UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Big Data Privacy by Design Computation Platform

Rui Nuno Lopes Claro

Supervisor: Doctor Miguel Filipe Leitão Pardal
Co-Supervisor: Doctor José Miguel Ladeira Portêlo

Thesis to obtain the Masters Degree in
Information Systems and Computer Engineering

2018

Resumo

Abstract

We live in the age of Big Data. Personal user data, in particular, is necessary for the operation and improvement of everyday Internet services like Google, Facebook, WhatsApp, Spotify, etc. Many times, the capture and use of personal data is not made explicit to the users, but it is central to the business model of companies. However, each person's right to privacy has to be respected. How can these two conflicting needs be reconciled, *i.e.*, how can we build useful Big Data systems that are respectful of user privacy? The goal of this work is to design and implement a proof-of-concept of a platform for performing privacy preserving computations, providing an easy-to-use method to implement privacy-preserving techniques. This system could be used, for example, to monitor the vital signs of patients (without exposing them to other people), to produce real time recommendations based on location (without disclosing location to others), sports/fitness applications, etc. This proof-of-concept will implement privacy-preserving versions of Machine Learning algorithms and compare them against a baseline reference, allowing a better understanding of the trade-offs of using this technology.

so the trade-offs can be quantified and better understood.

Palavras-Chave
Keywords

Palavras-Chave

Preservação de privacidade em Computações
Aprendizagem automática
Extracção de Informação
Big Data
Processamento de Dados
Computação Multi-Entidade Segura

Keywords

Privacy-preserving Computations
Machine Learning
Data Mining
Big Data
Data Processing
Secure Multi-Party Computation

Acknowledgements

Contents

List of Figures

List of Tables

List of Acronyms

PPDM Privacy-Preserving Data Mining

PII Personal Identifiable Information

SPI Sensitive Personal Information

PPML Privacy-Preserving Machine Learning

SMPC Secure Multi-Party Computation

STPC Secure Two-Party Computation

OT Oblivious Transfer

GC Garbled Circuits

DP Differential Privacy

HE Homomorphic Encryption

PHE Partial Homomorphic Encryption

FHE Fully Homomorphic Encryption

FE Functional Encryption

ENISA European Union Agency for Network and Information Security

EU European Union

GDPR General Data Protection Regulation

DT Decision Tree

k-NN k-Nearest Neighbors

SVM Support Vector Machine

k-M k-Means

LR Logistic Regression

CMU Carnegie Mellon University

ML Machine Learning

BARD Big dAta pRivacy by Design platform

EMR Eletronic Medical Records

1 Introduction

With the so called "Big Data revolution", vast amounts of data are now being analyzed and processed by companies that take advantage of the enormous quantities of information that is generated every day[1]. Big Data and Business Analytics market reflects this growth rate, expecting to hit the $210 billion mark in the year 2020[2]. Through this data processing, meaningful information can be obtained to improve existing systems or to discover new approaches in business models. An example of this is the deployment of *Data Mining* algorithms to better understand their customers, and to devise better recommendation systems, in order to surpass their competitors in customer satisfaction. Another example lies in the field of healthcare, where it can be beneficial to match patient records from different hospitals in order to identify inefficiencies and develop best practices [29].

Most times data contains private information about individuals, such as health records or daily routines. This kind of data cannot be freely processed because that leads to breaches of private information, such as the AOL Search Leak or the Microsoft Hotmail privacy breach[3]. Due to these breaches, and despite the value that Data Mining adds to businesses and medical systems, consumers show an increasing concern in the privacy threats posed by Data Mining [7]. The privacy of an individual may be violated due to, for example, unauthorized access to personal data, or the use of personal data for purposes other than the one for which data was collected.

To deal with the privacy issues in Data Mining, a sub-field known as Privacy-Preserving Data Mining (PPDM) has been gaining influence over the last years [9]. The objective of PPDM

[1] http://www.vcloudnews.com/every-day-big-data-statistics-2-5-quintillion-bytes-of-data-created-daily/

[2] https://www.idc.com/getdoc.jsp?containerId=prUS42371417

[3] https://www.networkworld.com/article/2185187/security/15-worst-internet-privacy-scandals-of-all-time.html

is to guarantee the privacy of sensitive information, while at the same time preserve the utility of the data for Data Mining purposes [1]. This can be achieved by using one or more privacy-preserving techniques, such as Differential Privacy (DP) [10] or Secure Multi-Party Computation (SMPC) [9].

Machine Learning (ML) algorithms in the context of Big Data processing are also producing significant results, so that it is possible to do knowledge learning from datasets in order to predict future labels (*i.e.* classes of data) or clusters (groups of related data) for new data. An example of an application of ML algorithms in Data Mining is *Classification* [27], in which a training set is processed in order to create a classifier for data, and then that classifier is used to predict class labels for new data. These applications show a greater impact in the field of medicine as mentioned above, with Google's DeepMind building ML algorithms to process admissions in hospitals faster[4], and with IBM's Watson supporting medical personnel consider treatment options for their patients[5]. Some examples of ML algorithms include Decision Tree, k-Nearest Neighbors (k-NN), and Support Vector Machine (SVM) [27].

muito pequeno, expandir, explicar melhor

manter esta ultima frase? ou excluir?

By combining ML algorithms and privacy-preserving techniques, it is possible to create Data Mining processes that, not only allow for knowledge learning on large datasets but also to maintain a level of privacy that is desirable by individuals and that complies with the laws in force [9].

1.1 Contributions

The main contribution of this thesis is the design and creation of a proof-of-concept platform for privacy-preserving distributed ML computations without resorting to third parties. With it, we aim to give the users a ML platform without having to build their own, meaning less maintenance costs derived of maintaining code developed *in-house*, dedicated servers, or even dedicated personnel. And since the platform has its foundations on privacy-preserving techniques, it complies with the level of privacythat those users want for their data.

"level of privacy" não fica bem.. algo melhor??

[4] https://deepmind.com/applied/deepmind-health/
[5] https://www.mskcc.org/about/innovative-collaborations/watson-oncology

We show in

ref to use case

possible usages of this platform, detailing the context of the data to process, the trade-offs that are required, etc.

Expand Contributions

1.2 Structure of this Document

The remainder of this thesis is structured as follows.

2 Related Work

This chapter provides and overview on the privacy-preserving Machine Learning paradigm. We start by defining Data Security and Data Privacy, in sections 2.1 and 2.2 respectively. We detail each concept so that it gives an understanding of the differences between them. Section 2.3 presents the concepts of data processing and Data Mining, gives an overview of the CRISP-DM model, and defines the attack models that can be assumed when developing a Privacy-Preserving Data Mining (PPDM) algorithm. For developing a PPDM algorithm, one can implement one or more of the privacy-preserving techniques briefly presented in section 2.4. We present Machine Learning applied in Data Mining in section 2.5. Finally, we discuss known use cases that show what can be achieved in the field in section 2.6.

2.1 Data Security

Data Security refers to protective digital measures that are applied to prevent unauthorized access to computers, databases, and websites, as well as to prevent data destruction or alteration.

2.1.1 Data Protection Goals

We define here the core protection goals widely accepted in the literature, often known as the CIA triad (*Confidentiality*, *Integrity*, and *Availability*) [23].

- *Confidentiality* is defined as the property that data, and services that process such data, cannot be accessed by unauthorized entities.

- *Integrity* is defined as the property that data, and services that process such data, cannot be modified in an unauthorized or undetected manner.

- *Availability* is defined as the property that access to data, and services that process such data, is always possible when needed by the authorized parties and in a timely manner.

For applying Data Security measures, various technologies can be implemented, such as:

- *Data backups* ensure that data that has been lost can be recovered. This technique is standard procedure for most companies since the permanent loss of crucial data can seriously cripple a company's business.

- *Data erasure*, in contrast to backups, is a technique to permanently delete data from a hard drive or other digital media, to ensure that no sensitive data is leaked when a company wants to permanently remove an asset from usage or when required by court order.

- *Data encryption*, or disk encryption, refers to techniques that allow a user to encrypt data in a disk or part of it, such that it remains protected and cannot be decrypted easily by an unauthorized party.

- *Identity-based security* is a method to limit the access to data such that only a user that has been authenticated and has permission to access a piece of data can do so.

These techniques offer ways to protect data, but sometimes this is not enough. Usually due to incorrect programming that cause bugs in the system, vulnerabilities occur in the software that allows unauthorized parties to bypass these mechanisms and get access to data that should be confidential.

2.1.2 Examples of Data Security Breaches

Data Security breaches refer to attacks, usually by unauthorized access, to systems that contain private data. These attacks are commonly made by organized hacker groups to gain leverage against companies or to make a profit by selling the data in black markets. Next, we present some examples of recent Data Security breaches.

- Sony Pictures hack[1]. In 2014, a hacker group leaked confidential data from Sony Pictures in an attempt to gain leverage with the company to make it comply to their demands. The hacker group threatened to commit acts of terrorism in theaters if Sony released a movie related to the North Korean leader.

- Yahoo! data breach[2]. In 2016, Yahoo! reported two separate data breaches occurring in 2014 and 2013, of over 1.5 billion user accounts, including Yahoo! email access, which in turn can reveal bank and family details as well as passwords for other services.

- Ashley Madison data breach[3]. In 2015, a group of hackers stole user data from the adultery website Ashley Madison, and threatened to release usernames and personally identifying information if the website was not shut down.

2.2 Data Privacy

Privacy is an important field in information security because it gives a person his/her personal space and defines his/her personal private information, giving the person the right to decide which personal information is for sharing and which should be kept confidential. Privacy also limit the access that other entities, being them the government or private companies, have to personal data.

[1] https://www.washingtonpost.com/news/the-switch/wp/2014/12/18/the-sony-pictures-hack-explained/

[2] https://www.theguardian.com/technology/2016/dec/14/yahoo-hack-security-of-one-billion-accounts-breached

[3] http://fortune.com/2015/08/26/ashley-madison-hack/

2.2.1 Privacy Definitions

Privacy can be defined as the ability or right that an individual has of protecting his/her personal information and extends to the ability or right to prevent invasions on the personal space of said individual [2].

One of the prime examples of Privacy applied to information technology problems and mentioned in the literature is related to Eletronic Medical Records (EMR) [29]. These records must be handled with extra care because they contain a large number of sensitive information about the patients. The Patient Record Systems should be able to disclose information only to selected personnel, but not all the information about the patient, only what is necessary to proceed in helping the patient. This example illustrates the tension between having access to the data, that can be useful, but at the same time keeping it closed to other users.

2.2.2 Privacy Protection Goals

New concepts have arisen in recent years for privacy specific protection goals [10]. Their definitions are as follows:

- *Unlinkability* is defined as the property that ensures privacy-relevant data cannot be linked across domains that are constituted by a common purpose and context. In other words, multiple actions from the same user/entity must be unlinkable.

- *Transparency* is defined as the property that ensures all privacy-relevant data processing can be understood and reconstructed at any time. Transparency has to cover not only the actual processing but also the planned processing and after processing to fully know what has happened and which entities were involved. Transparency is related to the principles concerning openness and it is a prerequisite to accountability. The user/entity must know and understand how his/her private data is being handled.

- *Intervenability* is defined as the property that ensures mediation is possible concerning

privacy-relevant data processing, in particular by those persons whose data is being processed. Intervenability is related to the rights of an individual, in a way that the owner of privacy-relevant data must have the means to rectify or erase said data.

2.2.3 European Union Legislation

It is also important to mention in the context of this work the current legislation in the European Union (EU) regarding data protection.

The Data Protection Directive[4] is the current law regarding privacy in the EU and is in force since 1995. More recently a replacement has been proposed and accepted in the EU, the General Data Protection Regulation (GDPR)[5], that will take effect in May 2018. Both these laws are regulated by European entities, namely the European Union Agency for Network and Information Security (ENISA)[6] and the Article 29 Data Protection Working Party[7].

According to ENISA, EU data protection law applies to any processing of personal data [9]. This personal data is defined as any information related to an identified or identifiable natural person. In the context of Big Data analysis, the focus is more on indirect identification, which translates into three different approaches: *i)* The possibility of isolating some or all records which identify an individual in a dataset; *ii)* The linking of at least two records concerning the same individual in the same database or in different databases; and *iii)* The possibility to infer the value of an attribute in a dataset from the value of other attributes.

Another important cornerstone of EU data protection law are the principles relating to data quality:

- The *fairness principle* requires that personal data should never be processed without the individual being actually aware of it.

[4] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:31995L0046
[5] http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2016.119.01.0001.01.ENG&toc=OJ:L:2016:119:TOC
[6] https://www.enisa.europa.eu/
[7] http://ec.europa.eu/newsroom/just/item-detail.cfm?item_id=50083

- The *purpose limitation principle* implies that data can only be collected for specified, explicit and legitimate purposes.

- The *data minimization principle* states that data processed should be the one which is strictly necessary for the specific purpose previously determined by the data controller.

These three principles together mandate that data processing must be done with the consent of the subject, informing the subject of what is the purpose of the processing, and do not deviate from this purpose without informing the subject.

Finally, and also important to mention in the context of this work, are the rights of the data subject according to the EU law. There are two important rights that a subject has: the *right of access* and the *right to object*. The *right of access* ensures that any data subject is entitled to obtain from the data controllers communication of the data that is subjected to processing and to know the logic involved in any processing of data concerning him/her. This is particularly relevant in the context of Big Data analysis because it limits technological lock-ins and other competition impediments, and it enhances transparency and trust between users and service providers. The *right to object* ensures that data subjects have a right to revoke any prior consent, and to object to the processing of data relating to them, giving to the subject the power to remove himself completely or partially to any data processing mechanisms using his personal data.

2.2.4 Examples of Data Privacy Breaches

In recent years we can find a number of attacks made to systems that handle personal information. Next are a number of examples found relevant regarding Data Privacy breaches.

- Target Pregnancy Leak[8]. In 2012, Target, an American retail company, started merging data from user searches and demographics data in order to learn when their customers

[8] https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/#3001668b6668

were pregnant, in order to approach them with a specific advertisement. This is a clear violation of private sensitive information about their customers and their private life.

- Netflix Prize[9]. In 2007, Netflix created a contest to improve their recommendation system. For that, they released a training dataset, with all the personal information regarding customers removed and customer *ids* replaced by randomized *ids*. Later it was shown that it was not enough when a group of researchers linked public information in another movie-rating website (IMDB) with the released dataset and were able to partially de-anonymize the training dataset, compromising the identity of some users.

- AOL search data leak[10]. In 2006, AOL released to the general public a text file containing search keywords for numerous of their users, intended for research purposes. The users were not identified, but personally identifiable information was present in many of the queries. These queries contained a user *id* attributed by AOL, and an individual could be identified and matched to their account and search history by such information.

- Massachusetts GIC medical encounter database[11]. A researcher from Carnegie Mellon University (CMU) linked the anonymized database (which contained birth date, sex and ZIP code) with voter registration and was able to link medical records with individuals.

2.3 Privacy Implications of Personal Data Processing

Data processing is the conversion of raw data to meaningful information through a process. Data is manipulated to produce results that lead to a resolution of a problem or improvement of an existing situation.

Data mining is the process of discovering interesting patterns and knowledge from large amounts of data [22]. We will describe the Data Mining process according to the widely used

[9] https://www.wired.com/2009/12/netflix-privacy-lawsuit
[10] https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data
[11] https://techpinions.com/can-you-be-identified-from-anonymous-data-its-not-so-simple/7627

CRISP-DM model [46], in which the process is separated into six major phases, as described next and in Figure 2.1.

- **Business Understanding:** In this initial phase, the project goals and requirements must be understood from a business perspective, and then converted into a Data Mining problem.

- **Data Understanding:** During this phase, an initial data collection is done, followed by a number of activities in order to get familiar with the data, to understand how the data is organized, to identify if the data has quality problems, or even to detect interesting subsets in the data collected.

- **Data Preparation:** This phase covers all the data preparation tasks to construct the final dataset from the initial raw data collected. These tasks include attribute selection, data cleaning, and transformation of data to fit the modeling tools.

- **Modeling:** In this phase, various modeling techniques are selected and applied, and the underlying parameters are calibrated to optimal values. Usually, there are several techniques for the same Data Mining problem type, and some of these techniques require specific data formats.

- **Evaluation:** At this point in the Data Mining process, one or more models have been developed that appear to have high quality, from a data analysis perspective. These models must be evaluated thoroughly so that we can be certain that they properly achieve the business goals.

- **Deployment:** In this last phase, the knowledge gained by the Data Mining process will need to be organized and presented in a way that the customer can use it. This, of course, depends on the requirements presented at the beginning of the process. An example of a common deployment that results from Data Mining is a simple report on the knowledge obtained.

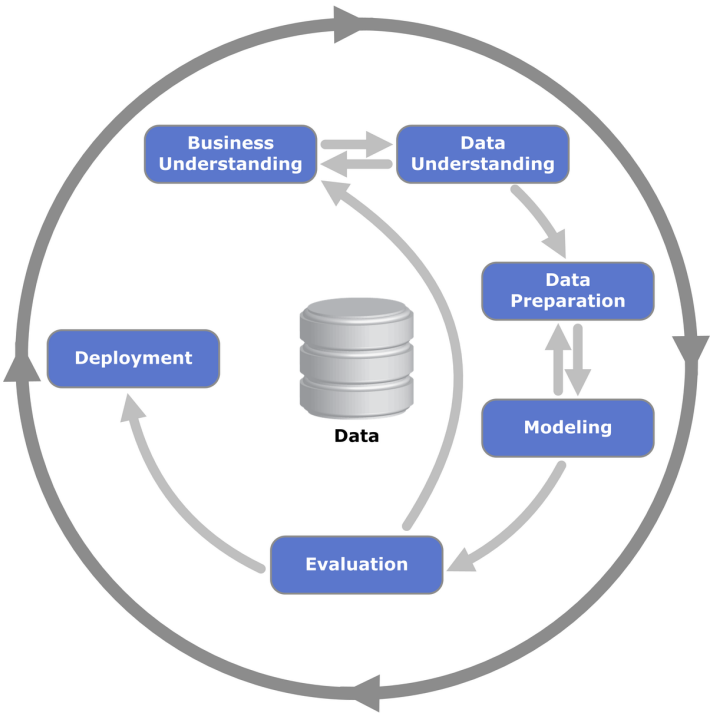


Figure 2.1: Process diagram showing the relationship between the different phases of CRISP-DM [46].

In the Data Mining process, we must be aware that, sometimes, private information about individuals can be used and it may lead to breaches of privacy. We define this private information as Personal Identifiable Information (PII) or Sensitive Personal Information (SPI) [38]. This concept can be defined as information that can be used alone or in conjunction with other information to identify, contact, or locate a single person, or to identify an individual in a context.

2.3.1 Attack Models

When considering the security of a system, one must have into account the concepts of threat model, attack model, and the two types of adversaries, *honest-but-curious* and *malicious*, to understand how to better implement an efficient and trustworthy security layer.

A threat model specifies the potential threats that can be used against a system, by identifying, enumerating and prioritizing them. A well known and defined threat assessment model

is the STRIDE model developed by Microsoft [21].

Cryptographic attacks are used whenever the target system relies on cryptography for protection. An attack model is, in terms of cryptanalysis, a classification of cryptographic attacks specifying the type of access the attacker has to a system when attempting to break an encrypted message. We can summarize the cryptographic attack in the following four categories:

- **Ciphertext-only attack:** In this type of attack, the cryptanalyst has access only to the ciphertext and has no access to the plaintext. This is the most common type of attack, and it is a requirement for modern ciphers to be resistant to it. An example of a ciphertext-only attack is the brute force attack, where the attacker makes a trial-and-error approach to decrypt the ciphertext.

- **Known-plaintext attack:** In this type of attack, the cryptanalyst has access to a number of pairs of plaintext and the corresponding ciphertext.

- **Chosen-plaintext attack:** In this type of attack, the cryptanalyst is able to encrypt arbitrary plaintext and have access to the resulting ciphertext, allowing him to make a statistical analysis on the plaintext state space.

- **Chosen-ciphertext attack:** In this type of attack, the cryptanalyst is able to choose arbitrary ciphertext and obtain the corresponding plaintext.

We can also classify the types of adversaries according to their strategies in two major groups: *honest-but-curious* and *malicious* adversaries. The honest-but-curious adversary follows the protocol in plan, but he will try to extract information from his viewpoint of the protocol to gain some form of advantage or gain access to confidential information. The malicious adversary is one that can deviate from the protocol specification as he desires, and will try to disrupt and/or collect as many information as he can.

2.4 Privacy-Preserving Techniques

In this section, we will describe some of the techniques used in preserving Data Privacy, namely anonymization, Differential Privacy (DP), Secure Multi-Party Computation (SMPC), Garbled Circuits (GC), Oblivious Transfer (OT), Homomorphic Encryption (HE) and Functional Encryption (FE).

2.4.1 Anonymization

Data anonymization is a type of information sanitization that has the final intent of privacy protection. This can be achieved by either removing or encrypting PII from datasets so that the individuals whom the data describe remain anonymous [35].

Anonymization techniques use a variety of approaches, for example, suppression, where a piece of information (*e.g.*, name, age) is removed from the dataset; generalization, where data is coarsened into less refined sets; perturbation, where data is modified by adding noise; and permutation, where sensitive associations between entities in the dataset are swapped.

The goals behind anonymization of data are tightly intertwined with the privacy goals we want to achieve for the data being processed. Usually, one or more techniques mentioned above are applied to the data until certain properties are met, for example, *k-anonymity*, *ℓ-diversity* or *t-closeness*.

- *k-anonymity* property states that, for each individual whose data is released in some dataset, he must not be distinguishable from at least k individuals that are also present in the release [43].

- *ℓ-diversity* property is an extension of the *k-anonymity* property, and furthers the anonymization of data by reducing the granularity of the data representation such that for any given record, there exists at least ℓ different sensitive attribute values, in addition to the guarantees made by *k-anonymity* [30].

- *t-closeness* property is a further refinement of *ℓ-diversity*. This property requires that the distribution of a sensitive attribute in any equivalence class is close to the distribution of the attribute in the overall table, effectively limiting the amount of individual-specific information an observer can learn [28].

2.4.2 Differential Privacy

The concept of Differential Privacy (DP) arose due to recent Data Privacy breaches mentioned in section 2.2.4. In [14] it is shown that the security standard for statistical databases, that states that access to a statistical database should not enable one to learn anything about an individual that could not be learned without access, is not achievable because of the existence of auxiliary information, *i.e.*, information available from sources other than the statistical database.

DP is a process of maximizing the accuracy of queries in statistical databases while minimizing the chances of identifying its records. The core of the procedure is based on *randomized response* [45], giving the possibility to infer statistical information from the dataset, while still ensuring high levels of privacy.

Detailed information about DP and algorithms designed to achieve it are described in [15].

2.4.3 Secure Multi-Party Computation

fica bem distributed aqui?

Secure Multi-Party Computation (SMPC) (also known as secure computation, multi-party computation, or privacy-preserving computation) is a protocol for distributedparties to jointly compute a function over their inputs while keeping those inputs private. The problem of computing functions while also preserving the privacy of the inputs is referred in the literature as a SMPC problem [47]. Generally speaking, a SMPC problem deals with computing any probabilistic function on any input, while also ensuring the correctness of the computation and guaranteeing that no more information is revealed to a participant in the computation that can be inferred from that participant's input and output [19].

A strategy to solve these problems is to trust an external entity (a trusted third party), that can mediate the computation. This approach can be risky because it requires a third party that all participants agree to trust, which can sometimes be difficult to find. Sometimes, the data has such high degree of importance to the participants that even disclosing them to a trusted third party is not viable.

When building a SMPC protocol, the most important properties that must be ensured are *input privacy* and *correctness* [18]:

- *Input privacy* property states that no information about the private data held by the parties can be inferred during the execution of the protocol. The only inferences about private data are those that could be inferred from seeing the output of the computation made by the protocol.

- *Correctness* property relates to the existence of malicious parties that could try to deviate from the normal functioning of the protocol. In these cases, the protocol should prevent honest parties to output incorrect results. The approach to implementing the correctness property comes in two alternatives: either the protocol is robust, *i.e.* it guarantees that the honest parties compute the correct output; or the honest parties abort the computation if they find an error during the execution of the protocol.

The first implementation of secure computation was introduced as Secure Two-Party Computation (STPC) [47]. It is a simplification of the problem of SMPC, and a known protocol for STPC is Yao's GC protocol, which is detailed in subsection 2.4.5. In STPC there are only two participants in the computation, and usually one is responsible for starting and encoding the computation mechanism, while the other is the one responsible for evaluating the computation. In multi-party computation, the parties have no special roles and, instead, the encoding is shared amongst the parties, by secret sharing, and the evaluation is made by a protocol.

MP:add some more text about the protocol.

Recent implementations of SMPC protocols are based on *secret sharing*. Secret sharing allows

one party to distribute a secret over a number of parties by distributing shares to each party. Three of the types of secret sharing techniques more commonly used are: Shamir Secret Sharing [39], Additive Secret Sharing and Replicated Secret Sharing.

2.4.4 Oblivious Transfer

Oblivious Transfer (OT) [34] is a protocol in which a sender transfers one of the potentially many pieces of information he has to a receiver, but remains oblivious as to what piece has been transferred. Let s^0 and s^1 be two strings held by a sender that wants to transfer one of the strings to a receiver, holding a selection bit b; the protocol allows for only one of the inputs s^b to be transfered; the receiver learns nothing about s^{1-b}, and the sender does not learn b.

An interesting implementation of OT is the one done by Pinkas and Naor [31]. In it, the authors describe an extension to the basic 1-out-of-2 OT protocol, to a 1-out-of-N protocol, and a k-out-of-N protocol.

2.4.5 Garbled Circuits

Yao's Garbled Circuits (GC) [48] is a cryptographic protocol that allows two mistrusting parties to evaluate a function without resorting to a trusted third party. In other words, GC allow parties holding input x and y to evaluate an arbitrary function $f(x, y)$ without leaking any information about their inputs beyond what is inferred from the function output. The idea behind GC is that one party prepares an encrypted version of a circuit that computes f and the second party then computes the output of the circuit without learning any intermediate values.

Some optimizations have been proposed for Yao's GC. Kolesnikov and Schneider [26] present a technique that eliminates the need to garble XOR gates. Pinkas *et al.* present a technique that reduces the size of a garbled table from four to three ciphertexts [33].

2.4.6 Homomorphic Encryption

Homomorphic Encryption (HE) [36] is a cryptographic technique that allows computations to be carried with the ciphertext, so that, when decrypted, the resulting plaintext reflects the computation made. In other words, HE allows to make some computation over the ciphertext, for example, addition, without decrypting it, and the result is the same as making that computation on the plaintext. This is of great importance because it allows chaining multiple services that make computations on a ciphertext, without the need to expose the data to those services. A prime example of HE in real-world problems is electronic voting [25].

MP:It is not clear why evoting is an example of HE

Homomorphic cryptosystems can be classified into two distinct groups: partially homomorphic cryptosystems and fully homomorphic cryptosystems.

- In Partial Homomorphic Encryption (PHE), there is a operation or operations, like addition or multiplication, that offer the homomorphic property, but not for all possible operations. Some examples of existing partially homomorphic cryptosystems are: ElGamal cryptosystem [16]; Unpadded RSA [37]; and Pailier cryptosystem [32].

- In Fully Homomorphic Encryption (FHE), it is possible to make any arbitrary computation on the ciphertext. This concept was first introduced in [36], and for many years it remained just as a concept, until recent years when fully homomorphic implementations were developed, for instance, Gentry's cryptosystem [17].

2.4.7 Functional Encryption

In Functional Encryption (FE) systems, a decryption key allows a user to learn a specific function of the encryption data, while also stopping that same user from learning anything more about the encrypted data. In other words, having a secret key only allows for a specific computation of a function over the ciphertext [5]. When comparing FE with HE, the main

difference is that, in HE, the result of a computation in the ciphertext remains encrypted, while in FE, the result of the computation is available in the clear.

2.5 Privacy-Preserving Machine Learning

In the context of Data Mining, Machine Learning techniques come as an important addition to the data processing step. An example of that is the *Classification* method. Classification is described by a two-step process, in which a classification algorithm is employed to build a classifier for the data by analyzing a training set made of tuples of data and their associated labels, and then the classifier is used to predict class labels for new data. As such, because the large size of the datasets produced in Big Data operations, classification algorithms have a large quantity of data to learn from, making them less prone to erroneous classification of new data.

The conjunction between Machine Learning and privacy-preserving comes from the need to do knowledge learning over large datasets, while also maintaining protection over the privacy of the data, without degrading the quality of data by using anonymization techniques.

In the field of Machine Learning, we can identify a number of algorithms that can be used in Data Mining. In Table 2.1, we list some of them, giving a brief description, and identifying a privacy-preserving technique that has been implemented with it.

Table 2.1: Privacy-Preserving Machine Learning (PPML) algorithms.

Machine Learning Algorithm	**Short summary**	**Privacy-Preserving Technique**	**Reference**
Decision Tree	Protocol for distributed learning of decision-tree classifiers.	SMPC	[8]
Naive Bayes	Differentially private naive Bayes classifier. Centralized access to the dataset.	DP	[44]
SVM	Algorithm for support vector machine classification over vertically partitioned data.	SMPC	[49]
k-NN	Nearest neighbors of records in horizontally distributed data.	SMPC	[40]
k-Means	k-Means clustering based on additive secret sharing.	SMPC	[13]

maybe add logistic regression in this table?

2.6 Use Cases

In terms of PPML and its applications, it is important to distinguish the subject of the data that is being processed. Different data can be subject to different constraints regarding laws and privacy. Some sensitive data may be only processable in a local environment, while other data can only be processed in a less individualized way.

We now detail three relevant and different subjects that are subject to different privacy constraints.

- **Health records:** Healthcare systems are one of the examples where vast amounts of data are collected every day, and it is of relevance to do Data Mining on patient records, for a better understanding of patients and to improve the healthcare system. Patient records contain very sensitive information about individuals and cannot be processed without the Data Mining system being in compliance with the legislation on Data Privacy, therefore it is of interest to build privacy-preserving systems for the healthcare system, so that hospitals and other health-related organizations can share and infer knowledge without violating the privacy of their patients.

- **Governance (students and taxes):** Bogdanov *et al.* [4] made a statistical study in 2015 using SMPC to look for correlations between working during university studies and failing to graduate in time. For this study, it was necessary to link the database of individual tax payments and the database of higher education universities. These types of government data are subject to strict legislation and cannot simply be handled without strong privacy guarantees. To solve this problem, a SMPC system was developed and deployed that could assure a level of privacy that would be in compliance with the laws on Data Privacy. The data processing steps were all made using SMPC between three parties, using OT so that each party would not know each other inputs.

  In the end, the study using SMPC was compared with an anonymized study using 3-anonymity. The loss of samples in the latter was 10%-30%, depending on the demographic group, thus suggesting that producing studies on existing databases using SMPC to enforce privacy can give more accurate results than the same study run using k-anonymity measures.

- **Human mobility:** Another subject that provides great challenges in the field of Data Privacy are the mobility traces generated by people when driving, walking, etc. Mobility traces are highly unique so it is possible, even after anonymizing the dataset, to link an individual to his mobility patterns, as shown by Montjoye *et al.* [11]. Since mobility data contains the approximate whereabouts of individuals, it can be used to reconstruct their

movements across space and time. Applying privacy-preserving techniques to process this highly sensitive data can result in better geographic-based recommendation systems.

2.7 Summary

The previous sections provided an overview of the state of the art surrounding privacy-preserving Machine Learning.

expand

3 Implementation

When building a platform in the scope of privacy-preserving Machine Learning (ML), we must consider not only the traditional steps in data processing mentioned in section 2.3 and in Figure 2.1, but also have an increased care when preprocessing data to incorporate the cryptographic techniques that we used. This chapter describes the work that was done in implementing a privacy-preserving ML platform. We start off with a description of the datasets chosen to evaluate our platform 3.1. Then, we explain the preprocessing that was done to those datasets 3.2. Section 3.3 presents the baseline implementation of the chosen ML algorithms, resorting to a widely used ML toolkit for Python. In section 3.4 the implementation of the prediction phase of the algorithms is detailed. Finally, in section 3.5, we detail which cryptographic protocols we used, why, and how we implemented them, mentioning which toolkits were used.

3.1 Datasets Used

For running the experiments, we used datasets that are highly used in the literature, namely the Breast Cancer Wisconsin dataset[1], the Pima Indians Diabetes dataset[2], the Credit Approval dataset[3], and the Adult Income dataset [4]. Table 3.1 presents a brief description of each dataset.

[1] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
[2] https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes
[3] http://archive.ics.uci.edu/ml/datasets/credit+approval
[4] https://archive.ics.uci.edu/ml/datasets/adult

Table 3.1: Datasets used in Big dAta pRivacy by Design platform (BARD).

Dataset	**Subject**	**Instances**	**Features**
Breast Cancer Wisconsin	HealthCare	569	30
Pima Indians Diabetes	HealthCare	768	8
Credit Approval	Finance	690	15
Adult Income	Governance	48842	14

3.2 Data Preprocessing

Although our data is obtained from publicly available data sources, it is still required to do some preprocessing on the data. For example, the datasets contained missing values on some of the entries that needed to be addressed. Another example is the existence of categorical data that ML algorithms cannot process. We now describe some of the techniques we used in the data preparation phase for the datasets described in Table 3.1.

- **One-hot Encoding**[24] was initially used in digital circuits in order to determine the state of a state machine without using a decoder. The binary code is converted in a group of bits in which only one bit can have the value high (1), and all the others low (0). For example, the binary 110 is converted to 01000000. In ML, we apply this technique to deal with the problems created by using datasets with categorical (or nominal) data. Most ML algorithms require numerical representations in the data. A solution can be to assign an integer number to each different value present in the data, but this leads to the model assuming a natural ordering between categories that may not exist. Using one-hot encoding in categorical data allows us to circumvent this problem, creating additional binary variables for each unique value, *e.g.* if a variable describing pets has the values “dog”, “cat” and “fish”, after encoding three more variables will be added to the dataset, each representing a possible value. Then, a high (1) will be placed on the binary variable that represents the original value, and low (0) on all the others. Using the pets example, if the original value was “dog”, the resulting three binary variables

will be high on the binary variable representing "dog", and low on the other two ("dog" becomes $1, 0, 0$).

- **Feature Scaling** is a technique to normalize the range of data. For some ML algorithms, having broad range of values in one of the features may cause this feature to govern the modeling. There are different ways of achieving this, for example, *rescaling*, where the range of values is scaled to a target range (usually $[0, 1]$ or $[-1, 1]$), or *standardization*, where the features are rescaled so that they have the properties of a standard normal distribution, *i.e.* the mean average at zero and the standard deviation from the mean at one.

In our implementation, we used one-hot encoding in the Adult Income dataset and in the Credit Approval dataset. These two datasets contain categorical features that required the use of encoding to be processed by the ML algorithms.

In the Adult Income dataset, we have multiple examples of categorical features, namely the *workclass*, *education*, *marital-status*, *occupation*, *relationship*, *race*, *sex* and *native-country*. these were all encoded so that only numerical values were left. Of course, this caused a exponential growth on the number of features, going from the initial 14 to 108. In the Credit Approval dataset, we encoded all the non-numerical features, changing the number of features from 15 to 51. Feature scaling was used in all of the datasets, in order to reduce the errors caused by wide ranges. We used rescaling of all values to numbers between 0 and 1.

Besides the techniques mentioned above, it was also necessary to deal with the missing values that existed in the datasets. These values must be dealt with, since many ML algorithms cannot work with them. So, for the numerical features, we replaced those missing values with the median of the other existing values. In the case of the categorical features, the missing values are just replaced with zeros.

Finally, it was required for our setup a validation and a testing set, which some of the datasets did not contain originally. So we divided those datasets into training, validation and testing

sets, using a proportion of 70/15/15. When a testing set already existed, the validation set was created from the training set, so that it was of the same size as the testing set.

3.3 Baseline

The baseline approach consists on setting up ground values so that meaningful comparisons can be achieved. For understanding the overhead created by privacy-preserving technologies, we implemented the baseline using the publicly available ML toolkit, scikit-learn for Python[5].

In order to understand and explain what was done and why, we detail next the ML algorithms that were implemented in the baseline and in sections 3.4 and 3.5.

3.3.1 Decision Trees

Decision Tree (DT) are a decision support tool that uses a tree-like graph that represents decisions and possible outcomes of those decisions. It is an algorithm that is composed of conditional control sequences. Decision tree learning uses DT as a predictive model for classification. These DT are composed of nodes and leaves. The nodes represent decisions to take, or more specifically, thresholds that a feature is compared against, in order to decide which branch of the tree to follow. The leaves represent class labels.

To build a decision tree classifier there are a number of algorithms that can be used, each with different approaches and benefits. For example, ID3, C4.5, C5.0 and CART [42]. These algorithms focus in maximizing one or more metrics, such as Gini impurity or information gain.

The classification process of a sample in DT is an intuitive method. Each node of the tree has the information on which feature in the sample to compare against the threshold. So, classifying a sample in DT is done by traversing the tree from the top, comparing the values selected on each node with it's respective threshold, and depending on the result, choose one

[5] http://scikit-learn.org

child node or the other. When a leaf is reached, a class label is retrieved from the leaf and assigned to the sample, ending the classification process.

3.3.2 Support Vector Machines

Support Vector Machine (SVM) are supervised learning models that analyze data used for classification and regression analysis. An SVM model represents the examples as points in space, mapped so that the margin between the two classes for the data is as wide as possible. The vectors that define this margin, or *hyperplane*, are called support vectors. For classifying new samples, one maps them in that space, and predicts which class they belong based on which side of the gap they lie.

For calculating the SVM for linear classification, and in the case of a *hard-margin*, *i.e.*, when the training data is linearly separable, we select two parallel hyperplanes so that the distance between them is as large as possible. These hyperplanes can be described by the equations in 3.1.

$$\begin{aligned} \vec{w}.\vec{x} - b &= 1, \\ \vec{w}.\vec{x} - b &= -1 \end{aligned} \tag{3.1}$$

where $\vec{w}$ is the normal vector to each hyperplane, $\vec{x}$ is the training set and b is a scalar number. To maximize the distance between hyperplanes, we minimize the value of $||\vec{w}||$ subject to $y_i(\vec{w} \cdot \vec{x}_i - b) \geq 1, for i = 1, \ldots, n$, where the y_i are either 1 or -1, depending on the class label, and n is the number of samples in the training set.

In the case where the data is not linearly separable (*soft-margin*), we minimize instead the *hinge* loss function given by the equation in 3.2.

$$f(w, \lambda) = \left[\frac{1}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i(\vec{w} \cdot \vec{x}_i - b)\right) \right] + \lambda \|\vec{w}\|^2 \tag{3.2}$$

where the parameter λ determines the tradeoff between increasing the margin-size and ensuring that the $\vec{x}_i$ lie on the correct side of the margin.

For SVM non-linear classification, we employ a *kernel trick*, in which the dot product is replaced by a non-linear kernel function. The most used kernels are mentioned bellow.

- Linear: $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})$
- Polynomial: $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d$
- Gaussian radial basis function: $k(\vec{x_i}, \vec{x_j}) = \exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2)$

The classification of new samples in SVM is done using the scoring function in ref, where each testing sample x is attributed to a prediction label.

$$f(x) = \sum_{i=1}^{m} \alpha_i K(x_{SV}^{(i)}, x) + b \tag{3.3}$$

where α_i is the coefficient associated with the support vector $x_{SV}^{(i)}$, K is the kernel function chosen, and b is a scalar number.

3.3.3 *k*-Nearest Neighbors

k-Nearest Neighbors (k-NN) learning algorithm is based in the principles of pattern recognition. This algorithm is among the simplest of all ML algorithms, since it does not require a training step. To classify a new sample, we calculate the Euclidean Distance of it with the training samples, and discover the k samples that are close to it. Then, by majority vote of its k neighbors, the class label is assigned to the new sample.

3.3.4 *k*-Means

k-Means (k-M) clustering is a method commonly used to partition a dataset into k groups. It proceeds by selecting k initial cluster centers (centroids) and then iteratively refining them.

this refining is done in two distinct steps. 1) Each instance is assigned to its closest cluster. This is done by calculating the Euclidean Distance between each instance and each cluster center.Then, the lowest distance indicates which cluster the instance must be assigned to. 2) Each cluster center is updated to be the mean of all the instances assigned to it. The algorithm stops when the centroids no longer change position. This means that, depending on the data, it is not guaranteed that the optimal solution is found.

The classification of each testing sample is done similarly as the first step,*i.e.*, by computing the Euclidean Distance of the new sample with each centroid discovering the cluster whose centroid is closest to it. The label of the cluster becomes the predicted label of the sample.

3.3.5 Logistic Regression

Logistic Regression (LR) is a regression model in which the dependent variable is categorical. This variable is usually binary, *i.e.*, it can only take two values, usually 0 or 1 that represents outcomes such as "win/lose" or "healthy/sick". This binary logistic model is used to estimate the probability of a binary response based on one or more variables. To define LR, one must begin with the Logistic function, which in turn is given by equation 3.4.

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{3.4}$$

where t is the input. If we express t as $t = \beta_0 + \beta_1 x$, we can write the logistic function as in equation 3.5.

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{3.5}$$

acabar a seccao LR

To classify testing samples we use equation 3.6 to attribute a prediction label to each sample x.

$$f(x) = \beta_0 + \sum_{i=1}^{m} \beta_i x_i \tag{3.6}$$

3.4 Expanded Algorithms

Taking into account that the baseline implementation was done using a toolkit, we could not explicitly compare execution times, due to the fact that the operations in the toolkit were made in a "black box" mode. To solve this problem, we implemented the evaluation prediction processes of the ML algorithms without using the toolkit.

We successfully recreated the prediction processes by retrieving from the toolkit the specifications for each ML model, and we now enumerate them.

- For the DT, it was needed to retrieve all of the model, *i.e.*, the binary tree, so that we could traverse it according to each testing sample, comparing the feature with the threshold of each node, and choosing which branch of the tree to follow, until a label is reached.

- In the case of SVM, we retrieved from the toolkit all the coefficients needed to compute equation 3.3, namely the support vectors, the α coefficients for each support vector, the kernel function that was used(linear or polynomial), the exponent for the polynomial kernel if needed, and the scalar value b.

- For k-M, we just required from the toolkit the centroids for each cluster, as well as the prediction labels associated with each one. The classification of each testing sample was done by discovering which centroid was closest to it.

- In LR, we extracted the β values that appear in equation 3.6. β_0 is the intercept from the linear regression, and β_i are each regression coefficient that are multiplied by each feature of the sample.

In the case of k-NN, since training does not exist, there was no need to use the toolkit, so no information was retrieved from it, and it was not required to do a expanded version of the algorithm.

3.5 Cryptographic Domain

In the final step of our implementation, we made adjustments to the evaluation processes of the ML algorithms discussed above so that we could apply two privacy-preserving techniques, Garbled Circuits (GC) 2.4.5 and Homomorphic Encryption (HE) 2.4.6. These two techniques offer different means to obtain privacy-preserving computations, and we must consider them when choosing which ML algorithm to use with each cryptographic algorithm. GC builds ciphered Boolean circuits, which means that most of the computations are possible to implement on them. However, arithmetic computations require a large number of logic gates, creating a overhead that makes GC very slow. So, for some of the ML algorithms, we used a HE system, since it offers arithmetic operations as a core operation. Due to these constrains, we decided to adapt the DT and k-M to be evaluated using GC, and SVM and LR to be evaluated using HE.

3.5.1 Garbled Circuits and Decision Trees

The process of evaluating a DT in a privacy-preserving context is similar to evaluating in the usual manner, as described in 3.3.1. The main differences are the use of ciphered Boolean circuits instead of operations, *i.e.*, basic operations such as additions, comparisons, are replaced with logic gates, and the evaluation of the DT involves evaluating every single node in it, to disclose the least possible information due to observation of the computations.

In figure 3.1 we show how the computations are done inside each node of the DT. Each node contains the *featureID*, the ID of the feature to be selected from the sample to be classified, and the *threshold*, the value that is compared against the selected feature and that decides which branch of the tree to follow next. The first MUX gate selects from the sample the

feature to be compared. The GREATERTHAN gate compares the selected feature with the threshold. Then, the value from the comparison (0 or 1) is used as a selection bit in the second MUX to choose the *next_featureID* and *next_threshold* for the next node in the tree.

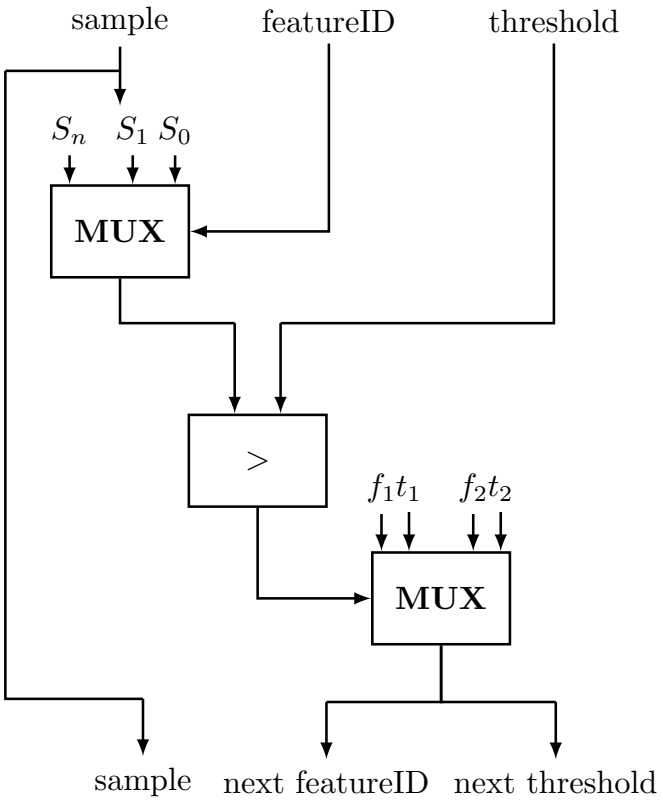
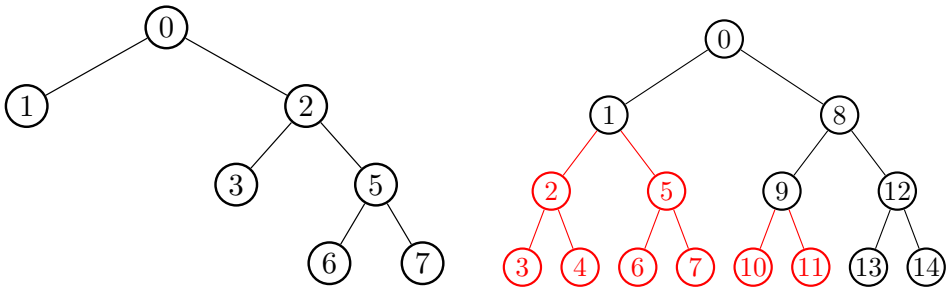


Figure 3.1: Diagram representing the Boolean circuit of each node in a DT.

It is also important to mention that the trees that we used are always complete, *i.e.* the number of nodes n is always the maximum possible, and it can be defined as $n = 2^{h+1} - 1$, where h is the *height* of the tree. We can see in figure 3.5.1 the implications of this expansion of the binary trees.



(a) Original DT. (b) Complete DT.

Figure 3.2: Expansion of binary trees.

In figure 3.2(a) we have a binary DT with $height = 3$ and with different path lengths to the leaves, depending on the branch of the tree taken, but on the tree in figure 3.2(b), all paths have the same length. With this, we can effectively hide the sparseness of the tree, which can leak relevant information about the original data. However, this solution increases exponentially the total amount of nodes that need to be evaluated, and that decreases performance significantly, as we will see in chapter 5.

3.5.2 Garbled Circuits and *k*-means

As in the previous section, evaluating the k-M algorithm in a privacy-preserving manner is similar to evaluating in the usual manner. We took the operations in the prediction step and transformed them in Boolean circuits, with logic gates such as multiplexers, adders, etc.

In figure 3.3 we show how we designed the circuit to represent the k-M prediction. The ED blocks represent the operations to perform the Euclidean distance between the sample and each centroid provided by the k-M model. The ARGMIN block represent the operations needed to choose, between all the distances calculated, which of it is the smallest.

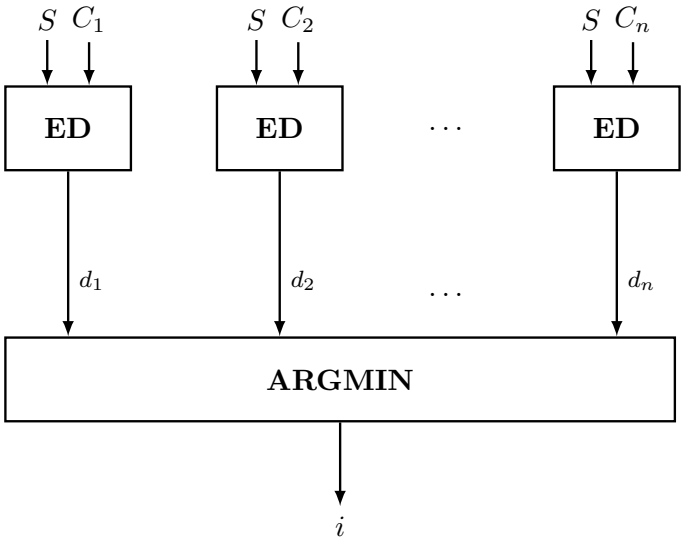


Figure 3.3: Circuit representing the prediction of the k-M algorithm.

As we show in ref(evaluation-k.means), there is an exponential decrease in performance when the number of clusters increases, or the sample has a larger number of features. This is caused by the multiplications that are done calculating the Euclidean distance. Usually this is reason to use HE instead, but the algorithm to implement comparisons is hard to implement [3].

is this justification enough?

3.5.3 Homomorphic Encryption and Support Vector Machines

Estas duas secções devem estar descritas como trabalho feito? (foi implementado pelo José Portelo) Ou ficam para future work? ou só mencionar noutro local da tese?

3.5.4 Homomorphic Encryption and Logistic Regression

3.6 Use Case: Healthcare

As mentioned in section 2.6, healthcare systems generate vast amounts of data every day. Processing this data can be beneficial for both the health institution (hospitals, clinics) and

for the patients. However, usage of these Eletronic Medical Records (EMR) cannot be freely done by institutions without consent of the patients and compliance with the privacy laws. As a result, this processing exists only *in-house*, with only a few exceptions[6]. The problem is that, developing and/or maintaining a Data Mining infrastructure in an institution amass costs that can it may not be willing to spare.

Our contribution to mitigate this standoff between the gains and costs of Data Mining EMR, is to be able to provide a product that would remove the costs of maintenance and development from the institutions, while at the same time provide enough privacy guarantees that comply with the law.

We now describe a typical use case scenario for privacy-preserving processing of EMR.

- **Description:** Design and implementation of a platform to process EMR in order to improve treatments and diagnoses, while maintaining identities private. This is achieved by training models using this data and then predict medical conditions for future patients. All the computations should be done resorting to privacy-preserving techniques.

- **Actors involved:** Healthcare institutions, patients, medical staff.

- **Preconditions:** Access to data and to EMR of patients. Consent from each patient regarding the processing of his/her data.

- **Basic Flow:**

    1. The institution supplies the platform with data to train the models for one or more ML algorithms. This training must be done in an encrypted and/or anonymised domain.
    2. A new patient arrives at the institution and it is asked of him if he/she consents to the use of the platform to speedup his/her diagnosis. The patient agrees.

[6] https://www.reuters.com/article/us-health-medicalrecords-sharing/few-u-s-hospitals-can-fully-share-electronic-medical-records-idUSKCN1C72UV

3. Data of the patient is collected by a person from the medical staff, including his/her symptoms, medical history, etc.
4. This data is supplied to the platform, and the platform performs one or more predictions, depending on the number of models the platform has, using privacy-preserving techniques to do so.
5. The platform informs the doctor of what are the results of the predictions.
6. The doctor decides on the appropriate medical action, taking into account his medical background and the information supplied by the platform.

- **PostConditions:** The platform has received data from the institution. The platform has trained different instantiations of ML algorithms.

3.7 Summary

Summary chapter 3.

4 BARD

In this chapter we present the project from Altran that this thesis is a part of.

Expand this introduction

4.1 Architecture

This section presents the architectural specifications of Big dAta pRivacy by Design platform (BARD).

Expand

4.1.1 Internal Structure

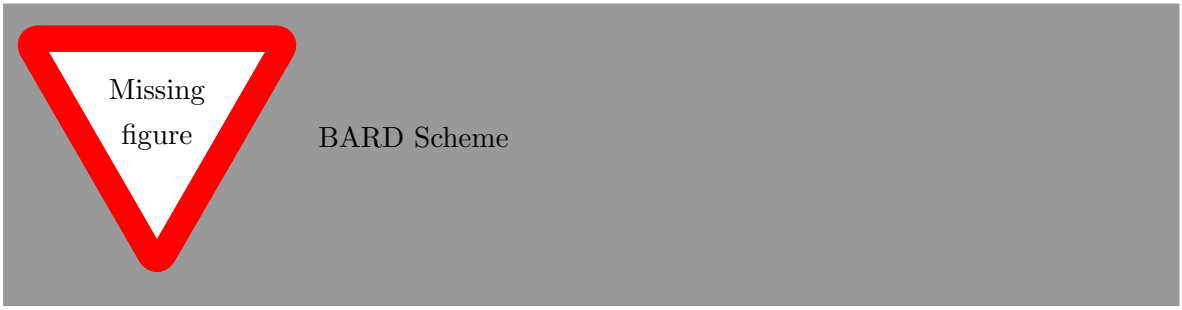
add a smallish intro to this section

BARD is composed of:

- A dataset to train the Machine Learning (ML) algorithm, or the values representing the already trained algorithm.
- A sample or a set of samples that represent the input of the "user", to be predicted.
- A prediction algorithm that depends on the ML algorithm and the privacy-preserving technique chosen.
- A set of toolkits for each of the techniques used.

more?

- ...



BARD Scheme

fazer um esquema para o BARD. não só de arquitectura, mas como as pessoas interagem com o BARD.

passos no flow geral de processamento de dados : recolha de dados, pre-processamento, etc

4.2 BARD API

4.3 Summary

Summary chapter 4.

5 Evaluation

In this chapter we describe the experiments that were conducted regarding the implementation of Machine Learning (ML) algorithms using privacy-preserving techniques. In section 5.1, we present the metrics used in the experiments.

5.1 Evaluation Metrics

To evaluate our implementation, it is important to understand the metrics that were considered. To define the evaluation functions bellow, we present the notation in table 5.1.

Table 5.1: Notation.

Real label	**Predicted label**	**Event**
+1	+1	True Positive (TP)
+1	-1	False Negative (FN)
-1	+1	False Positive (FP)
-1	-1	True Negative (TN)

Accuracy is defined as how much of the measurements of a value differ to the real value. In our implementation, it represents how many times the predictions calculated by the ML generated models match the class of the testing samples. In mathematical terms, it is represented by:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Precision is defined by the fraction of relevant instances among the retrieved instances 5.2.

Recall is defined by the fraction of relevant instances that have been retrieved over the total of the relevant instances 5.3.

$$precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$recall = \frac{TP}{TP + FN} \tag{5.3}$$

F-measure is a measure of a accuracy of a test. It considers both the precision and the recall of the test to compute the score:

$$F_1 = 2.\frac{precision.recall}{precision + recall} \tag{5.4}$$

A *confusion matrix* is a specific table layout that allows visualization of the performance of a ML algorithm. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. In 5.1 we give an example on how to compute a confusion matrix for the binary case.

$$confusion\_matrix = \begin{array}{c} \\ +1 \\ -1 \end{array} \begin{array}{c} \begin{array}{cc} +1 & -1 \end{array} \\ \begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix} \end{array}$$

Besides these metrics, we also used additional ones in order to understand how much the computational overhead due to the use of cryptography influences the system. We compared the results obtained by the privacy-preserving versions of ML algorithms with the ones obtained using the baseline.S We also take into account the execution times of the system, which shows the overhead caused by the additional computational cost added by cryptography. Finally, we also show the increase in communication costs that happen when cryptography is involved, as all values must be represented by ciphertext, instead of integer or float values.

5.2 Experimental Setup

All the experiments were performed using a machine with an Inter Core i5-4300M CPU @2.60Gz with a 3MB L3 cache memory and a 12 GB RAM memory.

For obtaining the experimental results, we started by applying the pre-processing techniques mentioning in 3.2 in the datasets described in table 3.1. As mentioned, all datasets that were composed by a single file were split into three sets, training, validation and testing sets, with the proportion 70/15/15. Each ML model was trained using the training set, the best model configuration was chosen using the validation set, and the model performance was evaluated using the testing set.

5.2.1 Parameters - Baseline

For the experiments with Decision Tree (DT), we tested the values for *max_depth* of 5%, 10%, 20%, 50%, 100%, 200%, and 500% of the total number of features, and the values for *min_samples_leaf* of 0.001%, 0.002%, 0.005%, 0.01%, 0.02%, 0.05%, 0.1%, 0.2%, 0.5%, 1%, 2% and 5% of the total number of training samples.

For the experiments with Support Vector Machine (SVM), we used *kernel* values of *linear*, *poly* and *rbf*. For all kernels, we used C values of 2^{-10}, 2^{-6}, 2^{-2}, 2^{2}, 2^{6} and 2^{10}. For the polynomial kernel, we used *degree* values of 2, 3 and 4. For the radial basis function (*rbf*) kernel, we used γ values of 2^{-9}, 2^{-5}, 2^{-1}, 2^{1} and 2^{3}.

For the experiments with k-Means (k-M), we tested with a variable number of clusters, *i.e.*, with *num_clusters* values of 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100.

For the experiments with Logistic Regression (LR), we used the *liblinear* solver with C values of 2^{-10}, 2^{-6}, 2^{-2}, 2^{2}, 2^{6} and 2^{10}.

These variations on parameters allowed to train models with all possible configurations without the need to specifically adapt the parameters to the different datasets.

5.2.2 Parameters - Garbled Circuits

In the experiments using Garbled Circuits (GC), we tested values of 8, 12, 16, 20 and 24 bits for the numeric precision of the data and model parameters. This will be reflected in the circuit size and the accuracy of the results. Larger values were not considered because 24 bits is already sufficient for an exact representation of the input values and model parameters.

5.2.3 GC toolkits

For the experiments with GC, we used the toolkit developed by VIPP group from the University of Sienahttp://clem.dii.unisi.it/~vipp/index.php/home. This toolkit was not our first choice, since it has known issues in computation times, but the other toolkits that we tested contained limitations that we could not overcome, as stated below:

- **ABY[12]:** We found it impossible to define gate-to-gate wires, and that removed the ability for fine control on how to use and combine wires.

- **JustGarble**https://github.com/irdan/justGarble**:** This toolkit could not be fully compiled due to conflicts with current versions of the GNU gcc compiler.

- **Ciphermed[6]:** This toolkit is efficient for small DT, but is exponentially slower for larger trees (above 10 nodes).

- **TinyGarble[41]:** The current version of this toolkit does not support the open source synthesis tool (Yosis http://www.clifford.at/yosys/) recommended by the authors, and only supports a paid one.

- **CompGC[20]:** The implementation of all the examples of ML algorithms in this toolkit are hardcoded, making it extremely difficult to adapt to our needs.

5.3 Experimental Results - Baseline

We now show the experimental results obtained by applying different ML algorithms to the datasets mentioned above. We show in each subsection below, corresponding to each dataset, the different ML algorithms and parameters, and mention results found in the literature. Each result that we show in the following tables are the ones obtained by the best combination of parameters, *i.e.*, the parameters that provided the best accuracy or F-Measure results with the validation set.

5.3.1 Breast Cancer Wisconsin Dataset

We present the best baseline results obtained in the testing set for the Breast Cancer Wisconsin Dataset in table 5.2.

Table 5.2: Baseline results for Breast Cancer Wisconsin Dataset. "A" represents Accuracy, "F" represents F-Measure.

ML algorithm	DT	k-Means	LR	SVM		
				Linear	Poly	RBF
Baseline	A: 92.94% F: 90.91%	A: 91.76% F: 90.91%	A: 95.29% F: 93.75%	A: 94.12% F: 92.06%	A: 94.12% F: 92.31%	A: 94.12% F: 92.06%
Literature	A: 95.13% F: 94.88%	A: 92.79% F: -	A: 93.50% F: -	A: - F: -	A: 97.54% F: -	A: 97.13% F: 96.25%

As we see, our baseline results are near the ones found in the literature.

5.3.2 Pima Indians Diabetes Dataset

We present the best baseline results obtained in the testing set for the Pima Indians Diabetes Dataset in table 5.3.

Table 5.3: Baseline results for Pima Indians Diabetes Dataset. "A" represents Accuracy, "F" represents F-Measure.

ML algorithm	DT	k-Means	LR	SVM		
				Linear	Poly	RBF
Baseline	A: 73.04% F: 63.53%	A: 72.17% F: 52.94%	A: 75.65% F: 58.82%	A: 75.65% F: 61.11%	A: 76.52% F: 59.70%	A: 77.39% F: 62.86%
Literature	A: 75.39% F: -	A: 73.7 F: -	A: 77.95% F: -	A: - F: -	A: -% F: -	A: 80.2% F: -

We can see that our results are comparable to the ones found in the literature.

5.3.3 Credit Approval Dataset

We present the best baseline results obtained in the testing set for the Credit Approval Dataset in table 5.4.

Table 5.4: Baseline results for Credit Approval Dataset. "A" represents Accuracy, "F" represents F-Measure.

ML algorithm	DT	k-Means	LR	SVM		
				Linear	Poly	RBF
Baseline	A: 78.64% F: 75.00%	A: 83.50% F: 81.32%	A: 85.44% F: 84.21%	A: 85.44% F: 84.54%	A: 85.43% F: 83.87%	A: 84.47% F: 83.33%
Literature	A: 85.5% F: -	A: 86.3 F: -	A: 87.9% F: -	A: 86.2 F: -	A: 84.8% F: -	A: 85.5% F: -

We obtained results that are similar to the ones found in the literature.

5.3.4 Adult Income Dataset

We present the best baseline results obtained in the testing set for the Adult Income Dataset in table 5.5.

Table 5.5: Baseline results for Adult Income Dataset. "A" represents Accuracy, "F" represents F-Measure.

ML algorithm	DT	k-Means	LR	SVM		
				Linear	Poly	RBF
Baseline	A: 85.56% F: 67.37%	A: 81.95% F: 55.80%	A: 85.08% F: 66.87%	A: 69.67% F: 57.04%	A: 80.82% F: 65.91%	A: 82.79% F: 61.15%
Literature	A: 82.20% F: -	A: - F: -	A: 80.00% F: -	A: - F: -	A: 84.55% F: -	A: 84.93% F: -

5.4 Experimental Results - Garbled Circuits

In this section, we present the results of using the toolkit mentioned above, VIPP, to create a privacy-preserving prediction for all the datasets mentioned in this thesis. Firstly we will compare the results we obtained with the baseline, and afterwards we present the execution times using the toolkit for each of the ML algorithms.

Missing the references for literature in these tables. look for better results.

5.4.1 Comparison with the Baseline

It is important to mention that, after analyzing the results obtained using the toolkit, we verified that changing the amount of bits for the actual numeric precision of the data and model parameters affects the accuracy of the results. The degree of this error is depicted in table 5.6 and 5.7, for the experiments for DT and k-M respectively. It is to be noted that this error is computed versus the baseline prediction results, not the prediction labels from the dataset.

Table 5.6: Average prediction error for DT when compared with the baseline.

bits	Pima Indians	Breast Cancer	Credit Approval	Adult Income
8	1.88%	0.55%	8.70%	0.00%
12	0.00%	0.13%	1.11%	0.00%
16	0.00%	0.13%	0.31%	0.00%
20	0.00%	0.13%	0.31%	0.00%
24	0.00%	0.13%	0.31%	0.00%

Table 5.7: Average prediction error for k-M when compared with the baseline.

bits	Pima Indians	Breast Cancer	Credit Approval	Adult Income
8	2.03%	3.07%	0.05%	0.02%
12	0.39%	0.85%	0.00%	0.00%
16	0.29%	0.72%	0.00%	0.00%
20	0.29%	0.72%	0.00%	0.00%
24	0.00%	0.00%	0.00%	0.00%

By observing these tables, we can conclude that the loss of prediction performance caused by using the privacy-preserving versions of the ML algorithms is not relevant, as long as at least 16 bits are used to represent the data. Since both DT and k-M only output an integer representing the label, and not a real number, the visible effect of changing the amount of bits is minimal.

5.4.2 Execution Times - Decision Tree

We present the execution times obtained in using the toolkit to build a GC implementation of DT for all the datasets in the tables below. The results are presented in terms of average pre-computation times per data sample and runtimes per data sample. Table 5.8 presents the average pre-computation times for each data sample for all datasets.

Table 5.8: Average pre-computation times per data sample, in seconds.

dataset	**numeric precision**				
	8 bits	**12 bits**	**16 bits**	**20 bits**	**24 bits**
Breast	0.205	0.240	0.281	0.325	0.356
Pima	0.219	0.285	0.310	0.344	0.356
Credit	0.224	0.253	0.271	0.290	0.315
Adult	0.233	0.271	0.313	0.355	0.373

We can observe that average pre-computation times are very similar to one another despite the size of the GC circuits, that are represented by the numeric precision, meaning that this poses no restrictions regarding the scalability of our solution.

Table 5.9: Runtime per data sample, in seconds. Breast Cancer Wisconsin Diagnostic dataset.

DT depth	numeric precision				
	8 bits	12 bits	16 bits	20 bits	24 bits
1	1.015	1.383	1.735	2.082	2.442
3	1.057	1.425	1.804	2.187	2.550
4	1.107	1.533	1.920	2.319	2.724
5	1.307	1.713	2.122	2.665	2.974
6	1.538	2.030	2.522	2.939	3.330
7	1.966	2.393	2.823	3.507	3.787

Table 5.10: Runtime per data sample, in seconds. Pima Indians Diabetes dataset.

DT depth	numeric precision				
	8 bits	12 bits	16 bits	20 bits	24 bits
1	0.374	0.502	0.663	0.782	0.901
4	0.457	0.548	0.677	0.888	1.087
6	0.654	0.877	1.051	1.177	1.195
8	1.622	1.689	1.889	2.099	2.180
10	3.469	3.644	3.112	4.300	4.343
12	7.884	9.727	12.459	16.270	17.488
13	16.289	18.353	20.926	25.120	33.508

Table 5.11: Runtime per data sample, in seconds. Credit Approval dataset.

DT depth	numeric precision				
	8 bits	12 bits	16 bits	20 bits	24 bits
2	1.598	2.333	2.956	3.591	4.190
5	1.852	2.603	3.295	3.953	4.576
7	2.702	3.600	4.359	5.743	5.848
8	3.365	4.280	6.541	7.337	8.998
9	4.511	6.930	8.202	10.990	12.227

Table 5.12: Runtime per data sample, in seconds. Adult Income dataset.

DT depth	numeric precision				
	8 bits	12 bits	16 bits	20 bits	24 bits
5	3.777	5.169	6.502	8.171	9.424
6	4.111	5.774	7.074	8.813	10.816
9	9.256	13.864	20.294	24.314	27.236
10	17.167	23.056	30.553	39.114	54.987

Finish analysis of tables and add graphs.

5.4.3 Execution Times - *k*-Means

We present the execution times obtained in using the toolkit to build a GC implementation of k-M for all the datasets in the tables below. The results are presented in terms of average pre-computation times per data sample and runtimes per data sample. Table 5.13 presents the average pre-computation times for each data sample for all datasets.

Table 5.13: Average pre-computation times per data sample, in seconds.

dataset	**numeric precision**				
	8 bits	**12 bits**	**16 bits**	**20 bits**	**24 bits**
Breast	0.225	0.251	0.274	0.289	0.301
Pima	0.260	0.283	0.307	0.331	0.339
Credit	0.226	0.249	0.253	0.265	0.272
Adult	0.214	0.214	0.232	0.266	0.262

We can see that average pre-computation times are all very similar to one another despite the slight dependence on the GC size, meaning that it does not impact the scalability of our solution.

Table 5.14: Runtime per data sample, in seconds. Breast Cancer Wisconsin Diagnostic dataset.

number clusters	**numeric precision**				
	8 bits	**12 bits**	**16 bits**	**20 bits**	**24 bits**
2	1.619	2.102	2.788	3.616	3.838
3	1.868	2.451	3.037	3.550	5.674
4	1.800	3.082	3.180	5.028	5.604
5	1.912	2.598	4.760	5.410	7.402
6	2.062	2.707	4.872	5.275	7.951
7	2.351	2.882	5.133	7.149	7.906
8	2.930	2.805	4.978	6.946	9.124
9	2.570	3.917	4.935	7.044	9.136
10	2.291	4.215	5.990	7.349	10.802
20	4.223	6.797	9.580	14.877	25.232
30	5.269	9.234	15.693	24.228	36.767
40	7.126	14.724	25.754	37.984	49.574
50	7.885	14.701	24.245	40.602	67.651
60	9.156	19.862	35.970	56.269	-
70	11.585	23.550	41.306	-	-
80	11.433	23.367	45.067	-	-
90	13.642	27.030	53.124	-	-
100	16.684	32.046	59.372	-	-

Table 5.15: Runtime per data sample, in seconds. Pima Indians Diabetes dataset.

number clusters	**numeric precision**				
	8 bits	**12 bits**	**16 bits**	**20 bits**	**24 bits**
2	0.629	0.848	1.163	1.350	1.533
3	0.813	1.055	1.270	1.494	1.761
4	0.852	1.143	1.429	1.664	1.969
5	0.943	1.246	1.531	1.824	2.066
6	1.140	1.261	1.513	2.266	2.145
7	1.124	1.319	1.620	2.319	2.188
8	1.178	1.339	1.697	2.130	2.337
9	1.227	1.464	2.356	2.178	2.504
10	1.228	1.544	2.439	2.497	3.717
20	1.568	1.896	3.374	3.952	5.751
30	2.142	2.223	3.794	5.315	7.346
40	1.762	3.379	5.099	6.941	9.697
50	1.803	3.670	5.979	8.300	12.376
60	3.005	4.927	7.025	12.218	16.839
70	3.420	5.250	7.616	15.795	19.339
80	3.513	6.187	8.649	15.795	21.302
90	3.490	6.239	12.278	16.691	23.669
100	3.580	7.876	14.497	23.599	32.248

Table 5.16: Runtime per data sample, in seconds. Credit Approval dataset.

number clusters	**numeric precision**				
	8 bits	**12 bits**	**16 bits**	**20 bits**	**24 bits**
2	2.541	3.652	4.151	5.729	6.837
3	2.615	3.752	5.547	6.369	8.512
4	2.902	3.663	5.934	8.114	9.692
5	3.016	5.036	5.825	8.633	10.725
6	3.129	5.359	7.474	9.820	12.961
7	3.058	5.216	7.463	10.031	16.429
8	3.033	5.360	7.912	11.342	15.633
9	3.082	5.326	8.911	13.911	19.112
10	4.196	6.569	9.412	14.731	18.997
20	6.368	10.641	19.121	28.797	44.666
30	9.060	17.660	28.409	49.931	-
40	11.152	25.673	42.835	-	-
50	14.681	27.539	54.795	-	-
60	18.326	37.088	-	-	-
70	19.318	41.164	-	-	-
80	23.371	56.422	-	-	-
90	27.361	62.862	-	-	-
100	27.908	67.835	-	-	-

Table 5.17: Runtime per data sample, in seconds. Adult Income dataset.

number clusters	numeric precision				
	8 bits	**12 bits**	**16 bits**	**20 bits**	**24 bits**
2	4.703	5.536	8.098	11.307	13.672
3	4.361	7.535	9.620	14.500	18.896
4	4.396	6.986	10.364	15.635	20.343
5	6.083	8.688	11.681	17.928	25.292
6	5.587	11.307	15.884	23.006	31.464
7	6.441	10.278	15.981	24.306	31.277
8	6.394	10.871	17.157	27.155	38.732
9	8.029	12.958	21.050	32.607	48.497
10	8.706	14.956	24.468	32.010	47.241
20	14.834	24.567	41.386	-	-
30	19.150	39.870	-	-	-
40	27.248	60.023	-	-	-
50	33.243	-	-	-	-
60	42.933	-	-	-	-
70	52.514	-	-	-	-
80	61.952	-	-	-	-
90	-	-	-	-	-
100	-	-	-	-	-

5.5 Summary

6 Conclusions and Future Work

6.1 Conclusions

6.2 Future Work

Todo list

Bibliography

[1] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450. ACM, 2000.

[2] Ross Anderson. *Security engineering*. John Wiley & Sons, 2008.

[3] Ian F Blake and Vladimir Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 515–529. Springer, 2004.

[4] Dan Bogdanov, Liina Kamm, Baldur Kubo, Reimo Rebane, Ville Sokk, and Riivo Talviste. Students and taxes: a privacy-preserving study using secure computation. *Proceedings on Privacy Enhancing Technologies*, 2016(3):117–135, 2016.

[5] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.

[6] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS*, volume 4324, page 4325, 2015.

[7] Ljiljana Brankovic and Vladimir Estivill-Castro. Privacy issues in knowledge discovery and data mining. In *Australian institute of computer ethics conference*, pages 89–99, 1999.

[8] Justin Brickell and Vitaly Shmatikov. Privacy-preserving classifier learning. In *International Conference on Financial Cryptography and Data Security*, pages 128–147. Springer, 2009.

[9] Giuseppe D'Acquisto, Josep Domingo-Ferrer, Panayiotis Kikiras, Vicenç Torra, Yves-Alexandre de Montjoye, and Athena Bourka. Privacy by design in big data: An overview of privacy enhancing technologies in the era of big data analytics. *European Union Agency for Network and Information Security*, 2015.

[10] George Danezis, Josep Domingo-Ferrer, Marit Hansen, Jaap-Henk Hoepman, Daniel Le Metayer, Rodica Tirtea, and Stefan Schiffner. Privacy and data protection by design-from policy to engineering. *European Union Agency for Network and Information Security*, 2015.

[11] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3: 1376, 2013.

[12] Daniel Demmler, Thomas Schneider, and Michael Zohner. Aby-a framework for efficient mixed-protocol secure two-party computation. In *NDSS*, 2015.

[13] Mahir Can Doganay, Thomas B Pedersen, Yücel Saygin, Erkay Savaş, and Albert Levi. Distributed privacy preserving k-means clustering with additive secret sharing. In *Proceedings of the 2008 international workshop on Privacy and anonymity in information society*, pages 3–11. ACM, 2008.

[14] Cynthia Dwork. Differential privacy. *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming*, pages 1–12, 2006. ISSN 03029743. doi: 10.1007/11787006_1.

[15] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[16] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4):469–472, 1985.

[17] Craig Gentry et al. Fully homomorphic encryption using ideal lattices. In *Symposium on the Theory of Computing*, volume 9, pages 169–178, 2009.

[18] Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, pages 86–97, 1998.

[19] Shafi Goldwasser. Multi party computations: past and present. In *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 1–6. ACM, 1997.

[20] Adam Groce, Alex Ledger, Alex J Malozemoff, and Arkady Yerukhimovich. Compgc: Efficient offline/online semi-honest two-party computation. *IACR Cryptology ePrint Archive*, 2016:458, 2016.

[21] Munawar Hafiz, Paul Adamczyk, and Ralph E Johnson. Organizing security patterns. *IEEE software*, 24(4), 2007.

[22] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[23] Marit Hansen, Meiko Jensen, and Martin Rost. Protection goals for privacy engineering. *Proceedings - IEEE Security and Privacy Workshops, SPW*, pages 159–166, 2015. doi: 10.1109/SPW.2015.13.

[24] David Harris and Sarah Harris. *Digital design and computer architecture*. Morgan Kaufmann, 2010.

[25] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. In *Advances in Cryptology—EUROCRYPT 2000*, pages 539–556. Springer, 2000.

[26] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free xor gates and applications. *Automata, Languages and Programming*, pages 486–498, 2008.

[27] Lei Xu, Chunxiao Jiang, Jian Wang, Jian Yuan, Yong Ren, Lei Xu, Chunxiao Jiang, Jian Wang, Jian Yuan, and Yong Ren. Information Security in Big Data: Privacy and Data Mining. *IEEE Access*, 2:1149–1176, 2014. ISSN 2169-3536.

[28] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering. ICDE. IEEE 23rd International Conference on*, pages 106–115. IEEE, 2007.

[29] Rongxing Lu, Hui Zhu, Ximeng Liu, Joseph Liu, and Jun Shao. Toward efficient and privacy-preserving computing in big data era. *IEEE Network*, 28(4):46–50, 2014. ISSN 08908044. doi: 10.1109/MNET.2014.6863131.

[30] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[31] Moni Naor and Benny Pinkas. Computationally secure oblivious transfer. *Journal of Cryptology*, 18(1):1–35, 2005.

[32] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[33] Benny Pinkas, Thomas Schneider, Nigel P Smart, and Stephen C Williams. Secure two-party computation is practical. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 250–267. Springer, 2009.

[34] Michael O Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[35] Balaji Raghunathan. *The Complete Book of Data Anonymization: From Planning to Implementation*. CRC Press, 2013.

[36] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[37] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[38] Paul M Schwartz and Daniel J Solove. The pii problem: Privacy and a new concept of personally identifiable information. *NYUL rev.*, 86:1814, 2011.

[39] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[40] Mark Shaneck, Yongdae Kim, and Vipin Kumar. Privacy preserving nearest neighbor search. In *Data Mining Workshops. ICDM Workshops. Sixth IEEE International Conference on*, pages 541–545. IEEE, 2006.

[41] Ebrahim M Songhori, Siam U Hussain, Ahmad-Reza Sadeghi, Thomas Schneider, and Farinaz Koushanfar. Tinygarble: Highly compressed and scalable sequential garbled circuits. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 411–428. IEEE, 2015.

[42] Carolin Strobl, James Malley, and Gerhard Tutz. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, 14(4):323, 2009.

[43] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[44] Jaideep Vaidya, Basit Shafiq, Anirban Basu, and Yuan Hong. Differentially private naive bayes classification. In *Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)-Volume 01*, pages 571–576. IEEE Computer Society, 2013.

[45] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[46] Rüdiger Wirth and Jochen Hipp. Crisp-dm: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, pages 29–39. Citeseer, 2000.

[47] Andrew C Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*, pages 160–164. IEEE, 1982.

[48] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167. IEEE, 1986.

[49] Hwanjo Yu, Jaideep Vaidya, and Xiaoqian Jiang. Privacy-preserving svm classification on vertically partitioned data. *Advances in Knowledge Discovery and Data Mining*, pages 647–656, 2006.