.4×10^{-3}	2.07×10^{-2}
to send $\mathbf{x}$	2.8×10^{-4}	3.5×10^{-3}	1.9×10^{-2}	1.19×10^{-1}

Table 4.5: **(Costs for Protocols 1, 2 and 3 for AWS and CERSEI)**: $\boldsymbol{X}$ is a $20,640 \times 2$ matrix containing the original data set. The components in the vectors and matrices take $\gamma = 8$ bytes. The storage is given in bytes (B) and the computational and communication costs are given in seconds (s).

	Protocol 1			Protocol 3		Protocol 2	
	Alice (C_0)	Bob (C_j)	CLARUS	Alice	CLARUS	Alice	CLARUS
Long-term(B)	3.3×10^5	3.3×10^5	32	3.3×10^5	0	3.6×10^6	8
Temporary(B)	0	0	64	0	6.6×10^5	124	160
AWS	Protocol 1			Protocol 3		Protocol 2	
	Alice (C_0)	Bob (C_j)	CLARUS	Alice	CLARUS	Alice	CLARUS
$\mathbf{X^T X}$ comp.(s)	0.2	0	4.4×10^{-5}	0	0.9	0.05	2.03×10^{-5}
$\mathbf{X^T X}$ comm.(s)	1.2×10^{-4}	0	0.4	2.2	2.6	3.6×10^{-4}	0.4
CERSEI	Protocol 1			Protocol 3		Protocol 2	
	Alice (C_0)	Bob (C_j)	CLARUS	Alice	CLARUS	Alice	CLARUS
$\mathbf{X^T X}$ comp.(s)	0.08	0	4.2×10^{-5}	0	0.7	0.03	1.9×10^{-5}
$\mathbf{X^T X}$ comm.(s)	5.4×10^{-5}	0	0.2	0.3	0.5	8.2×10^{-5}	0.2

4.7 Summary

We have presented two new protocols for outsourcing to untrusted clouds the matrix products of sensitive data. Based on this operation, more complex data analyses can be performed, such as correlations and contingency tables. The goal is to minimize the amount of work that needs to be performed locally by the controller, who wants to use the cloud as much as possible to compute on her outsourced sensitive data. For the sake of flexibility and efficiency, we have considered non-cryptographic methods for data protection, such as data splitting and anonymization, rather than the heavier fully homomorphic encryption (e.g. [34]). A distinguishing feature of our approach is that the outsourced data on which the clouds compute retain some of the utility of the original data, which entails added value with respect to outsourcing encrypted or otherwise gibberish data.

In case clouds can share information but have no side knowledge on the original data set, we have proposed a sharing-resistant protocol based on orthogonal noise matrices that shifts most of the computational burden to the clouds. For the worst case, in which clouds share information and have side knowledge allowing them to recognize the original data set, we have proposed a sharing-resistant protocol relying on noise matrices derived via anonymization. Although the latter protocol is heavier, it still substantially relieves the controller (CLARUS) in

computational terms. We have provided complexity analyses and benchmarking for all proposed protocols, in order to show their computational advantages for the outsourcing controller. Further, we have provided experimental evidence that the new protocols take less effort from CLARUS than the benchmark protocols consisting of downloading and local processing.

In this way, clouds are not only used to store sensitive data, but also to perform computations on these data in a privacy-aware manner. This is especially interesting for large sensitive data sets.

Chapter 5

Multivariate Categorical Analyses in Untrusted Clouds

5.1 Introduction

Statistical analyses involve collecting and investigating (potentially large) data samples. In turn, collecting and investigating data requires storing them and computing on them. Among the usual analyses, measuring the dependence between attributes in multivariate data sets is one of the costliest operations. For instance, measuring the correlation between (just) two categorical attributes in a data set containing one million records may require computing and storing matrices of size one million times one million [55]. If matrix values are as short as 4-byte integers (real numbers would take more), then storing one matrix alone takes nearly 4 terabytes.

Coping with such huge data amounts is often infeasible for data controllers. In this scenario, outsourcing storage and computation to the cloud is an attractive alternative because of the large, cheap and highly scalable resources it offers. Nevertheless, when the data to be outsourced contain sensitive information (e.g., personal data, clinical outcomes, etc.), data controllers may be reluctant to embrace the cloud due to privacy concerns [4]. These concerns are not only related to the fact that the cloud service providers (CSPs) may read,

use or even sell the data outsourced by their customers, but also because CSPs may suffer attacks or data leakages that can compromise data confidentiality.

To mitigate the above concerns, privacy-preserving methods for storing and processing the data outsourced to the cloud should be designed. This is the main goal of the European project CLARUS [17] in which the current work is framed. Following its architecture, we will assume a proxy located in a domain trusted by the data controller (e.g., a server in her company's intranet or a plug-in in her device) that implements security and privacy-enabling features towards the cloud service providers. We will call this trusted proxy CLARUS, see Section 2.2 for more details.

However, performing many statistical analyses, such as data dependence or correlation assessment, requires using the whole data set or at any rate more than a single fragment. With split data, the problem is for the controller to manage as effortlessly as possible the data fragments stored at different untrusted CSPs to conduct such computations while ensuring that data attributes in different fragments cannot be linked by the CSPs [12]. The problem is even more challenging when dealing with nominal categorical data, i.e., data whose attribute values are noun-phrases corresponding to jobs, interests, conditions, etc. These data, that are textual and non-ordinal and on which the standard arithmetic operators cannot be used, account for most of the personal information currently being collected (e.g., in social networks, B2C transactions, etc.) [72]. In particular, accurately measuring the dependence or correlation between nominal attributes requires semantically grounded techniques [59] that are costly, both in computational power and storage.

Contributions

Chapter 3 shows several non-cryptographic proposals for statistical computations (basically correlations) on split data across several honest-but-curious non-sharing clouds (see Section 2.2). All these protocols and methods were designed for numerical data. In this chapter, Protocols 1.2 and 2.1, which are the two best protocols identified in Chapter 3, are adapted to categorical data. In this way, we present efficient protocols to securely compute statistical dependence analyses on outsourced split data for a variety of methods, encompassing frequency-based and semantic-based tests. In all cases, the goal of the protocols is to outsource as much workload as possible to the cloud, while ensuring that confidential data are not leaked to the CSPs. Semantic-based tests, that are the costliest ones and those that would benefit the most from outsourcing the computation to the cloud, are analyzed in greater detail. Specifically, several

semantic measures to quantify distances between nominal values are discussed and evaluated, both theoretically and in practice. We also report on empirical work on Amazon Web Services cloud instances. Performance figures show that our protocols are able to outsource most of the workload to the CSPs, thereby reconciling data privacy with the cost saving benefits of the cloud.

The contributions in this chapter have been published in [55]. An extended version has been submitted to a journal, feedback on it has been received, and a revised version has been submitted.

The rest of this chapter is organized as follows. In Section 5.2 we propose protocols to securely outsource the computation of frequency-based tests (χ^2-test, ANOVA or Cramer's V), and semantic-based tests (semantic-distance covariance). Section 5.3 reports the experimental results obtained when implementing our protocols for the costliest analysis (the semantic-based test) and compares the workload savings against a local computation. Section 5.4 contains the conclusions.

5.2 Privacy-preserving multivariate analyses on the cloud

In this section, we show how the multivariate analyses introduced in Section 2.4 can be performed on split data outsourced to honest-but-curious non-sharing CSPs by relying on the secure scalar product protocols. Protocols 1.2 and 2.1 represent the most efficient protocols among the ones compared in Section 3.4. If we focus on the cost/security trade-off, Protocol 1.2 is probably the best choice, whereas Procotol 2.1 guarantees the highest security.

For each analysis, the CLARUS proxy decomposes and orchestrates calculations and aggregates partial results. To avoid overloading the local system in which the CLARUS proxy runs, the protocols are designed to keep the workload of the CLARUS proxy as low as possible by outsourcing as much storage and computation as possible to the CSPs in a privacy-preserving way.

5.2.1 Frequency-based analyses

To calculate the χ^2-test, ANOVA or Cramer's V, CLARUS orchestrates the calculation of the contingency table of the split attributes stored in separated CSPs, which is the input of the aforementioned tests, see Section 2.4 for more details.

To obtain the contingency table from data vertically split among several clouds, one just needs to compute the table cells. Let $(a_1, \ldots, a_n)^T$ and $(b_1, \ldots, b_n)^T$ be the vectors of values from the attributes $\mathbf{a}$ and $\mathbf{b}$, owned by the CSPs Alice and Bob, respectively. A cell C_{ij} (for every $i = 1, \ldots, h$ and $j = 1, \ldots, k$) is computed by counting the number of records in the original data set containing both the categories $c_i(\mathbf{a})$ and $c_j(\mathbf{b})$. Alice creates a new vector $\mathbf{x} = (x_1, \ldots, x_n)^T$ such that

$$x_l = \begin{cases} 1 & \text{if } a_l = c_i(\mathbf{a}) \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } l = 1, \ldots, n. \tag{5.1}$$

Bob creates $\mathbf{y} = (y_1, \ldots, y_n)^T$ such that

$$y_l = \begin{cases} 1 & \text{if } b_l = c_j(\mathbf{b}) \\ 0 & \text{otherwise} \end{cases} \qquad \text{for } l = 1, \ldots, n. \tag{5.2}$$

The scalar product $\mathbf{x}^T\mathbf{y}$ (computed by means of the Protocols 1.2 and 2.1 detailed in Section 3.3) gives the number C_{ij} of records in the original data set containing both the categories $c_i(\mathbf{a})$ and $c_j(\mathbf{b})$.

Specifically, Alice and Bob can use Protocols 1.2 and 2.1 to securely compute C_{ij} by just adding two preliminary steps to the scalar product computation part: one step by Alice to generate $\mathbf{x}$ from $(a_1, \ldots, a_n)^T$ using Equation (5.1), and another step by Bob to generate $\mathbf{y}$ from $(b_1, \ldots, b_n)^T$ using Equation (5.2).

Security. The only modification with respect to Protocols 1.2 and 2.1 is that Alice and Bob compute $\mathbf{x}$ and $\mathbf{y}$, respectively. These computations are done by the clouds in isolation, i.e., without exchanging information; hence, the security of the protocol is preserved.

Cost. Once the contingency table is obtained, frequency-based tests, which have a low computational cost, can be run locally by CLARUS. In fact, for the χ^2-test, ANOVA and Cramer's V the most demanding computation is a linear regression; therefore, given a $h \times k$ contingency table with $h < k$, the linear regression has complexity $O(h^2k + h^3)$ (notice that h and k are much smaller than the number n of records of the original data set). For CSPs, Alice and Bob, the computation of one cell has $O(n)$ cost in both Protocol 1.2 and Protocol 2.1. In particular, in Protocol 1.2 Alice and Bob have to perform, respectively, n products and n reads as the most demanding computations; Charlie (the third cloud needed in the protocol) generates two random n-vectors and CLARUS

just performs two sums. In Protocol 2.1, Alice performs n encryptions, n reads and n random number generations. Bob performs n reads and n products as demanding computations. CLARUS computes one sum. Since the contingency table has $h \times k$ cells, Alice's and Bob's calculation has $O(n \times h \times k)$ cost. CLARUS just needs to compute 2 sums in Protocol 1.2 or 1 sum in Protocol 2.1 for each table cell, that is, constant cost. Therefore, the CLARUS computation has complexity $O(h \times k)$.

Another reason to locally conduct the calculation of the frequency-based tests is that sharing the contingency table with a CSP can lead to privacy issues, because the table may contain cells with values one or zero that may allow re-identifying some subjects. For instance, if a cell representing the number of Asian people with HIV that answered a specific survey has value equal to one, just knowing that only one participant of the survey was Asian discloses that he is sick. Moreover, this information can be enough to recognize a subject if, for example, the survey was carried out in an area with only few Asian families.

Observe that, if one CSP stores in its own data fragment all the attributes required for the contingency table computation, all the calculations are done by that CSP in isolation, and CLARUS just receives the result of the required frequency-based test.

5.2.2 Semantic-based analyses

As introduced in Section 2.4, the calculation of the distance covariance requires measuring the pairwise semantic distance between the nominal values of each attribute. The pairwise distances as well as the double-centered matrices are computed among the values of one attribute at once and, therefore, the CSP owning the attribute performs the calculation in isolation. Each CSP can also compute the distance variance of its attribute in isolation. Then the CSPs use Protocols 1.2 or 2.1 to securely compute the distance covariances in view of completing the distance covariance matrix $\hat{\mathbf{\Sigma}}$.

Formally, let $\mathbf{x^1} = (x_1^1, \ldots, x_n^1)^T$ and $\mathbf{x^2} = (x_1^2, \ldots, x_n^2)^T$ be vectors of values of two nominal attributes owned by CSPs Alice and Bob, respectively. Alice computes $\mathbf{X^1}$ and $\hat{\mathbf{X}}^\mathbf{1}$ and Bob computes $\mathbf{X^2}$ and $\hat{\mathbf{X}}^\mathbf{2}$. In this case, the distance covariance matrix $\hat{\mathbf{\Sigma}}$ of $\mathbf{X} = (\mathbf{x^1}, \mathbf{x^2})$ is given by

$$\hat{\mathbf{\Sigma}} = \left(\begin{array}{c|c} d\mathcal{V}_n(\mathbf{x}^1) & d\mathcal{V}_n(\mathbf{x}^1, \mathbf{x}^2) \\ \hline d\mathcal{V}_n(\mathbf{x}^2, \mathbf{x}^1) & d\mathcal{V}_n(\mathbf{x}^2) \end{array} \right).$$

Note that $d\mathcal{V}_n(\mathbf{x}^i, \mathbf{x}^j)$ is the square root of $d\mathcal{V}_n^2(\mathbf{x}^i, \mathbf{x}^j)$, for $i, j = 1, \ldots, m$,

and that X^j, X^j_{kl}, $d\mathcal{V}_n(\mathbf{x^j})$, for $j = 1, \ldots, m$, are separately computed by the CSP storing the respective attribute. The most challenging task is, therefore, calculating the squared sample distance covariance, i.e., Equation (2.4), which requires performing n secure scalar products of vector pairs, where the two vectors in each pair are respectively held by two different CSPs.

In fact, by calling $\mathbf{X^1_k} = (X^1_{k1}, \ldots, X^1_{kn})^T$ and $\mathbf{X^2_k} = (X^2_{k1}, \ldots, X^2_{kn})^T$ for $k = 1, \ldots, n$, we can rewrite Equation (2.4) as

$$d\mathcal{V}^2_n(\mathbf{x^1}, \mathbf{x^2}) = \frac{1}{n^2} \sum_{k=1}^{n} \left(\sum_{l=1}^{n} X^1_{kl} X^2_{kl} \right) = \frac{1}{n^2} \sum_{k=1}^{n} (\mathbf{X^1_k})^T \mathbf{X^2_k}, \tag{5.3}$$

where the n scalar products are $(\mathbf{X^1_k})^T \mathbf{X^2_k}$ for $k = 1, \ldots, n$.

Therefore, once the double-centered matrices are obtained, the distance covariance matrix computation with data split among different CSPs can be decomposed into several secure scalar products to be conducted between pairs of clouds. Protocols 1.2 and 2.1 are perfectly suited to compute $(\mathbf{X^1_k})^T \mathbf{X^2_k}$ for $k = 1, \ldots, n$ (see Section 3.3). The only adaptation needed is to add two preliminary steps: one step for Alice to compute $\mathbf{x} = \mathbf{X^1_k}$ from $\mathbf{x^1}$, and another step for Bob to compute $\mathbf{y} = \mathbf{X^2_k}$ from $\mathbf{x^2}$.

Security. The two preliminary steps added before the secure scalar product are separately performed by Alice and Bob, so there is no additional exchange of information between the clouds. Hence, the security of Protocols 1.2 and 2.1 is preserved.

Cost. Calculating the distance covariance matrix between two nominal attributes has a quadratic cost, both in time and storage. Moreover, generating the semantic-distance matrices (Eq. (2.1)) requires using the semantic measures. Let h_1 be the number of categories of $\mathbf{x^1}$ and h_2 be the number of categories of $\mathbf{x^2}$, where $h_1,\ h_2 \leq n$; then $h_1^2/2$ and $h_2^2/2$ semantic distances are computed for each attribute by Alice and Bob, respectively. Recalling the costs discussed in Section 2.5 for each type of semantic measure, we have that the cost of generating the semantic-distance matrix is $O(h_j^2 \times D)$ for the edge-counting measure, $O(h_j^2 \times S)$ for the feature-based measure and $O(h_j^2 \times (C + D))$ for the information content-based measure, for $j = 1, 2$ and where D is the depth of the taxonomy, S is the maximum number of subsumers of any concept and C is the total number of concepts in the ontology (which can be in the order of thousands or hundreds of thousands in large ontologies). On the other hand,

the double-centered matrix (Eq. (2.2)), which is also computed by each CSP independently, has $O(n^2)$ computational cost.

Finally, the distance covariance matrix computation is decomposed into several scalar products, where the total number of scalar products performed by the CSPs is $3n$. Each scalar product has $O(n)$ computational cost for both Protocols 1.2 and 2.1. In particular, in Protocol 1.2 Alice and Bob have to perform, respectively, n products as the most demanding computations, Charlie (a third cloud needed in the protocol) generates two random n-vectors and CLARUS just performs two sums. In Protocol 2.1, Alice performs n encryptions and n random number generations. Bob performs n products as the most demanding computations. CLARUS computes one sum. Consequently, the CSPs' computation has $O(n^2)$ cost and CLARUS's computation has $O(1)$ cost. The storage needs at the CSPs are also quadratic due to the need to create several $n \times n$ matrices, i.e., 1 semantic-distance matrix and 1 double-centered matrix per attribute.

One can notice that the calculation of the semantic-distance covariance is significantly costlier than the frequency-based method (both in time and storage); yet, the protocol we propose is able to outsource the cost to the CSPs, thus keeping the CLARUS workload low even with large data sets.

5.3 Experimental results

This section reports the results of the implementation of our protocols in a real setting. As a use case, we employed the most computationally demanding analysis: the semantic-distance covariance. As evaluation metrics, we report the workload of each entity (CLARUS and the CSPs) and quantify the percentage of workload that our protocols were able to securely outsource to the CSPs w.r.t. a local implementation of the analysis.

The tests were run in a free-tier CSP provided by Amazon Web Services (AWS). It is important to note that the computing power and storage of such a free-of-charge service are substantially limited and, therefore, significant improvements can be expected when moving to payment services. On the client side, a local computer was configured to act as the CLARUS proxy, which is in charge of orchestrating the storage and calculations on the outsourced data. The specifications of AWS and CLARUS are summarized in Table 5.1.

The experiments were conducted on a sample of $1,000$ records with two nominal attributes extracted from a patient discharge database provided by the

Table 5.1: Specifications of CLARUS (the trusted proxy running on a local computer) and the AWS CSPs (free-of-charge t2.micro Amazon EC2 instances)

Machine	Operating System	Width(bits)	CPU(GHz)	RAM(GB)	HDD(GB)	Instances
CLARUS	Windows 7	64	2.5	8	500	1
AWS CSP (t2.micro instance)	Ubuntu Server 16.04 LTS	64	2.4	1	30	3

Table 5.2: AWS instance types. The t2.micro free-of-charge instance was used in our experiments.

AWS instance type	Name	CPU Cores	RAM(GB)	Clock Speed(GHz)
General purpose	t2.micro	1	1	Up to 3.3
General purpose	t2.2xlarge	8	32	Up to 3.0
General purpose	m4.16xlarge	64	256	2.3
Compute optimized	c4.8xlarge	36	60	2.9
Accelerated computing	f1.16xlarge	64	976	2.3
Memory optimized	r4.16xlarge	64	488	2.3
Memory optimized	x1.32xlarge	128	1,952	2.3

California Office of Statewide Health Planning and Development [11]. The two nominal attributes represent the diagnosis ($\mathbf{x}$) and medical procedure ($\mathbf{y}$) of each patient. Notice that the size of the sample is deliberately small because of the limited resources of the CSPs instances we used. To cope with larger data sets, one just needs to hire more powerful CSP instances, e.g., see those offered by AWS [3] in Table 5.2. Given the computational cost figures we discussed in the former section, scaling the obtained results for larger data sets and more attributes is straightforward.

SNOMED-CT was used as the ontology for the semantic distance calculation in the semantic-based test. SNOMED-CT models 321,901 clinical concepts and constitutes the largest and most detailed medical knowledge base [66].

In all the experiments, two AWS CSPs (Alice and Bob) separately stored the two attributes (diagnosis ($\mathbf{x}$) and procedures ($\mathbf{y}$), respectively), whereas a third CSP (Charlie) was used as commodity server. For each analysis, we reported the storage requirements and workload of each CSP and CLARUS for the protocols we propose, and compared them against a local implementation in which CLARUS should store the whole data and perform all the computations.

As detailed in Section 5.2.2, first each CSP computes in isolation the semantic-distance matrix (Eq. (2.1)) and the double-centered matrix (Eq. (2.2)) of the attribute it stores; then, Alice and Bob jointly work on the calculation of the distance covariance. CLARUS performs a small part in this latter calculation,

Table 5.3: Long-term and temporary storage for the semantic-distance covariance calculation with two attributes and 1,000 records.

Storage requirements (MB)									
LOCAL		CLOUD							
Long-term	Temporary	Long-term				Temporary			
CLARUS	CLARUS	Alice	Bob	CLARUS	Charlie	Alice	Bob	CLARUS	Charlie
40	242.5	16	16	8	0	242.5	242.5	0	0

and its workload depends on the secure scalar protocol in use: Protocol 1.2 or Protocol 2.1. In particular, the total number of scalar products performed by the CSPs is $(m(m-1)/2) * n + m * n$, out of which $(m(m-1)/2) * n$ are secure scalar products, being n the number of records and m the number of attributes.

In the local implementation, CLARUS plays the part of the data controller and is required to perform all the computation by itself: semantic-distance matrices, double-centered matrices and the distance covariance. However, since CLARUS owns the whole data and runs in a trusted environment, no secure scalar products are needed.

Table 5.3 shows the storage requirements of the calculations for the cloud-based and local scenarios. The storage is broken down into long-term and temporary: the former corresponds to the storage of the split data, whereas the latter is the storage required to conduct the calculation at some point, which can be discarded once the calculation is finished. In terms of temporary storage, the CSPs (or CLARUS in the local solution) need to load into RAM the SNOMED-CT ontology, which requires 242.5 MB. The semantic-distance and the double-centered matrices of the attributes are stored in the long-term storage for them to be re-used in further calculations. The local solution, which requires storing the matrices of all the attributes, can be considerably heavy for CLARUS when the number of records and/or attributes is large. In contrast, in the cloud-based solution only the semantic-distance covariances are stored by CLARUS. The use of secure scalar products imperceptibly increases the required storage (for 1,000 records, the storage increases by around 0.032 MB for Protocol 1.2 and by 0.065 MB for Protocol 2.1).

Table 5.4 shows the computation and communication runtimes of the distance covariance calculation with the three ontology-based semantic measures. Notice that in the local scenario there is no exchange of information between separate entities and, therefore, there is no communication cost. In the cloud-based solution, Protocols 1.2 and 2.1 were used for the computation of the secure

Table 5.4: Computation and communication runtimes for the distance-covariance calculation with the three semantic-distance measures and the two secure scalar product protocols. The computation runtime (Comp.) represents the time each entity spent following the protocol. The communication runtime (Com.) is an approximation of the time the CSPs and CLARUS spent sending and receiving the data. The times are given in minutes (m.).

LOCAL	CLOUD						
Comp. (m.)	Comp. (m.)						Com. (m.)
Edge-counting measure (Eq. (2.6))							
CLARUS		Alice	Bob	Charlie	CLARUS	Total comp.	Total com.
21.3	Prot.1.2	9.9	2.4	3.5×10^{-4}	2.8×10^{-5}	12.3	16.5
	Prot.2.1	34.2	2.4	–	8.7×10^{-5}	36.7	6.3
Feature-based measure (Eq. (2.7))							
CLARUS		Alice	Bob	Charlie	CLARUS	Total comp.	Total com.
22.8	Prot.1.2	9.6	2.5	3.5×10^{-4}	2.7×10^{-5}	12.1	16.5
	Prot.2.1	33.8	2.4	–	9.1×10^{-5}	36.2	6.6
Information content-based measure (Eq. (2.8))							
CLARUS		Alice	Bob	Charlie	CLARUS	Total comp.	Total com.
183.8	Prot.1.2	836.5	49.2	3.9×10^{-4}	2.6×10^{-5}	885.7	16.5
	Prot.2.1	863.2	49.3	–	9.7×10^{-5}	912.5	6.6

scalar products. Observe that Protocol 2.1 results in higher costs in terms of computation due to the use of cryptographic primitives. Moreover, the runtime of Alice is significantly larger than Bob's, although the attributes have the same length. The reason is that the SNOMED-CT taxonomy for the attribute stored by Alice (diagnosis, i.e., clinical finding) is much larger than that of Bob's attribute (procedure), as shown in Table 5.5. Furthermore, within the 1,000-record data set, Alice's attribute has 434 categories, whereas Bob's attribute has only 342; hence, Alice needs to perform a greater number of semantic-distance assessments.

The reported runtime figures are consistent with the cost of the semantic measures we detailed in Section 2.5: the edge-counting and the feature-based measures have similar costs, because both analyze the set of ancestors of the concepts to be compared, the measure based on information content is significantly costlier (around 8 times slower in the local solution) due to the need to

iterate through all the hyponyms of each concept. In fact, the calculation of the semantic-distance matrix takes around 13 hours for Alice's attribute (whose domain is significantly larger than Bob's). Even though the runtime of the cloud-based scenario is around 5 times greater that the local one, we should consider the very limited resources of the free t2.micro instance we use. With more powerful instances, runtimes will be decreased to reasonable figures, e.g., a general-purpose t2.2xlarge instance should be around 8 times faster than the free instance (see Table 5.2), which would make the cloud-based calculation faster than the local one.

Since absolute runtime figures depend on the amount of resources of the CSPs, in Table 5.6 we report a more general metric stating the percentage of runtime saved by CLARUS (which runs on local premises) when outsourcing local calculations to the cloud. The runtime saved by CLARUS was computed with the formula

$$100 * \frac{\text{CLARUS}_l - \text{CLARUS}_c}{\text{CLARUS}_l}, \tag{5.4}$$

where CLARUS_l represents the computation runtime of CLARUS in the local scenario and CLARUS_c represents the computation runtime of CLARUS in the cloud-based scenario.

Since the calculation of the semantic-distance matrices is, by far, the costliest operation (especially for the measure based on information content), outsourcing this calculation results in very large savings (i.e., very low workload and also very low storage requirements) for CLARUS.

5.4 Summary

Data splitting is an alternative to encryption that is more flexible and efficient for securing sensitive data outsourced to the cloud. With data splitting, CSPs do not only store data, but they can efficiently conduct computations on the data

Table 5.5: Number of concepts in some taxonomies of SNOMED-CT

Taxonomy	Number of concepts
Body structure	31,206
Clinical findings	104,737
Pharmaceutical/biologic product	17,425
Procedure	55,880
Substance	25,911

Table 5.6: Percentage of computation runtime saved by CLARUS when moving from the local scenario to the cloud-based scenario for the different semantic measures and scalar product protocols

Computation runtime saved by CLARUS (%)							
Edge-counting			Feature-based			Information content-based	
Prot. 1.2	99.99987		Prot. 1.2	99.99988		Prot. 1.2	99.99999
Prot. 2.1	99.99959		Prot. 2.1	99.99960		Prot. 2.1	99.99995

they store in a privacy-preserving manner. Multivariate analyses are, however, challenging; the reason is not just their potentially large computational cost, but also the difficulty of performing the calculations involving data fragments stored in different clouds.

In this paper, we have presented protocols to securely outsource the computation of several multivariate statistical analyses on nominal data split among a number of honest-but-curious clouds. Our protocols are designed to outsource as much workload as possible to the CSPs, which is especially interesting for computationally demanding calculations that may not be affordable locally. In this way, we retain the cost-saving benefits of the cloud while ensuring that the outsourced data do not incur privacy risks.

Empirical tests conducted on AWS free-tier cloud instances confirm our theoretical assumptions. Experimental results clearly show that outsourcing the calculations to the cloud considerably decreases the workload of the data controller, who can save more than 99.999 per cent of the runtime for the most demanding test we considered.

Chapter 6

A Methodology to Compare Anonymization Methods Regarding Their Risk-Utility Trade-Off

6.1 Introduction

With the expansion of information technology, the importance of data analysis (e.g. to support decision making processes) has significantly increased. Although data collection has become easier and more affordable than ever before, releasing data for secondary use (that is, for a purpose other than the one that triggered the data collection) remains very important: in most cases, researchers cannot afford collecting themselves the data they need. However, when the data released for secondary use refer to individuals, households or companies, the privacy of the data subjects must be taken into account.

Statistical disclosure control (SDC) methods aim at releasing data that preserve their statistical validity while protecting the privacy of each data subject. Among the possible types of data releases, this work focuses on microdata (that is, on the release of data about individual subjects).

While there is a great diversity of SDC methods for microdata protection, all of them imply some level of data masking. The greater the amount of

masking, the greater are both privacy protection and information loss. Different SDC methods tackle the trade-off between privacy and utility in different ways. For example, in global recoding the level of information loss is set beforehand (the amount of coarsening of the categories of each attribute), whereas the disclosure risk is evaluated afterwards on the protected data set. In contrast, in k-anonymity [61] the risk of disclosure (the risk of record re-identification, in particular) is set beforehand, whereas the actual information loss results from the masking needed to attain the desired level of disclosure risk.

Although some general assertions about specific SDC methods/models can be made, comparing the latter regarding the privacy-utility trade-off is not straightforward. Let us illustrate this point with two well-known privacy models: differential privacy [31] and k-anonymity [61]. In terms of privacy protection, ϵ-differential privacy is regarded as stronger than k-anonymity. On the contrary, k-anonymity is regarded as more utility-preserving than ϵ-differential privacy. The practical value of these general statements is dubious. After all, by increasing ϵ we reduce the protection of differential privacy, and by increasing k we reduce the utility of k-anonymous data. An accurate comparison between SDC methods has to take into consideration both aspects of the privacy-utility trade-off.

Related works and contributions

Many risk and utility measures have been proposed in the literature, but some of them are designed for use with specific SDC methods. For example, the probability of record re-identification is the natural risk measure in k-anonymity, but it may not be appropriate in SDC methods that are not predicated on protecting privacy by hiding each data subject within a crowd. In this work, we propose a framework based on general empirical measures of utility and risk to compare the risk-utility trade-off of several SDC methods.

Previous comparative studies (e.g. [24]) usually start by selecting some parameter values for a set of SDC methods and evaluate the disclosure risk and the information loss yielded by the methods for those parameterizations. In contrast, here we start by setting a certain risk level (or a certain utility level) and then we find which parameter values are needed to attain that risk (resp. that utility) under different SDC methods; finally, once we have achieved an equivalent risk level (resp. utility level) across methods, we evaluate the utility (resp. the risk) provided by each method, in order to rank methods according to their utility preservation (resp. disclosure protection), given a certain level of risk (resp. utility) and a certain original data set. Furthermore, we present

experimental work that illustrates the application of the proposed methodology.

The contributions in this chapter have been published in [22].

The rest of the paper is organized as follows. In Section 6.2, we describe the proposed framework for comparing methods regarding their risk-utility trade-off. In Section 6.3, we propose an empirical measure of disclosure risk that is based on record linkage. Experimental results are reported in Section 6.4. Conclusions are gathered in Section 6.5.

6.2 A methodology for comparing the risk-utility trade-off in SDC

In this section we describe a methodology for comparing SDC methods. Looking only at either the disclosure risk or the utility of an SDC method would be a flawed comparison. We need to analyze the privacy-utility trade-off, as explained in the introduction. Even if this principle may seem evident, very often it is not followed.

To make the proposed methodology as general as possible, we will employ empirical measures of risk and utility. That is, we will choose risk and utility measures that depend on the original and the anonymized data sets, rather than being prior conditions. To select specific measures, we need to define the aspects of risk and utility that we consider relevant for our comparison. In turn, the choice of measures will shape the outcome of the evaluation.

Let us illustrate the difference between empirical measures and prior conditions by taking differential privacy as an example. As a privacy model, differential privacy states some privacy guarantees but does not tell how they ought to be attained. Let us assume that A_1 and A_2 are ϵ-differentially private algorithms that output a data set. Let us also assume that A_2 is a refined version of A_1 that manages to attain ϵ-differential privacy while adding less noise than A_1. If we use the level ϵ of differential privacy as our risk measure, both A_1 and A_2 are equally good (they are both ϵ-differentially private). However, the fact that A_1 adds more noise to the original records may indicate that the data set output by A_1 entails less disclosure risk than the data set generated by A_2, even if differential privacy is unable to capture the difference. Alternative measures of disclosure risk (e.g. risk measures based on record linkage) should be able to capture the difference in risk between A_1 and A_2. In this work, we do not deny the value of any measure of disclosure risk, but, due to their broader applicability, we will employ empirical risk measures based on record linkage.

Let us assume that we are given functions

$$\mathcal{U} : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$$

$$\mathcal{R} : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$$

such that, for any given original data set $\mathbf{X}$ and anonymized data set $\mathbf{Y}$,

- $\mathcal{U}(\mathbf{X}, \mathbf{Y})$ measures the utility of $\mathbf{Y}$ as a replacement for $\mathbf{X}$.
- $\mathcal{R}(\mathbf{X}, \mathbf{Y})$ measures the disclosure risk of $\mathbf{Y}$ as a replacement for $\mathbf{X}$.

We have described some utility measures in Section 2.8. In Section 6.3 we will describe several risk measures based on record linkage.

SDC methods usually accept some parameters that can be adjusted to select the desired level of disclosure risk/utility. Let $M_\alpha(\mathbf{X})$ be the anonymized data output by SDC method M with parameter α when applied to data set $\mathbf{X}$.

Given an original data set $\mathbf{X}$ and two anonymization algorithms M^1 and M^2, we say that M^1 is more utility-preserving than M^2 at risk level r if

$$\mathcal{U}(\mathbf{X}, M^1_\alpha(\mathbf{X})) \geq \mathcal{U}(\mathbf{X}, M^2_\beta(\mathbf{X})),$$

for α and β such that $\mathcal{R}(\mathbf{X}, M^1_\alpha(\mathbf{X})) = \mathcal{R}(\mathbf{X}, M^2_\beta(\mathbf{X})) = r$.

In a similar fashion, we can compare the risk associated to a given level of data utility. We say that M^1 is less disclosive than M^2 at utility level u if

$$\mathcal{R}(\mathbf{X}, M^1_\alpha(\mathbf{X})) \leq \mathcal{R}(\mathbf{X}, M^2_\beta(\mathbf{X})),$$

for α and β such that $\mathcal{U}(\mathbf{X}, M^1_\alpha(\mathbf{X})) = \mathcal{U}(\mathbf{X}, M^2_\beta(\mathbf{X})) = u$.

The results of the previous utility (resp. risk) comparison depend not only on the SDC method, but also on the original data set, the risk and the utility measures selected, and the target level of risk (resp. utility). Actually, this comparison methodology is designed for use by a data controller who must decide which among several SDC methods is best suited to anonymize a given data set with a given target level of disclosure risk or utility. In other words, the aim is not to make general statements about the relative goodness of several SDC methods. Although such statements may make sense in some cases, our results can only be taken as empirical clues of such underlying truths.

6.3 Empirical measures of disclosure risk

To compare the risk-utility trade-off between SDC methods, we need adequate measures of disclosure risk. For the methodology described in Section 6.2 to be broadly applicable, the risk measure should be as general as possible (rather than based on specific characteristics of an SDC method).

We propose a risk measure based on record linkage [76], which is a technique that seeks to match original records that correspond to the same individual. Among its several uses, record linkage has a direct application in disclosure risk assessment [70]. Such an application bears some resemblance to the way an intruder having access to the anonymized data and to some side knowledge would proceed. Let $\mathbf{E}$ be a data set that represents the non-anonymous side information available to the intruder. By linking records in $\mathbf{E}$ to records in $\mathbf{Y}$, the intruder associates identities to the records in $\mathbf{Y}$.

The number (or the proportion) of correct re-identifications is a common record linkage-based measure of disclosure risk. However, this measure has some limitations that we next discuss. It is certainly appropriate when SDC is achieved by masking the quasi-identifier attributes, whereas the sensitive attributes are left unmodified (or are only slightly modified). However, if the sensitive attributes have been significantly altered, a correct linkage may not be equivalent to disclosure. Furthermore, if we use SDC methods that are not based on masking the original records, we may not even be able to tell what a correct linkage is. Generating a synthetic data set by repeatedly sampling from a statistical model adjusted on the original data is an example of an SDC method not based on masking; and indeed, it is not possible to say what is the correct mapping between the original records and the synthetic records.

In the spirit of [20], rather than measuring the disclosure risk as the proportion of correct re-identifications, we will measure the risk of disclosure associated to a record in the original data set $\mathbf{X}$ by means of a distance to its linked record in $\mathbf{Y}$. Such an approach has two important advantages with respect to counting the number of correct re-identifications:

- It is more broadly applicable. The linkage between records in $\mathbf{X}$ and $\mathbf{Y}$ can be performed independently of the SDC methodology used, even when the correct mapping between original and anonymized records cannot be established.

- The distance between a record in $\mathbf{X}$ and its linked record in $\mathbf{Y}$ provides more detailed information about the risk associated to a record in $\mathbf{X}$ than a mere binary outcome (right/wrong linkage):

- On the one hand, the binary nature of correct linkages could lead to understating the risk of disclosure when, in spite of failing to find the correct linkage, the intruder links to a record that is similar to the correct one.
- On the other hand, if all the attributes in $\mathbf{Y}$ have been thoroughly altered by the SDC method, a correct linkage may not disclose any useful information to the intruder; in this case, the proportion of correct linkages would overstate the risk of disclosure.

Any record $\mathbf{x}$ in the original data set $\mathbf{X}$ is linked to the record $\mathbf{y_x} \in \mathbf{Y}$ at the smallest distance, that is, such that

$$d(\mathbf{x}, \mathbf{y_x}) = d(\mathbf{x}, \mathbf{Y}) = \min_{\mathbf{y} \in \mathbf{Y}} d(\mathbf{x}, \mathbf{y}).$$

The distance $d(\mathbf{x}, \mathbf{Y})$ is an indicator of the disclosure risk associated to $\mathbf{x}$. If the distance is small, there is a record in $\mathbf{Y}$ that is quite similar to $\mathbf{x}$ and the risk of disclosure is high.

The choice of the distance $d(\mathbf{x}, \mathbf{Y})$ is an important step in determining the disclosure risk. Along the lines of the permutation paradigm (see Section 2.7), our proposal is based on ranks, but it differs from [20] in the way attributes are aggregated. Let $\mathbf{x} = (x^1, \ldots, x^m)$ be a record from an original data set $\mathbf{X}$ with attributes $X^1, \ldots, X^m$ and $\mathbf{y} = (y^1, \ldots, y^m)$ be a record from an anonymized data set $\mathbf{Y}$ with attributes $Y^1, \ldots, Y^m$. Take the distance between $\mathbf{x}$ and $\mathbf{y}$ to be the Euclidean distance between ranks, that is,

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{m} [rank_{X^i}(x^i) - rank_{Y^i}(y^i)]^2},$$

where the subscript of the rank function denotes the attribute within which the rank of the value in the argument is computed.

The overall risk of disclosure is an aggregation of the distances $d(\mathbf{x}, \mathbf{Y})$ for all $\mathbf{x} \in \mathbf{X}$. Many different aggregations are possible. In this work we focus on the average risk of disclosure by computing the mean of the record distances.

$$\mathcal{R}(\mathbf{X}, \mathbf{Y}) = \frac{1}{n} \log \sum_{\mathbf{x} \in \mathbf{X}} d(\mathbf{x}, \mathbf{Y}). \tag{6.1}$$

The smaller $\mathcal{R}(X, Y)$, the greater the risk of disclosure. The logarithm accounts for the fact that in disclosure risk the focus is on small distances. Without the

logarithm, a large distance for a single record $\mathbf{x} \in \mathbf{X}$ could reduce in a significant manner the perception of risk for the overall data set; the logarithm reduces the influence of large distances.

6.4 Experimental results

In this section we apply the methodology described in Section 6.2 to analyze the relative goodness of several anonymizations. Experiments are conducted by taking as original data the "Census" and "EIA" data sets [9], which are usual test sets in the SDC literature. The "Census" contains 13 numerical attributes and 1080 records, and "EIA" contains 11 numerical attributes and 4092 records.

The anonymized data sets have been generated by applying the following methods:

- *Correlated noise addition.* Multivariate normally distributed noise is added to the records in the collected data set, that is

$$\mathbf{Y} = \mathbf{X} + N(\mathbf{0}, \gamma\mathbf{\Sigma}),$$

  where $\mathbf{\Sigma}$ is the covariance matrix of $\mathbf{X}$ and γ is an input parameter. Note that the covariance matrix of $\mathbf{Y}$ is proportional to the covariance matrix of $\mathbf{X}$.

- *Multiplicative noise.* We have used Höhne's variant [38]. In a first step, each attribute value $x_j^i \in \mathbf{X}$ is multiplied by $1 \pm N(0, s)$, where s is an input parameter. Then, a transformation is applied to preserve the first and second-order moments.

- *Multivariate microaggregation.* We have used the MDAV heuristic [25]. In microaggregation, we partition the records of $\mathbf{X}$ in groups of k or more records, where records in a group are as similar as possible, and we replace each record by the corresponding centroid.

- *Rank swapping.* Independently for each attribute, this method swaps the attribute's values within a restricted range: the ranks of two swapped values cannot differ by more than $p\%$ of the total number of records, where p is an input parameter.

More details about these methods can be found in Section 2.6.

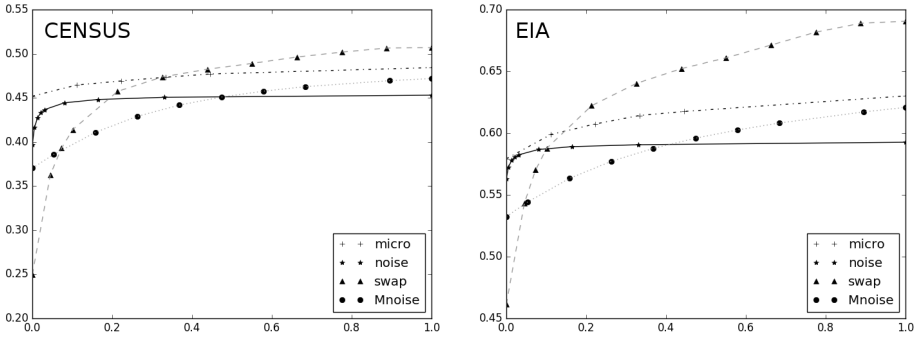

Figure 6.1: Disclosure risk computed according to Eq. (6.1) for the anonymization methods under test and several input parameters. The x-axis shows the input parameter of the anonymization method (k, γ, p and s, respectively), so its scale should be disregarded. The y-axis shows the disclosure risk value. Left, "CENSUS" data set. Right, "EIA" data set.

6.4.1 Disclosure risk assessment

Recall that the comparison of anonymized data sets in Section 6.2 was performed on data sets that had either the same level of risk or the same level of utility. In this experimental work, we aim at determining which among the previous anonymization approaches gives better utility at a given level of disclosure risk. Thus, the first step is to find appropriate parameters for the previous anonymization algorithms that result in a given level of disclosure risk.

Figure 6.1 shows the disclosure risk computed according to Equation (6.1) for the anonymization methods under test:

1. The curve labeled "micro" shows the risk of multivariate microaggregation for values of $k \in \{5, 10, 15, 20, 25, 50\}$.

2. The curve labeled "noise" shows the risk of correlated noise addition when $\gamma \in \{0.01, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 1, 3\}$.

3. The curve labeled "swap" shows the risk of rank swapping when $p \in \{0.01, 0.05, 0.075, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

4. The curve labeled "Mnoise" shows the risk of multiplicative noise when $s \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.9, 1\}$.

For the "Census" data set, a possible match between methods occurs at $\mathcal{R}(X, Y) = 0.45$ and is given by:

1. multivariate microaggregation with $k = 5$,
2. correlated noise addition with $\gamma = 1$,
3. rank swapping with $p = 0.2$, and
4. multiplicative noise with $s = 0.5$.

The microaggregation cluster size $k = 5$ may seem small compared to the parameter values that we get for the other methods. However, such a difference in magnitude can be explained by the fact that multivariate microaggregation is known to yield poorly homogeneous clusters when the number of dimensions is large, even if the cluster size k is small.

For the "EIA" data set, a possible match between methods occurs at $\mathcal{R}(X, Y) = 0.58$ and is given by:

1. multivariate microaggregation with $k = 5$,
2. correlated noise addition with $\gamma = 0.05$,
3. rank swapping with $p = 0.08$, and
4. multiplicative noise with $s = 0.3$.

6.4.2 Utility assessment

We evaluate the utility of the anonymization methods for the parameters above that were found to yield the same level of disclosure risk. The utility is evaluated using the measures based on propensity scores and EMD, that were described in Section 2.8.

We found in Section 6.4.1 that, for the "Census" data set, the SDC methods being compared with parameters $k = 5$, $\gamma = 1$, $p = 0.2$ and $s = 0.5$, respectively, yielded the same risk of disclosure. By comparing the utility measures for these methods, we can determine which among them is preferable in this case. Table 6.1 shows the results for the propensity scores and EMD measures. Both utility measures are consistent and tell us that microaggregation has the best

Table 6.1: Utility loss measured using propensity scores (Equation (2.10)) and the earth mover's distance (Equation (2.11)) for the anonymization methods under test and for input parameters that were found to yield the same level of disclosure risk.

Methods	CENSUS		EIA	
	Propensity	EMD	Propensity	EMD
Microaggregation	4.28×10^{-4}	0.16	2.17×10^{-5}	0.040
Correlated noise addition	3.83×10^{-2}	0.38	4.22×10^{-5}	0.065
Rank swapping	3.51×10^{-3}	0.28	9.01×10^{-4}	0.091
Multiplicative noise	6.3×10^{-3}	0.29	9.85×10^{-5}	0.066

utility, followed by rank swapping, multiplicative noise and, finally, correlated noise addition.

For the "EIA" data set, the SDC methods being compared with parameters $k = 5$, $\gamma = 0.05$, $p = 0.08$ and $s = 0.3$, respectively, yielded the same risk of disclosure. The utility results for the propensity scores and EMD measures for this data set are shown in Table 6.1. Like in the other data set, methods are consistently ranked by the both measures, but the ranking is different: multivariate microaggregation has the best utility, followed by correlated noise addition, multiplicative noise, and, finally, rank swapping.

The results have shown that the SDC methods under comparison perform differently in different situations. Multivariate microaggregation always had the best utility (at the given level of disclosure risk), but the relative utility performance of the other methods changed between "Census" and "EIA". This shows that, unless there are good reasons for using a given anonymization method, it is usually better to make several anonymizations at the desired level of disclosure risk and select the one that has the greatest utility.

6.5 Summary

We have described a methodology to compare different anonymizations in terms of the risk-utility trade-off they attain. It is not enough to compare methods based on the level of risk or the utility they provide, because this gives only a partial picture.

We have proposed a disclosure risk measure based on record linkage and in the spirit of the permutation paradigm (which tells that disclosure risk control

comes essentially from rank permutation)

We have contributed an experimental analysis for two well-known data sets and four well-known anonymization methods. The results differ between data sets. As a conclusion from the experimental analysis, the best strategy seems to be to make several anonymizations at the desired level of disclosure risk and select the one that has the greatest utility.

Chapter 7

Conclusions and future work

7.1 Conclusions

In this thesis, we have presented several protocols (two of them variants of existing protocols, four of them new, plus two trivial benchmark protocols) for outsourcing two basic operations on sensitive numerical data vectors to untrusted clouds: scalar products and matrix products. Based on these operations, more complex data analyses can be performed, such as correlations and contingency tables. The goal is to minimize the amount of work that needs to be performed locally by the controller, who wants to use the cloud as much as possible to compute on her outsourced sensitive data. For the sake of flexibility and efficiency, we have considered non-cryptographic methods for data protection, such as data splitting and anonymization, rather than the heavier fully homomorphic encryption. A distinguishing feature of our approach is that the outsourced data on which the clouds compute retain some of the utility of the original data, which entails added value with respect to outsourcing encrypted or otherwise gibberish data.

If clouds can be assumed not to share information (perhaps because they do not know each other), data splitting is probably the best choice, due to simplicity and flexibility. We have proposed four protocols to compute on split data. In case clouds can share information but have no side knowledge on the original data set, we have proposed a sharing-resistant protocol based on orthog-

onal noise matrices that shifts most of the computational burden to the clouds. Finally, for the worst case, in which clouds share information and have side knowledge allowing them to recognize the original data set, we have proposed a sharing-resistant protocol relying on noise matrices derived via anonymization. Although the latter protocol is heavier, it still substantially relieves the controller (CLARUS) in computational terms.

If the data are nominal, multivariate analyses are even more challenging; the reason is not just their potentially large computational cost, but also the difficulty of performing the calculations involving data fragments stored in different clouds. In this thesis, we propose efficient protocols to securely compute statistical dependence analyses on split outsourced data for frequency-based and semantic-based tests.

We have provided complexity analyses and benchmarking for all proposed protocols, in order to show their computational advantages for the outsourcing controller. Further, we have provided experimental evidence that the new protocols take less effort from CLARUS than the benchmark protocols consisting of downloading and local processing. In this way, clouds are not only used to store sensitive data, but also to perform computations on these data in a privacy-aware manner. This is especially interesting for large sensitive data sets. For categorical data, experimental results clearly show that outsourcing the calculations to the cloud considerably decreases the workload of the data controller, who can save more than 99.999 per cent of the runtime for the most demanding test we considered.

Finally, we have described a methodology to compare different anonymizations in terms of the risk-utility trade-off they attain. It is not enough to compare methods based on the level of risk or the utility they provide, because that gives only a partial picture. We have proposed a disclosure risk measure based on record linkage and in the spirit of the permutation paradigm (which tells that disclosure risk control comes essentially from rank permutation). We have contributed an experimental analysis for two well-known data sets and four well-known anonymization methods. The results differ between data sets. As a conclusion from the experimental analysis, the best strategy seems to be to make several anonymizations at the desired level of disclosure risk and select the one that has the greatest utility.

7.2 Publications

Next, we enumerate the publications that back the contents of this thesis:

1. Domingo-Ferrer, J., Soria-Comas J., Ricci, S.: Disclosure risk assessment via record linkage by a maximum-knowledge attacker. In *13th Annual International Conference on Privacy, Security and Trust–PST 2015*, Izmir, Turkey, July 21-23, pp. 28-35. IEEE Computer Society (2015).

2. Calviño, A., Ricci, S., Domingo-Ferrer, J.: Privacy-preserving distributed statistical computation to a semi-honest multi-cloud. In *IEEE Conf. on Communications and Network Security – CNS 2015*, pp. 506-514. IEEE (2015).

3. Ricci S., Domingo-Ferrer J., Sánchez D.: Privacy-preserving cloud-based statistical analyses on sensitive categorical data. In *Modeling Decisions for Artificial Intelligence – MDAI 2016*, LNCS 9880, pp. 227–238. Springer (2016).

4. Domingo-Ferrer, J., Ricci, S., Soria-Comas, J.: A methodology to compare anonymization methods regarding their risk-utility trade-off. In *Modeling Decisions for Artificial Intelligence – MDAI 2017*, LNCS 10571, pp. 132-143. Springer (2017).

5. Domingo-Ferrer, J., Ricci, S., Domingo-Enrich, C.: Outsourcing scalar products and matrix products on privacy-protected unencrypted data stored in untrusted clouds. Information Sciences, vol. 436-437, pp. 320-342 (2018). Impact factor 4.832 (first decile).

6. Ricci, S., Muñoz-Batista, M., Sánchez, D., Domingo-Ferrer, J.: Outsourcing analyses on privacy-protected multivariate categorical data stored in untrusted clouds. In *Knowledge-Based Systems* (journal, second round of review).

7.3 Future work

Concerning scalar products and matrix products, we plan to combine the numerical methods presented in Chapter 3 to deal with data sets with heterogeneous attribute types. Other scenarios more challenging than the honest-but-curious CSPs assumption may also be considered for heterogeneous data, e.g., malicious or colluding clouds as in the case of numerical data (Chapter 4). Furthermore, we also intend to tackle outsourcing additional multivariate analyses for nominal data, such as multidimensional scaling, multiple correspondence analysis and non-linear principal component analysis.

Optimizing the splitting process is another research avenue worth pursuing. In our scenario, each CSP or CSP account stores a fragment of the orginal data set. In this thesis, we did not discuss the *right* number of attributes in each fragment. In fact, a fragment may contain one or several attributes. The number of attributes per fragment is related to some privacy and processing constraints. For instance, if we consider a medical data set, storing in the same fragment ZIP code, gender and ethnicity can be disclosive. In fact, this is shown in [68], where a 1990 federal census reports that in Dekalb, Illinois there were only two black women who resided in that town. Moreover, some attributes need to be stored in the same fragment for statistical analysis purposes. Therefore, as future work we can try to minimize the number of CSPs that are needed to store a data set using data splitting.

Bibliography

[1] Aggarwal, G., Bawa, M., Ganesan, P., Garcia-Molina, H., Kenthapadi, K., Motwani, R., Srivastava, U., Thomas, D., Xu, Y.: Two can keep a secret: a distributed architecture for secure database services. In CIDR 2005, pp. 186–199 (2005).

[2] Agresti, A., Kateri, M.: Categorical Data Analysis. Springer (2011).

[3] Amazon EC2 Instance Types. https://aws.amazon.com/ec2/instance-types/?nc1=h_ls

[4] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: A view of cloud computing. Communications of the ACM, 53(4):50–58 (2010).

[5] Atallah, M. J., Frikken, K. B.: Securely outsourcing linear algebra computations. In 5th ACM Symposium on Information, Computer and Communications Security – ASIACCS 2010, pp. 48-59. ACM (2010).

[6] Batet, M., Harispe, S., Ranwez, S., Sánchez, D., Ranwez, V.: An information theoretic approach to improve semantic similarity assessments across multiple ontologies. Information Sciences, vol. 283, pp. 197–2010 (2014).

[7] Batet, M., Sánchez, D.: A review on semantic similarity. In Encyclopedia of Information Science and Technology, Third Edition, pp. 7575–7583. IGI Global (2015).

[8] Big Data Value Association: BDVA Strategic Research and Innovation Agenda v3. visited on http://www.bdva.eu/?q=node/123

[9] Brand, R., Domingo-Ferrer, J., Mateo-Sanz,J.M.: Reference data sets to test and compare SDC methods for protection of numerical microdata. European Project IST-2000-25069 CASC (2002).

[10] California housing data set (cadata.txt), Louisiana State University, 2006. `http://statweb.lsu.edu/faculty/li/data/`

[11] California patient discharge data: California Office of Statewide Health Planning and Development (OSHPD), 2009. `http://www.oshpd.ca.gov/HID/DataFlow/index.html`

[12] Calviño, A., Ricci, S., Domingo-Ferrer, J.: Privacy-preserving distributed statistical computation to a semi-honest multi-cloud. In IEEE Conf. on Communications and Network Security – CNS 2015, pp. 506-514. IEEE (2015).

[13] Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. IEEE Transactions on parallel and distributed systems, 25(1):222-233 (2014).

[14] Cimiano, P.: Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer Science & Business Media (2006).

[15] Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Fragmentation and encryption to enforce privacy in data storage. In Computer Security – ESORICS 2007, Lecture Notes in Computer Science, vol. 4734, pp. 171–186, Springer (2007).

[16] Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Selective data outsourcing for enforcing privacy. Journal of Computer Security, 19(3):531–566 (2011).

[17] CLARUS - A Framework for User Centred Privacy and Security in the Cloud, H2020 project (2015-2017). `http://www.clarussecure.eu`

[18] Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.: Tools for privacy preserving distributed data mining. ACM SIGKDD Explorations Newsletter, 4(2):28–34 (2002).

[19] Coull, S., Collins, M., Wright, C., Monrose, F., Reiter, M.: On Web Browsing Privacy in Anonymized NetFlows. In Proceedings of 16-th USENIX Security Symposium. USENIX Association, Berkeley, CA, 23:1–23:14 (2007).

[20] Domingo-Ferrer, J., Muralidhar, K.: New directions in anonymization: permutation paradigm, verifiability by subjects and intruders, transparency to users. Information Sciences, 337:11-24 (2016).

[21] Domingo-Ferrer, J., Ricci, S., Domingo-Enrich, C.: Outsourcing scalar products and matrix products on privacy-protected unencrypted data stored in untrusted clouds. Information Sciences, vol. 436-437, pp. 320-342 (2018).

[22] Domingo-Ferrer, J, Ricci, S, Soria-Comas, J: A Methodology to Compare Anonymization Methods Regarding Their Risk-Utility Trade-off. In Modeling Decisions for Artificial Intelligence, pp. 132-143. Springer, Cham, 2017.

[23] Domingo-Ferrer, J., Sánchez, D., Rufian-Torrell, G.: Anonymization of nominal data based on semantic marginality. Information Sciences, vol. 242, pp. 35–48 (2013).

[24] Domingo-Ferrer, J., Torra, V.: A quantitative comparison of disclosure control methods for microdata. In: Confidentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies, pp. 111-134. North-Holland (2001).

[25] Domingo-Ferrer, J., Torra, V.: Ordinal, continuous and heterogeneous k-anonymity through microaggregation. Data Mining and Knowledge Discovery, vol. 11, no. 2, pp. 195–212 (2005).

[26] DOMO: Data never sleeps 5.0. visited on `https://www.marketingprofs.com/charts/2017/32531/the-incredible-amount-of-data-generated-online-every-minute-infographic`

[27] Du, W., Han, Y., Chen, S.: Privacy-preserving multivariate statistical analysis: linear regression and classification. In Proc. of the 2004 SIAM International Conference on Data Mining, pp. 222–233 (2004).

[28] Du, W., Zhan, Z.: A practical approach to solve secure multi-party computation problems. In Proceedings of the 2002 Workshop on New Security Paradigms, pp. 127–135. ACM (2002).

[29] Dubovitskaya, A., Urovi, V., Vasirani, M., Aberer, K., Schumacher, M.: A cloud-based e-health architecture for privacy preserving data integration. In ICT Systems Security and Privacy Protection. Springer, pp. 585–598 (2015).

[30] Durstenfeld, R: Algorithm 235: random permutation. Communications of the ACM 7(7):420 (1964).

[31] Dwork, C.: Differential privacy. Automata, Languages and Programming-ICALP 2006, LNCS 4052, pp. 1–12. Springer (2006).

[32] Ganapathy, V., Thomas, D., Feder, T., Garcia-Molina, H., Motwani, R.: Distributing data for secure database services. Transactions on Data Privacy, 5(1):253–272 (2012).

[33] Gelman A.: Analysis of variance—why it is more important than ever. The Annals of Statistics, vol. 33, no. 1, pp. 1–53 (2005).

[34] Gentry, C. Fully homomorphic encryption using ideal lattices. In 41st ACM Symposium on Theory of Computing – STOC 2009, pp. 169–178. ACM, 2009.

[35] Goethals, B., Laur, S., Lipmaa, H., Mielikäinen, T.: On private scalar product computation for privacy-preserving data mining. In Information Security and Cryptology – ICISC 2004, Lecture Notes in Computer Science, vol. 3506, pp. 104–120, Springer (2005).

[36] Goljan, M., Fridrich, J.: Camera identification from scaled and cropped images. In Proceedings of Electronic Imaging, Forensics, Security, Steganography, and Watermarking of Multimedia Contents, (2008).

[37] Greveler, U., Justus, B., Loehr, D.: Forensic content detection through power consumption. In Proceedings of the IEEE International Conference on Communications, Canada. IEEE Press, Psicataway, NJ, 6759–6763, (2012).

[38] Höhne, J.: Varianten von Zufallsüberlagerung (in German). Working paper of the project "Faktische Anonymisierung wirtschaftsstatistischer Einzeldaten" (2004).

[39] Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., Muehling, J., Pearson, J.V., Stephan, D.A., Nelson, S.F., Craig, D.W.: Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. PLoS Genetics 4 (8), (2008).

[40] Hundepool, A., Domingo-Ferrer, J., Franconi, L., Giessing, S., Nordholt, E.S., Spicer, K., De Wolf, P.P.: Statistical Disclosure Control. John Wiley & Sons (2012).

[41] Cloud computing (IBM), visited on `https://www.ibm.com/cloud/learn/what-is-cloud-computing`

[42] Ioannidis, I., Grama, A., Atallah, M.: A secure protocol for computing dot-products in clustered and distributed environments. In Proceedings of the International Conference on Parallel Processing. IEEE, pp. 379–384 (2002).

[43] IBM: 10 Key Marketing Trends for 2017. visited on `https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=WRL12345USEN`

[44] Kamara, S., Papamanthou, C., Roeder, T.: Dynamic searchable symmetric encryption. In 2012 ACM Conference on Computer and Communications Security – ACM CCS 2012, pp. 965–976. ACM 2012.

[45] Karr, A., Lin, X., Sanil, A., Reiter, J.: Privacy-preserving analysis of vertically partitioned data using secure matrix products. Journal of Official Statistics, 25(1):125-138 (2009).

[46] Lei, X., Liao, X., Huang, T., Li, H., Hu, C.: Outsourcing large matrix inversion computation to a public cloud. IEEE Transactions on Cloud Computing 1(1):78-87 (2013)

[47] Lei, X., Liao, X., Huang, T., Heriniaina, F.: Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud. Information Sciences 280:205-217 (2014)

[48] Lin, D.: An information-theoretic definition of similarity. In Proceedings of the 15th International Conference on Machine Learning, ICML 1998, pp. 296–304 (1998).

[49] Liu, F., Ng, W.K., Zhang, W. Encrypted scalar product protocol for outsource data mining. In IEEE CLOUD 2014, pp. 336-343. IEEE (2014).

[50] Nassar, M., Erradi, A., Sabry, F., Malluhi, Q. M.: Secure outsourcing of matrix operations as a service. In IEEE CLOUD 2013, pp. 918-925. IEEE (2014).

[51] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In Advances in Cryptology – EUROCRYPT '99, Lecture Notes in Computer Science, vol. 1592, pp. 223–238. Springer (1999).

[52] Rada, R., Mili, H., Bichnell, E., Blettner, M.: Development and application of a metric on semantic nets. IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, pp. 17—30 (1989).

[53] Ren, K., Wang, C., Wang, Q.: Security challenges for the public cloud. IEEE Internet Computing, 16(1):69–73 (2012).

[54] Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI, vol. 1, pp. 448–453 (1995).

[55] Ricci, S., Domingo-Ferrer, J., Sánchez, D.: Privacy-preserving cloud-based statistical analyses on sensitive categorical data. In Modeling Decisions for Artificial Intelligence, pp. 227–238. Springer (2016).

[56] Right Scale: 2017 State of the Cloud Survey, visited on `http://www.computerweekly.com/news/2240223928/UK-cloud-adoption-swells-by-61-in-four-years`

[57] Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In Proceedings of the 16th ACM conference on Computer and communications security 2009 Nov 9, pp. 199-212. ACM (2009).

[58] Rubner, Y., Tomasi, C. and Guibas, L.J.: The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision 40(2):99-121 (2000).

[59] Rodríguez-García, M., Batet, M., Sánchez, D.: A semantic framework for noise addition with nominal data. Knowledge-based Systems, vol. 112, pp. 103-118 (2017).

[60] Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transactions on Knowledge and Data Engineering, vol. 13, no. 6, pp. 1010-1027 (2001)

[61] Samarati, P., and Sweeney, L.: Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, SRI International (1998).

[62] Sánchez, D., Batet, M.: Privacy-preserving data outsourcing in the cloud via semantic data splitting. Computer Communications, vol. 110, pp. 187–201 (2017).

[63] Sánchez, D., Batet, M., Isern, D., Valls, A.: Ontology-based semantic similarity: a new feature-based approach. Expert Systems with Applications, vol. 39, no. 9, pp. 7718–7728 (2012).

[64] Sánchez, D., Batet, M., Isern, D.: Ontology-based information content computation. Knowledge-based Systems, vol. 24, no. 2, pp. 297-303 (2011).

[65] Sánchez, D., Batet, M., Martínez, S., Domingo-Ferrer, J.: Semantic variance: an intuitive measure for ontology accuracy evaluation. Engineering Applications of Artificial Intelligence, vol. 39, pp. 89–99 (2015).

[66] SNOMED-CT Ontology. `https://en.wikipedia.org/wiki/SNOMED_CT`

[67] Sun, Y., Yu, Y., Li, X., Zhang, K., Qian, H., Zhou, Y.: Batch verifiable computation with public verifiability for outsourcing polynomials and matrix computations. In Australasian Conference on Information Security and Privacy – ACISP 2016, Lecture Notes in Computer Science, vol. 9722, pp. 293–309. Springer (2016).

[68] Sweeney, L.: Simple demographics often identify people uniquely. Health (San Francisco);671:1-34, 2000.

[69] Székely, G.J., Rizzo, M.L.: Brownian distance covariance. The Annals of Applied Statistics, vol. 3, no. 4, pp. 1236–1265 (2009).

[70] Torra, V., Domingo-Ferrer, J.: Record linkage methods for multidatabase data mining. In: Information Fusion in Data Mining, pp. 101-132. Springer (2003).

[71] Trouessin, G.: Traitements fiables de données confidentielles par fragmentation-redondance-dissémination. Ph.D. dissertation, Université de Toulouse 3 (1991).

[72] U.S. Federal Trade Commission: Data Brokers, A Call for Transparency and Accountability (2014).

[73] Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In Proc. of the 9th International Conference on Knowledge Discovery and Data Mining – KDD 2003, pp. 206-215. ACM (2003).

[74] Wang, I.-C., Shen, C.-H., Hsu, T.-S., Liao, C.-C., Wang, D. W., Zhan, J.: Towards empirical aspects of secure scalar product. IEEE Trans. Systems, Man and Cybernetics, Part C, 39(4):440–447 (2009).

[75] Weiss, G.: Data mining in the real world: experiences, challenges, and recommendations. In DMIN, pp. 124–130 (2009).

[76] Winkler, W.E.: Matching and Record Linkage. John Wiley & Sons, Inc. (1995).

[77] Woo, M.J., Reiter, J.P., Oganian, A., Karr, A.F.: Global measures of data utility for microdata masked for disclosure limitation. Journal of Privacy and Confidentiality 1(1):7 (2009).

[78] Wu, Z., Palmer, M.: Verbs semantics and lexical selection. Proceedings of the Annual Meeting of the Association for Computational Linguistics, pp. 133—139 (1994).

[79] Yang, Q., Wu, X.: 10 challenging problems in data mining research. International Journal of Information Technology & Decision Making, 5(4):597–604 (2006).

UNIVERSITAT ROVIRA I VIRGILI
OUTSOURCING COMPUTATION ON NON-ENCRYPTED SENSITIVE DATA TO UNTRUSTED CLOUDS
Sara Ricci

UNIVERSITAT ROVIRA I VIRGILI
OUTSOURCING COMPUTATION ON NON-ENCRYPTED SENSITIVE DATA TO UNTRUSTED CLOUDS
Sara Ricci

UNIVERSITAT
ROVIRA i VIRGILI