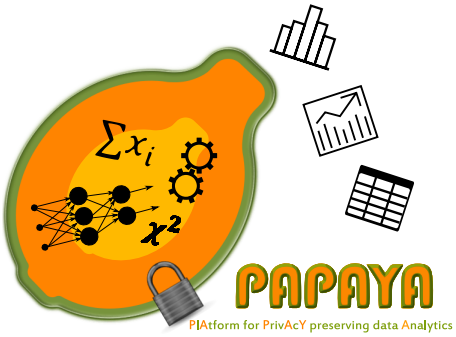


D3.1 - Preliminary Design of Privacy Preserving Data Analytics

Work Package	WP 3, Privacy Enhancing Technologies for Data Analytics
Lead Author	Beyza BOZDEMIR, Orhan ERMIS, Melek Önen (EURC)
Contributing Author(s)	Beyza BOZDEMIR, Orhan ERMIS, Melek ÖNEN (EURC), Muhammad BARHAM, Boris ROZENBERG, Ron SHMELKIN (IBM), Monir Azraoui, Sébastien CANARD, Bastien VIALLA (ORA)
Reviewers	Eleonora Ciceri (MCI), Angel Palomares Perez (ATOS)
Due Date	30.04.2019
Delivery	30.04.2019
Version	1.0
Dissemination Level	Public

The research leading to these results has received funding from the European Unions Horizon 2020 Research and Innovation Programme, through the PAPAYA project, under Grant Agreement No. 786767. The content and results of this deliverable reflect the view of the consortium only. The Research Executive Agency is not responsible for any use that may be made of the information it contains.

Revision History

Revision	Date	Author	Description
0.1	14.01.2019	Orhan ERMIS (EURC)	ToC Template
0.2	16.01.2019	Orhan ERMIS (EURC)	Updated ToC Template
0.3	25.03.2019	Beyza BOZDEMIR, Orhan ERMIS, Melek ÖNEN, Muhammad BARHAM, Boris ROZENBERG, Ron SHMELKIN, Monir AZRAOUI, Sébastien CANARD, Bastien VIALLA (EURC, IBM, ORA)	First version complete for internal review
0.4	02.04.2019	Beyza BOZDEMIR, Orhan ERMIS, Melek ÖNEN, Muhammad BARHAM, Boris ROZENBERG, Ron SHMELKIN, Monir AZRAOUI, Sébastien CANARD, Bastien VIALLA (EURC, IBM, ORA)	Updates following partners' feedback
0.5	15.04.2019	Beyza BOZDEMIR, Orhan ERMIS, Melek ÖNEN, Muhammad BARHAM, Boris ROZENBERG, Ron SHMELKIN, Monir AZRAOUI, Sébastien CANARD, Bastien VIALLA (EURC, IBM, ORA)	Updates following reviewers' comments
0.6	21.04.2019	Beyza BOZDEMIR, Melek ÖNEN, Ron SHMELKIN, Monir AZRAOUI, Sébastien CANARD (EURC, IBM, ORA)	Final updates according to latest comments and quality check
1.0	30.04.2019	Beyza BOZDEMIR, Orhan ERMIS, Melek ÖNEN, Ron SHMELKIN, Monir AZRAOUI, Sébastien CANARD (EURC, IBM, ORA)	Final updates, ready for submission

Executive Summary

This report overviews the preliminary design of new privacy preserving (PP) primitives for three main data analytics operations, namely: neural network classification and training, clustering and counting. The document focuses on these three particular algorithms as these are mainly derived from the definition of the PAPAYA use cases in Deliverable D2.1. All these newly proposed solutions aim at enabling an untrusted third-party data processor (the PAPAYA platform, in this case) to perform the underlying operations over protected data. Thanks to these primitives, data owners will be able to extract valuable information from their protected data while being cost-effective and accurate. This report first reviews the existing cryptographic tools including homomorphic encryption, secure multi-party computation, functional encryption and differential privacy, which are used as building blocks for the design of the newly proposed PAPAYA primitives. Then, the PAPAYA solutions are further described under the following three categories:

- **Privacy preserving neural networks (PP-NN)** that consist of allowing a data owner either to classify or to collaboratively (with other data owners) train neural networks (NN) while ensuring data privacy. Existing studies regarding the use of NN together with the new PP primitives are first overviewed. In the particular case of neural network classification, the preliminary designs of three solutions are presented: two based on homomorphic encryption (one of them involves fully homomorphic encryption, and the other involves partially homomorphic encryption) and another one based on two-party computation which has a particular focus on electro-cardiogram data processed by one of the PAPAYA use cases defined in deliverable D2.1. Furthermore, the preliminary design of a privacy preserving collaborative training algorithm based on differential privacy is described and shown to be robust against existing attacks on collaborative training.

- **Privacy preserving clustering** that allows data owners to group similar data objects in the same group in a privacy preserving manner. A concise summary on the well-known data clustering techniques is first provided. Some of the important PP clustering approaches in the literature are also introduced. Later, the problem of privacy preserving trajectory clustering is reminded as finding similar traveling routes in a set of trajectories. Finally, a preliminary version of a privacy preserving clustering solution based on the additively homomorphic Paillier encryption scheme is briefly presented.

- **Privacy preserving counting** that allows data owners to execute basic statistics such as counting, set union and set intersection on the protected data. Some background information about the PP primitives to be used for PP basic statistics are presented: Namely, functional encryption and Bloom filters. Then, for the particular operation of counting, a preliminary design of privacy preserving counting solution based on the use of functional encryption is proposeed. Moreover, An initial version of set interaction & set union operations based on encrypted Bloom filters is also described.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Purpose and Scope

With the recent advances in information technology, organisations are able to apply data analytics techniques over the data collected from clients' smart devices and derive valuable information and improve their place in the market. Nevertheless, the data being analyzed and processed usually contains sensitive information (such as health information or business confidential information) that may endanger clients' privacy. With the European General Data Protection Regulation (GDPR) [85] and the ePrivacy Regulation [31], there is a strong need for technological means to protect individuals' privacy, and to extract meaningful and useful information out of the collected data, at the same time. With this aim, the goal of this work package (WP3) is to design, develop and evaluate new Privacy Preserving (PP) data analytics modules to infer valuable insights and improve predictions by employing advanced cryptographic techniques such as homomorphic encryption, secure multi-party computation, functional encryption and differential privacy.

Homomorphic encryption (HE) schemes have been often employed for the privacy preserving data analytics solutions in the recent years since it is possible to make meaningful computations on the encrypted data without revealing the content of that data. In general, HE schemes are categorized as partially homomorphic encryption (PHE) [45] and fully homomorphic encryption schemes (FHE) [47]. In the first one, only a single group of operations (addition or multiplication) is supported, whereas in the second one, there is no such restriction and the supported operations varies depending on the data analytics.

Another important cryptographic technique to be utilized in the scope of the document is the secure multiparty computation, in particular secure Multiparty Computation (MPC). This technique enables parties to compute a common function that takes inputs from all parties without revealing these inputs to each other.

Functional encryption [8] is also one of the most important cryptographic techniques suitable to the privacy preserving data analytics. The idea behind this technique is that a party shares a functional decryption key with other authorized parties in order to compute the result of some function without revealing the output.

The last privacy enhancing technology suitable to the protection of PAPAYA data is differential privacy [40], which is used for guaranteeing the privacy on aggregated datasets whereby it is possible to add or remove any item to or from the dataset without affecting the result of the analysis. When the model is analyzed by the adversary, he/she cannot differentiate the actual result and the item added/removed version of it if at most one item is changed. The last privacy enhancing technology suitable to the protection of PAPAYA data is differential privacy [40], which is used for guaranteeing the privacy on aggregated datasets whereby it is possible to add or remove any item to or from the dataset without affecting the result of the analysis. When the model is analyzed by the adversary, he/she cannot differentiate the actual result and the item added/removed version of it if at most one item is changed.

The purpose of this deliverable is to analyse the PAPAYA use cases introduced in deliverable D2.1 [29] describing the particular analytics operations, the protocols and the different privacy requirements and further suggest dedicated primitives ensuring the execution of the

analytics operations while protecting the privacy of the data. Three main analytics operations have been identified accordingly and are the proposed privacy preserving analytics modules are enumerated as follows:

- The first analytics operation consists of neural networks (NN). We first introduce background and definitions on NN and some initial designs of NN classification modules that employ different privacy enhancing technologies, namely partially/fully homomorphic encryption and secure multi-party computation. For this respect, we also provide basic overview of NN and the use of NN in privacy preserving manner. Furthermore, we also study the problem of privacy preserving collaborative training involving many data owners who need to jointly construct a neural network model. This solution can be considered as a solution for the privacy-preserving stress management use case (UC2) defined in deliverable D2.1 [29]

- The document further investigates the problem of privacy preserving clustering which consists of regrouping data items in k clusters without disclosing the content of the data and provides a preliminary design of a privacy preserving trajectory clustering solution based on the use of partially homomorphic encryption. Moreover, we also provide a concise summary on well-known clustering algorithms. These operations illustrate the operations for the privacy-preserving mobility analytics (UC3, see deliverable D2.1 [29]).

- Finally, the document studies basic statistical operations such as counting, set intersection and set union suitable to the privacy-preserving mobile phone usage analytics (UC4) and introduces new solutions based on encrypted bloom filters and functional encryption. For each category of data analytics, a review of the existing solutions has been provided before the reporting of the new PAPAYA primitives.

1.2 Structure of the Document

The rest of the document is organized as follows:

- **Section 2** overviews the advanced cryptographic techniques, in particular homomorphic encryption, secure multiparty computation, functional encryption and differential privacy as these are planned to be employed in the scope of PAPAYA.

- The initial design of new privacy preserving neural network classification and privacy preserving collaborative training schemes are described in **Section 3**. In this section, we provide one solution for addressing the requirements of the arrhythmia detection use case (UC1) in the healthcare umbrella in deliverable D2.1 [29]. We also provide two more generic approaches that could be applied on other NN architectures as well. Finally, the preliminary design of a privacy preserving collaborative training solution addressing the requirements of the privacy-preserving stress management use case (UC2) is presented.

- **Section 4** overviews existing privacy preserving clustering algorithms and introduces the preliminary design of the newly proposed privacy preserving trajectory clustering solution.

- **Section 5** focuses on the problem of privacy preserving counting and introduces initial designs based on encrypted Bloom filters or functional encryption and further investigates other operations such as set interaction and set union.

2 Cryptographic Techniques for PAPAYA

2.1 Homomorphic Encryption

Homomorphic encryption is a public key cryptosystem that allows processing encrypted data without learning neither the input data nor the computation result. The data owner, Alice, can thus delegate some of her computations over sensitive data to a untrusted party, Bob. Alice encrypts her data under her own public key (in gray in Figure 1), and sends it to Bob. Upon receiving the encrypted data, Bob can evaluate a circuit over Alice's inputs, obtaining a result still encrypted under Alice's public key. Bob sends the result back to Alice, who is the only party able to decrypt it with her private key (in red in Figure 1).

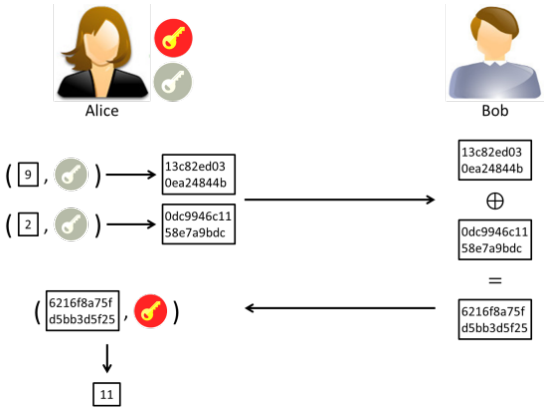


Figure 1: Alice delegates the computation of 2+9 to Bob.

Formally, a homomorphic encryption scheme is composed of four procedures, defined as follows.

- $KeyGen$: the key generation procedure takes as input the security parameter λ and outputs the public key, secret key pair (pk, sk).
- $Encrypt$: the encryption step uses the public key pk to transform a message M into a ciphertext c.
- $Eval$: the evaluation procedure takes as inputs a circuit C and ciphertexts $c_1, \ldots, c_l$ such that $c_i = Encrypt(M_i, pk)$. It outputs another ciphertext which corresponds to $c_{res} = Encrypt(C(M_1, \ldots, M_l), pk)$.
- $Decrypt$: the decryption step uses the secret key sk to transform back a ciphertext c to the message M.

We usually distinguish between three kinds of homomorphic encryption schemes:

- *Partially Homomorphic Encryption* (PHE) schemes, that support either additions, or multiplications, but not both (such as RSA [89], ElGamal [45, 32], Paillier [84]);

- *Somewhat Homomorphic Encryption* (SHE) schemes, that can perform both additions and multiplications, but in a limited number (such as BGN [16], and BGV/FV without bootstrapping [18, 43]);
- *Fully Homomorphic Encryption* (FHE) schemes, the Holy Grail, that are able to handle an arbitrary number of both additions and multiplications, and hence any circuit (such as BGV/FV with bootstrapping [18, 43]).

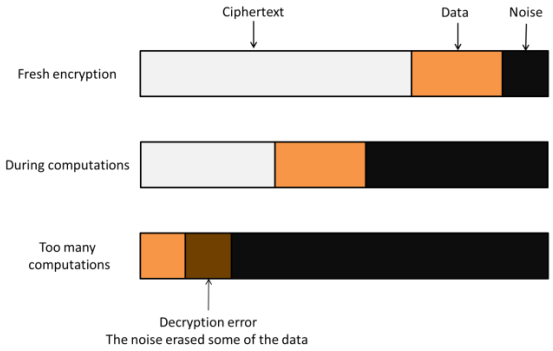


Figure 2: Noise growth in a ciphertext.

SHE schemes are based on noisy encryption, as schematized in Figure 2. The main idea is that some random noise is added to the message during the encryption to ensure security. However, computations on encrypted data result in a noise increase in the resulting ciphertext. As long as the noise remains below a certain threshold, the data can be correctly decrypted. Otherwise, when the threshold is reached, the plain message cannot be retrieved during the decryption. In a ciphertext, the room devoted to the noise growth is one of the most important parameters of the scheme. Basically, the size of the noise doubles with each homomorphic multiplication. The more computations we perform in the encrypted domain, the more space we need for the noise to grow, the slower will be each individual homomorphic operation. Hence, one of the main challenges of homomorphic encryption schemes is to manage the noise growth as tightly as possible.

However, there is a refreshing procedure, named *b*ootstrapping, that can be added to manage the noise growth and achieve fully homomorphic encryption (FHE). Although there has been recent progress regarding the efficiency of this procedure, the bootstrapping operation remains a bottleneck when evaluating circuits in the encrypted domain.

2.1.1 Four generations of lattice-based FHE

Homomorphic encryption is known since many decades ago. The concept, initially called privacy homomorphism, has been introduced in 1978 by Rivest, Adleman and Dertouzos [89] and several “basic” schemes having an homomorphic property followed: such as the well-known RSA [89] (multiplicatively), ElGamal [45] (multiplicatively, or additively, depending on the variant) and Paillier [84] (additively). In 2005, the Boneh-Goh-Nissim encryption [16] scheme

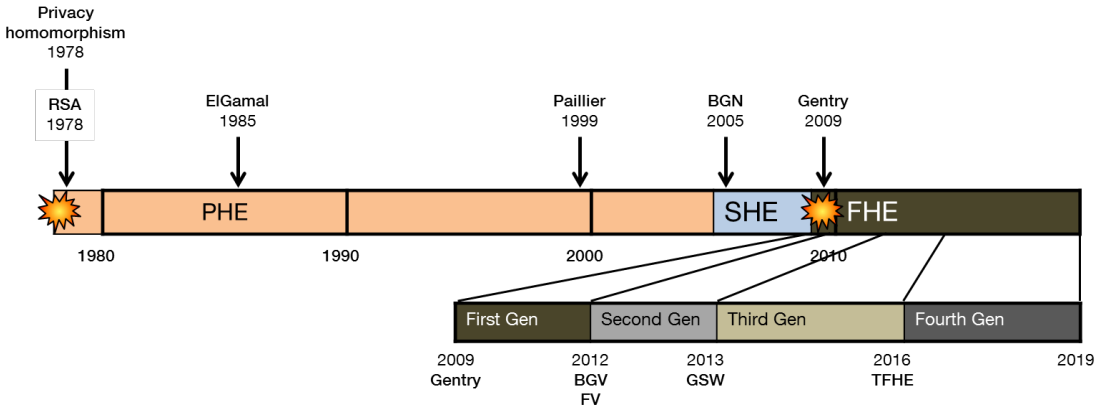


Figure 3: Timeline of homomorphic encryption.

was able to perform an arbitrary number of additions, and one single multiplication (hence becoming one of the first SHE schemes). The first FHE scheme was proposed in 2009 by Craig Gentry [48], due to an ingenious idea called bootstrapping. Since then, research on FHE has been prolific and new schemes have been proposed based on Gentry's first idea, while trying to improve the efficiency of the original but impractical scheme (see Figure 3 for a more detailed timeline).

While Gentry's work is today considered as the first generation of FHE, the second generation has been marked by Brakerski's work and two important schemes published in 2012: Brakerski-Gentry-Vaikuntanathan (BGV) [18] and Fan-Vercauteren (FV) [43], for which many optimizations have later been proposed. Nowdays one of the most useful scheme for practical application is the Cheon-Kim-Kim-Song (CKKS) [23] scheme. CKKS scheme takes a vector of complex number as plaintext, and provides native floating point arithmetic for addition and multiplication. The third generation has started in 2013 with the Gentry-Sahai-Waters (GSW) [50] scheme and a different way to represent keys. However, this generation is less used, because existing optimizations are not compatible with such a kind of representation. Recently, the fourth generation has been introduced by the scheme named TFHE [25, 26], which is based on a mathematical object called a torus, allowing to have the advantage of both second and third generations. Today, existing implementations are mostly based on the second and fourth generations.

2.1.2 Standardization

Regarding standardization, several organisations have started to work on that topic. The ISO/IEC instance has published the ISO/IEC FDIS 18033-6 on (partially) homomorphic encryptions schemes (with ElGamal and Paillier cryptosystems). The recent Homomorphic Encryption Standardization[1] is defined as an Open Industry / Government / Academic Consortium (e.g., Microsoft, IBM, NIST, MIT, etc.) to Advance Secure Computation. They propose to define white papers on security, API and applications related to (somewhat and fully) homomorphic

[1] http://homomorphicencryption.org/

Lib	Scheme	Bootstrapping	SIMD	Parallelism	API	Floats support	Entity	License
HELib	BGV, CKKS	yes	yes	yes	hard	yes	IBM	Apache2
SEAL	FV, CKKS	not public	yes	no	easy	yes	Microsoft	MIT
PALISADE	BGV, FV, LTV	no	no	no	average	no	NJIT	NJIT
TFHE	TFHE	yes	no	no	average	no	SPQlios team	Apache2

Table 1: FHE libraries properties.

encryption. There are also some parallel initiatives, such as the NIST call for proposal on post-quantum cryptography, with some schemes that are based on the same mathematical structure as SHE/FHE schemes, and the IEEE P1363-1 that integrates the NTRU scheme, which is the basis of some SHE constructions.

2.1.3 Available implementations

There are multiple implementations of (fully) homomorphic encryption. Most of them provide a leveled homomorphic scheme without bootstrapping. Many of those libraries are not updated or are programme with unsual language (e.g. $\lambda o \lambda$ with is code in Haskell). Hence we choose to focus our interest on the most mature and the more regulary updated ones. Here are the libraries we have chosen to work with:

- The Simple Encrypted Arithmetic Library (SEAL)[2] is edited by Microsoft and provides an implementation of the FV and CKKS schemes in C++.
- PALISADE[3] is a library written in C++ that lets the user choose between four schemes: FV, BGV, LTV and Stehlé-Steinfield. PALISADE depends on the libraries gmp[4] and NTL[5].
- HElib[6] implements BGV in C++ and provides bootstrapping. HElib also depends on the libraries gmp and NTL.
- TFHE[7] implements one of fourth generation of FHE schemes, that features boostrapping under 0.1 seconds.

The libraries offers different type of algorithmic optimizations. One of the more important is Single Instruction Multiple Data (SIMD). The mathematical properties of the message space allow to consider a plaintext as a vector of smaller messages. The computations are then performed in parallel for each element of the vector in a SIMD fashion, evaluating a function across many entries at the same time as shown in Figure 4.

Advantages and disadvantages of libraries are as given in Table 1.

[2] https://github.com/Microsoft/SEAL
[3] https://git.njit.edu/palisade/PALISADE
[4] https://gmplib.org/
[5] https://www.shoup.net/ntl/
[6] https://github.com/shaih/HElib
[7] https://github.com/tfhe/tfhe

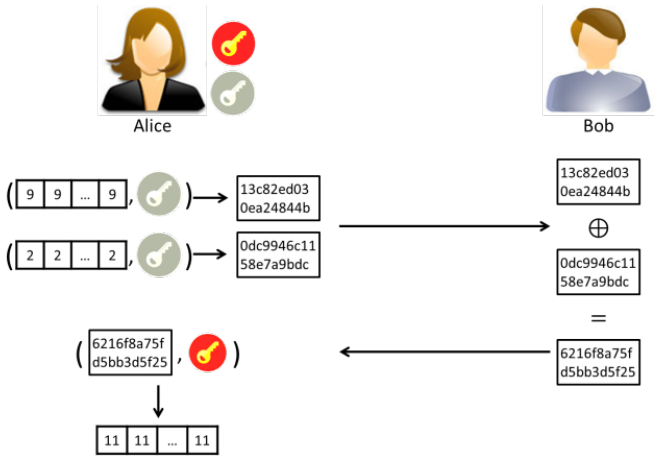


Figure 4: SIMD computations.

2.1.4 Comparison of libraries' performance

In this section, we compare the performance of each library on a small set of basic operations: Encode, Decode, Add, Mul. With the exception of TFHE, the libraries are updated regulary. The benchmarks were done a few months ago, since SEAL and HElib have been updated. We didn't update the benchmarks accordingly for this delivrable, we will for the next one. We are confident that the update did not change the conclusion of the benchmarks. We look at plaintext size of 2, 4 and 8 bits and for each plaintext size we set the parameters to allow a certain multiplicative depth: 5, 10, 20, 30. The timings are expressed in milliseconds in Table 2, and its main points are as follows:

- TFHE is the fastest to encrypt and decrypt. However it only works with binary operations, therefore to compute the addition or multiplication of two integers, one as to code the corresponding binary circuit. Hence the bad performance of THFE for the addition and multiplcation.
- SEAL is the overall best performer.
- HELib is average in terms of performance.

As a consquence, TFHE is well suited for binary computations. SEAL offers the best overall performance but does not provide a bootstrapping operation which limits the possible use cases. Finally, HELib does offer a bootstrapping which makes it well suited for complex use cases.

2.2 Secure Multiparty Computation

Secure multi-party computation is introduced in early 1980s by Yao [106, 107] who focused on the two-party computation (2PC) case by defining Yao's Millionnaire problem. Then, by Goldreich et al. in [53], the problem was generalised to multiple parties. In this section, we

		Encryption			Decryption			Addition			Multiplication		
#bits	#levels	HELib	SEAL	TFHE	HELib	SEAL	TFHE	HELib	SEAL	TFHE	HELib	SEAL	TFHE
2	5	25	3	0.03	10	0.5	0.003	1.2	0.04	157.49	43.3	5.3	140.3
2	10	73.3	5.2	0.03	32	1.13	0.003	3.3	0.1	157.49	120.9	16.3	140.3
2	20	289	11.7	0.03	131.3	3.8	0.003	13.5	0.4	157.49	461	56.3	140.3
4	5	26.2	2	0.06	9.7	0.39	0.003	1.2	0.05	308.53	45.2	4.8	631.02
4	10	76.9	4.5	0.06	31.4	1.1	0.003	3.4	0.1	308.53	124.6	14.7	631.02
4	20	289.9	12.6	0.06	130.6	4.6	0.003	13.7	0.5	308.53	485.9	67	631.02
8	5	26.8	3.4	0.1	9.5	0.79	0.003	1.2	0.05	613.3	45.1	11.3	2688
8	10	78.6	4.6	0.1	34.7	1.6	0.003	3.7	0.13	613.3	132	20.6	2688
8	20	273.6	15.6	0.1	130.2	4.9	0.003	12.6	0.4	613.3	460.6	74.9	2688

Table 2: Comparison of the performance of the FHE libraries.

formally define the problem, describe the underlying building blocks such as oblivious transfer, Yao's Protocol, secret sharing and hybrid protocol, and briefly overview some of the available implementations.

2.2.1 Definition

Secure multi-party computation (MPC) is defined as a system in which a group of data owners can jointly compute a function of their private inputs without disclosing the underlying inputs, but the output of the function. Formally, let $P_1, \ldots, P_n$ be n parties and each of them having input $x_i, \ldots, x_n$, respectively. As illustrated in Figure 5, these parties want to jointly compute the function f over all inputs $\{x_i, \ldots, x_n\}$ and learn the output without revealing their input.

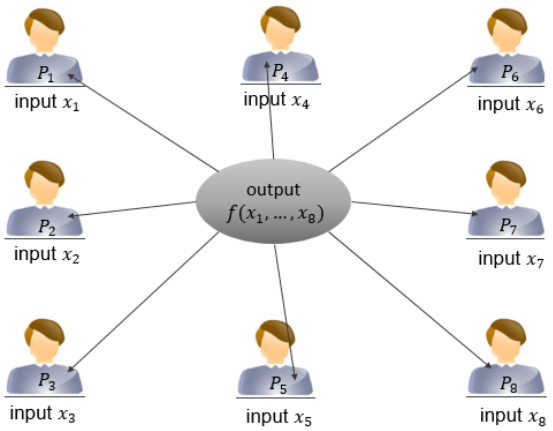


Figure 5: Secure multiparty computation

MPC should ensure the following two properties, at least: (i) *input privacy*, i.e., parties' inputs should remain private and only the output of the function is learned; (ii) *correctness*, i.e., even if some parties misbehave, the correct output is obtained.

2.2.2 Building Blocks

Existing MPCs leverage Yao's Garbled circuits [106, 107] and secret sharing (additive or Boolean) [11]. We briefly explain each method in the following sections. Before going into details, we first introduce Oblivious Transfer (OT) method.

Oblivious Transfer

Oblivious transfer (OT) [86] is a fundamental cryptographic primitive that is used as building block in MPC. OT allows a party to choose k out of n secrets from another party without disclosing which secrets have been chosen. Usually, the 1-out-of-2 OT is used, ensuring that one secret out of two of them is retrieved: Let Alice have two inputs x_0 and x_1, and Bob selects a bit b and wants to obtain x_b. OT ensures that Bob does not learn x_{1-b} and does not reveal b to Alice.

2.2.2.1 Yao's Protocol

Yao's protocol (*a.k.a.* Garbled Circuits (GC)) is a secure two-party computation that allows the two parties to evaluate a function $f(x_1, x_2)$ in the presence of semi-honest adversary (i.e., this adversary has to truly follow the protocol yet s/he can try to extract information during the execution of protocol), where inputs x_1 and x_2 are provided by two parties, namely Alice and Bob. Let Alice be the Garbler and Bob be the Evaluator. Alice builds a garbled version of a circuit for the function f by obfuscating all possible outputs in the truth table. The garbled circuit and Alice's garbled input $GI(x_1)$ are sent to Bob. Alice also provides a map from the garbled-circuit outputs to the actual bit values. After receiving the circuit, Bob uses 1-out-of-2 OT [86] with Alice to obliviously obtain his garbled circuit values $GI(x_2)$ without revealing it to Alice. Bob further evaluates the function $f(x_1, x_2)$ using $GI(x_1)$ and $GI(x_2)$.

The function f is evaluated through a Boolean circuit. The garbler assigns two keys that correspond to bit values 0 or 1 for each wire of the circuit. Then, Alice, the garbler, computes four ciphertexts for each binary gate with the input wires and the output wire. After obtaining ciphertexts, Alice randomly orders these four outcoming values. The evaluator can decrypt the correct row from the table if he successfully obtains the pair of keys from Alice via OT.

2.2.2.2 Secret Sharing

Alternatively to Yao's Garbled Circuits, MPC solutions based on secret sharing consist of distributing secrets among parties involved in the system and further evaluate the function defined as a circuit accordingly. The GMW protocol [53] relies on Boolean shares and mainly support XOR operations over single bits. The function to be evaluated is encoded as a Boolean circuit and OT is used during the evaluation of the circuit. The Boolean circuit takes as inputs bit u from Alice and bit v from Bob. These bits are first secret-shared between the parties as $u = u_1 \oplus u_2$ and $v = v_1 \oplus v_2$, where share 1 belongs to Alice and share 2 to Bob. Then both parties evaluate the circuit gate by gate. For example, given shared values, an XOR gate with input bits u and v and output bit w is evaluated locally (i.e. without communication) by each party by computing $w_i = u_i \oplus v_i$. Value w can be retrieved by exchanging and XORing

the shares. The evaluation of AND gates is more challenging and requires Alice and Bob to interact.

Some other solutions use arithmetic circuit whereby inputs are additively shared and addition gates (respectively multiplication gates) correspond to XOR gates (resp. AND gates). It is worth mentioning that Liu et al. [74] extend the additive secret sharing by introducing the dot-product triplets used for computing the dot product of two secret vectors in a privacy-preserving way.

2.2.2.3 Hybrid Protocol

Some MPC implementations rely on a hybrid method that consists of a mixed-protocol that combines the usage of arithmetic circuits with homomorphic encryption and/or the usage of Boolean circuits as defined in ABY framework [34] or TASTY tool in [58]. This mixed approach show better performance results.

2.2.3 Available Implementations

Several practical open-source implementations for 2PC/MPC systems have been proposed in recent years. Some consist of high-level description languages and corresponding compilers used to specify the function to be securely evaluated and to translate it into a Boolean or arithmetic circuit, for example: Fairplay [79] and its extension to multiple parties, FairplayMP [14]. Other implementations offer libraries for MPC such as SCAPI [41]. Finally, some other solutions propose more comprehensive frameworks consisting of libraries, languages and their compilers, runtime environments and OT tools such as TASTY [58], ABY [34] or EMPtoolkit [99].

2.3 Functional Encryption

Functional encryption is an encryption mechanism enabling to obtain an authorized function of an encrypted data. The data owner, Alice, can provide Bob with a functional decryption key (*a.k.a.* evaluation key). This key, which is related to a function f, allows him to recover the result of f applied to Alice's data. It is worth noting that, unlike homomorphic encryption, Bob actually gets the result of the computation. In return, Bob is only allowed to execute the function for which he has obtained the evaluation key, and no more than that. More formally, functional encryption is composed of four procedures:

- $Setup$ takes as input a security parameter, a set of functions $\mathcal{F}$ and outputs a master secret key msk and a public key pk.
- $KeyGen$ takes as input the master secret key and a function $f \in \mathcal{F}$ and outputs an evaluation key dk_f.
- $Encrypt$ takes as input a message m, the public key pk and outputs a ciphertext c.
- $Decrypt$ takes as input an evaluation key dk_f, a ciphertext c, and returns $f(m)$.

2.3.1 Possible functionalities

Regarding constructions, this is quite difficult to be exhaustive, as the number of papers on the subject has recently exploded. There are in fact different types of results.

The functionality (the function f above) that can be used in a functional encryption can by nature be anything, and some constructions treat this "generic" case as Boolean circuits [46, 4]. The resulting schemes are today far from being practical, and sometimes based on questionable assumptions [8].

There are more concrete constructions treating the case of some particular mathematical functionalities. The most largely studied ones are the inner product [5, 15, 3] and the quadratic polynomials [10, 38]. The resulting efficiency (in terms of timing and resources) is in this case quite good, even if such a cryptographic tool is considered as very complex.

Some papers offer to functional encryption various additional paradigms, such as multiple entries [3], hiding the functionality [19, 7, 44], using secure hardware [44] or some external help [39]. Many other properties or techniques can certainly be treated in the case of functional encryption. These new properties can be related to the underlying use case (e.g., distributed decryption, new specific mathematical functionality, etc.) or to improve the security (e.g., using a secure environment) or the effectiveness of specific constructions.

2.3.2 Available implementations

To the best of our knowledge, there is only one implementation of a real functional encryption available on the Web. This one is based on the work by Kim et al. [68] on function-hiding functional encryption (FHIPE)[8].

2.4 Differential Privacy

Differential Privacy (DP) defines a strong standard for privacy guarantees on aggregated datasets. Differential privacy defined in application context. In a neural network context DP defined in the following way:
Each dataset is a set of pairs ($x - input$, $y - label$), we define adjacent of two training sets if they differ only in a single pair. The definition of $\varepsilon - differential\ privacy$ form is: A deep learning model M provides $\varepsilon - differential\ privacy$ if for all adjacent datasets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(M)$:

$$Pr[M(D_1) \in S] \leq e^{\varepsilon} \cdot Pr[M(D_2) \in S]$$

The intuition behind the DP definition is that adding or removing any item to or from the dataset does not affect the result of the analysis. When an adversary is analyzing a model with knowledge that at most one item is added or removed from the dataset, he cannot differentiate the answers (identify the value of the pair added or removed).

The parameter ε in the definition is a privacy parameter. ε is the threshold that determines the privacy loss or the utility.

[8] FHIPE: `https://github.com/kevinlewi/fhipe/` [Last update: March 2016]

The additional definition of $(\varepsilon, \delta) - differential\ privacy$ form if:

$$Pr[K(D_1) \in S] \leq e^{\varepsilon} \cdot Pr[K(D_2) \in S] + \delta$$

This variant introduced by Dwork et al. [40] This definition allows the possibility that plain $\varepsilon - differential\ privacy$ is broken with probability δ (which is preferably smaller than $1/\|D_1\|$).

3 Privacy preserving Neural Networks

3.1 Background and Definitions

Neural Networks have been applied in many domains to model complex data such as text, images, speech, and others, where they have demonstrated results superior to other machine learning techniques. However, training and application of Neural Networks (e.g. for the purpose of classification) presents serious privacy issues. One of the main PAPAYA objectives is to develop techniques for privacy preserving training and application of neural networks to make them available for a wide range of different stakeholders. With this being mentioned, this section is structured as follows: subsection 3.1 provides background on Neural Networks and relevant definitions; subsection 3.2 is dedicated to privacy preserving classification on neural networks; and finally, subsection 3.3 addresses privacy preserving collaborative training of Neural Networks.

Neural Network

A standard Neural Network (NN) consists of many connected processing units called neurons (perceptron is a more common name in a computer science parlance). Each neuron defines a parameterized matrix of weights W also called neuron's weights. The matrix weight dimension depends on the input vector and the NN architecture. The neuron function calculates the multiplication of the input vector by weights matrix and the result proceeds to the activation function. The purpose of **Activation function** [101] is to add non-linearity to a neuron's output. Adding non-linearity allows us to make more complex decisions, where the relationship between the input to the output is a non-linear. Furthermore, the activation function determines whether a neuron will be activated or not. There are several commonly used activation functions such as **Sigmoid, hyperbolic tangent (tanh), Rectified Linear Unit (ReLU), Leaky ReLU, Maxout, etc.**, each of which provides some desirable properties. NN model is often organized in distinct layers.A model with multiple layers between the input layer and the output layer called Deep Neural Network (DNN). The type of each layer depends on NN functionality. This type defines the layers inner structure and connectivity to the next layer (see NN Architectures description below). A Dataset is defined as a set of pairs (x, y) where x associated with input, y associated with output and also called label. Given a dataset of pairs (x_i, y_i), **NN training** is about finding the best neurons' weights W that will provide the correct consistent output y_i for each input x_i. **Loss Function** [103] measures the quality of the neurons weights based on how similar the predicted label and the truth label are. The training process seeks to minimize the loss function. This is done by performing different types of backpropagation [103]. A Backpropagation

(backward propagation) is a method to calculate neuron weights update (gradients), based on the loss function value. Since the loss function evaluated only on the output of the network, the error should be backward propagated and update the weights from the output layer to the input layer through all the network's layers. Gradient descent is an optimization method and it is the most common and established way to optimizing NN loss function. The main idea of gradient descent optimization is to follow the gradient until achieving the satisfied behavior. The accuracy of NN depends directly on the similarity between the training data and the testing data distribution.

A few of a most common NN Architectures [102]:

- Multilayer Perceptron (MLP), is the simplest architecture of a Fully Connected (FC) NN that consists of at least three layers, an input layer, a hidden layer and an output layer. Fully Connected means that all neurons on each layer (except the output layer) are connected to all neurons on the consecutive layer. MLP with three layers is also called "vanilla" neural network.

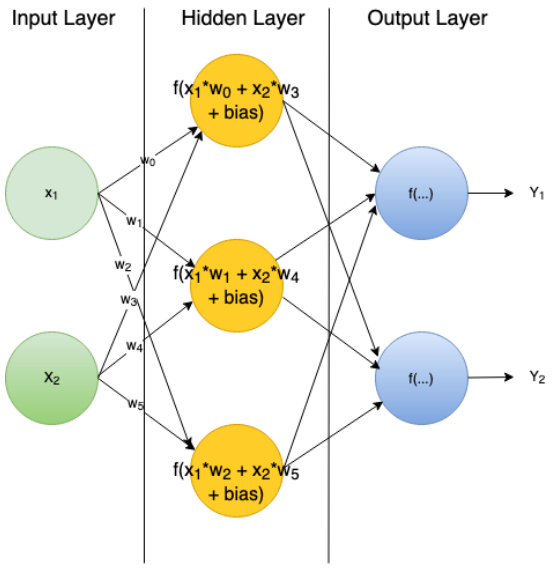


Figure 6: MLP Structure with one hidden layer. The Input layer consists of two neurons. The Hidden layer consists of three neurons. The Output layer consists of two neurons.

- Convolutional Neural networks, also known as convolutional networks or CNNs, are a specialized kind of neural networks for processing data that has a known grid-like topology. The name "convolutional neural network" hints that the network uses a mathematical operation called convolution. The convolution emulates the response of an individual neuron to visual stimuli [60]. CNNs use convolution operation instead of general matrix multiplication in at least one of their layers. CNNs may also include pooling layers to reduce the dimension of the data [104]. Convolutional (Conv) networks have been very successful in the image processing domain.

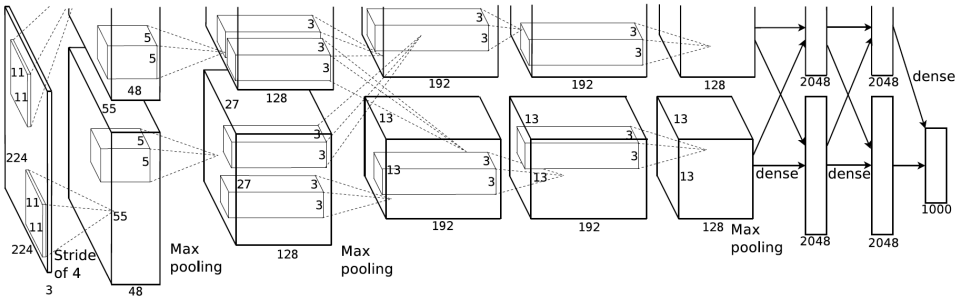


Figure 7: The architecture of AlexNet, one of the well-known CNN's, as presented in [69]. AlexNet contains eight layers: the first five are convolutional layers, some of them followed by max-pooling layers, and the last three are fully connected layers.

- Recurrent Neural Networks or RNNs, are a type of neural networks particularly used for processing sequential data. The connections between neurons form a direct graph along a sequence. Most RNNs can process sequences with variable length. As shown below, the input of each node consists of an input vector, previous node state and node input. Each node output is produced by using the same update rule applied to the previous outputs.

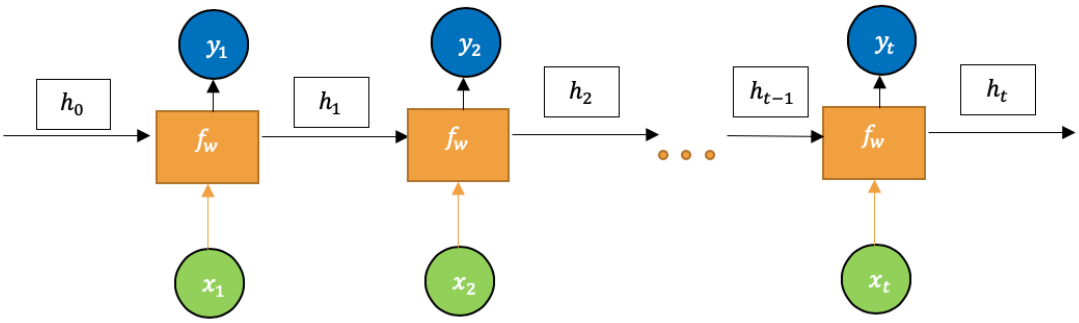


Figure 8: Basic RNN Structure.

As mentioned previously, the accuracy of NN directly depends on the amount and heterogeneity of the training dataset. These datasets may contain sensitive information that according to various regulations should not be exposed. To address this problem several privacy preserving deep learning techniques have been proposed. They could be divided into two groups according to stage of the learning process they address:

1. **privacy preserving training** - aims at inferring the algorithm parameters (i.e. weights for Neural Network) from a labelled dataset by optimizing some training objective in a privacy preserving manner.

2. **privacy preserving real time stage** – depending on model objectives (e.g. classification of a new data) applying the trained algorithm to new data in a privacy preserving manner.

With respect to the first stage, in PAPAYA we address only the privacy preserving **distributed** training of a deep neural network (DNN), where the proposed solutions should allow multiple participants to train Deep Neural Network collaboratively (to benefit from data of all participants) while preserving privacy of each participant's data. Regarding the second stage, in PAPAYA we are interested in Privacy Preserving classification on Neural Networks. In particular, we address the case when the trained Neural Network is available (and typically located on the server side), and the goal is to apply this network on (sensitive) data of different clients such that the server learns nothing about the data, and client learns the classification results only.

In the following subsections we discuss in details PP classification on DNN for single data owner setting followed by Privacy Preserving Collaborative training.

3.2 Single data owner setting

3.2.1 Privacy challenges

The main challenge is to provide privacy enhancing technologies which allow to various clients to apply Neural Network belonging to some 3rd party in order to classify his/her (sensitive) data, such that the 3rd party learns nothing about the client's data, and the clients learn the classification results only (and nothing about the NN model). We have identified the following three additional challenges when building a neural network model customized for the use of privacy enhancing technologies:

- Size of the NN: In order to efficiently integrate privacy enhancing technologies, the size of the neural network should be optimized. Therefore, while designing the NN compatible to the use of 2PC, one should reduce the size of the input layer, the hidden layers and, the output layer.
- Complex NN operations: A neural network involves various operations executed by each neuron during the classification phase. These include sophisticated operations such as sigmoid or tanh that may not be easily and efficiently supported by existing cryptographic tools. Hence, the underlying operations should be optimized and sometimes even transformed when designing the privacy-preserving variant of the neural network classification model.
- Real numbers instead of integers: Most of the operations in the neural network are executed over real numbers whereas cryptographic tools usually support integers. Therefore, there is a need for either supporting floating point numbers or approximating them to integers. Such an approximation should nevertheless not have a significant impact on the accuracy of the model.

The size of the neural network directly depends on the size of the input and output vectors, the number of layers, and the number of neurons in the model. Such parameters have a significant impact on the complexity of the model. To reduce the overhead resulting from introducing

the privacy-preserving variants of the underlying operations, the number of operations, hence the size of the neural network has to be optimized. Such an optimization, on the other hand, should not have an impact on the actual accuracy of the model.

3.2.2 Related work

All methods that have been proposed for the Privacy Preserving Classification on Deep Neural Networks are based on secure multi-party computation (MPC) or on homomorphic encryption (HE) or on a combination of both methods to compute scalar products and activation functions. In following subsections we discuss the latest approaches belonging to each of the aforementioned categories.

3.2.2.1 FHE Based Solutions

3.2.2.1.1 CryptoNets [36] In CryptoNets, authors presented a method for private classification of data using convolutional neural network models. Thus, this work does not consider privacy preserving training.

The proposed solution relies on fully homomorphic encryption (FHE) alone. That is, the client is required to encrypt its input and send it to the server, which in turn evaluates the neural network classification homomorphically and sends it back to the client. This simple approach has security benefits that some MPC solutions don't enjoy from (to be elaborated later). In addition, relying on FHE alone has the benefit of non-interaction: the server may perform classifications later in time in the absence of the client.

A drawback of the above approach is that activation functions need to be approximated by low-degree polynomials, otherwise noise growth becomes unmanageable. This incurs difficulties in the training process and reduces achievable accuracy. In addition, the use of Leveled FHE discourages deep neural networks as the computational complexity grows polynomially with depth, unless bootstrapping is used. Unfortunately, bootstrapping may be too prohibiting in terms of concrete computational complexity.

Authors employ batching for FHE not to accelerate the evaluation of a single neural network classification, but to perform many evaluations in parallel. This approach has the benefit of simplicity, as no special algorithms need to be developed to perform tasks such as matrix multiplication. Hence, this work achieves high throughput, as many classifications can be done in parallel, but the latency is also high, as performing a single classification takes as much time as an entire batch.

The system of CryptoNets is built on Simple Encrypted Arithmetic Library (SEAL) by Microsoft, which implements homomorphic encryption. The FHE scheme of choice is YASHE.

Security Analysis CryptoNets achieve privacy under the semi-honest adversary assumption [57] by the fact that client and the server may not learn anything about the input of the other. The client may not learn anything, as it receives ciphertexts containing only the classification output. The server also learns nothing due to the semantic security property of the FHE scheme.

As given above, CryptoNets in fact achieve security guarantees that are better than privacy under the semi-honest assumption, which is common among MPC approaches. Assume that the server is not confined only to neural network classification and may provide the function on which the client's output is evaluated as input, we conclude the CryptoNets approach is secure against malicious adversaries. Indeed, if the server affects the client's output such that it does not provide good classifications, the client may complain that the server does not hold its end of the bargain. Hence it does not count as a breach of security, as the server could have provided a bad model for classification even in the presence of a trusted third party.

Performance Analysis Authors compared the performance of a convolutional neural network for classifying the MNIST dataset(i.e., this dataset contains of 28×28 grayscale images of handwritten digits in range $[0 - 9]$) in their framework. The network considered is the following: 1-Conv and 2-FC with square activation. The choice of cryptographic parameters allows the evaluation of 4096 classifications in parallel. Table 3 summarises performance results:

Layer	**Time (sec)**	**Time per instance in batch (ms)**
Convolution	30	7
Square Activation	81	20
Fully Connected	127	31
Square Activation	10	2
Fully Connected	1.6	0.4
	Message Size (MB)	**Size per Instance (KB)**
Client $\rightarrow$ Server	367.5	91.875
Server $\rightarrow$ Client	4.7	1.17

Table 3: CryptoNets - performance results

Summary & Suitability to PAPAYA The work on CryptoNets presents a method for private neural network classification which enjoys from high throughput and good security guarantees. Unfortunately, there are many drawbacks to CryptoNets' approach. First, the neural network model is assumed to be given to the server. Next, this approach suffers from high latency for classification and difficulties in training and classification due to activation function approximation by polynomials. Finally, the choice of Leveled FHE becomes very prohibiting from a computational point of view when considering deep neural networks. This is because computational cost grows polynomially with depth and using bootstrapping, which makes the cost grow linearly, is too costly practically. Practically, the public availability of SEAL benefits CryptoNets, as their approach probably can be implemented as the cryptographic parameters appear in the paper. However, as PAPAYA aims for deep neural networks to be employed, assuming the

training problem is overcome somehow, even with the good security guarantees, CryptoNets does not seem like a good fit as it cannot be applied to deep networks.

3.2.2.1.2 Chabanne et al.'s Work [22] Similarly to CryptoNets, this solution also ensures privacy preserving classification of image data based on the use of fully homomorphic encryption within convolutional neural networks. The main improvement compared to CryptoNets' is the better level of accuracy obtained thanks to the integration of a batch normalization layer [61]. This new layer increases the accuracy of the model and mainly consists of subtracting the intermediate values from the previous layer with the mean and dividing the result with the standard deviation.

As the goal of the new solution is to keep accuracy as high as possible while using FHE to protect the input data, authors suggest to approximate the activation function with a low degree polynomial both on the learning and classification phases. They use only the even power of polynomials to approximate the nonlinear function, the rectified linear unit (ReLU).

The solution makes use of privacy preserving (PP) NN classification is based on BGV [18] with the Smart-Vercauteren ciphertext packing techniques [94] and the Gentry-Halevi-Smart optimizations [49]. HElib [56] is used for FHE in this study.

Security Analysis Thanks to the use of FHE, the input data remains confidential throughout the entire classification phase. Only the data owner who encrypted the input can retrieve the output of the classification by decryption.

Performance Analysis The proposed solution achieves an accuracy level of 99.30% using a polynomial of degree 2 in the NN studied in [36]. Authors propose to evaluate two CNNs: one with 9 layers and one with 24 layers. The classification accuracy of the 9-layer CNN reaches 97.95% when using the ReLU function as an activation and 97.91% when approximating the ReLU function with a polynomial of degree 6. On the other hand, authors obtain an accuracy level of 99.59% with the 24-layer CNN when six ReLU activation layers are implemented with batch normalization (BN), and 97.91% when these activation functions are approximated with polynomials of degree 4. Regarding the computational overhead, authors unfortunately do not provide details on the actual implementation of the solution. Details on the integration of the BN layer are omitted. Hence, the efficiency of the solution is not discussed.

Summary & Suitability for PAPAYA The accuracy of the solution is strongly linked to the quality of the polynomial approximation of the activation function. Authors combine this polynomial approximation (with degree higher than 2) with an additional batch normalization layer which results in a significant increase on the accuracy. The proposed solution can be interesting for PAPAYA whenever convolutional neural networks are used.

3.2.2.2 MPC Based Solutions

3.2.2.2.1 MiniONN [74] MiniONN consists of a privacy preserving convolutional neural networks that uses secure two-party computation to achieve input privacy during prediction.

Authors design some oblivious transformations for each CNN operation and achieve some negligible loss in accuracy. In their setting, a server S has a CNN model and a client C gives its input to S to learn the predictions. The underlying model is defined in Equation 1.

$$z = (W_L \cdot f_{L-1}(\dots f_{L-1}(W_1 \cdot X + B_1)\dots) + B_L) \tag{1}$$

where $f_i(\cdot)$, $i = 1, \dots, L$ is an arbitrary transformation, $W = (W_1, W_2, \dots, W_L)$ and $B = (B_1, B_2, \dots, B_L)$ are the weight matrix and the bias vector used in the CNN, respectively.

The underlying cryptographic tools are secret sharing and Garbled circuits during the online prediction phase and the additive HE with SIMD batch processing technique during the offline precomputation phase. The activation functions such as as $tanh$ and $Sigmoid$ are approximated with polynomials. In MiniONN, polynomial splines are used to approximate nonlinear functions (activation functions or pooling layers' functions) within negligible loss in prediction accuracy.

Security analysis The main security assumption for this solution is that the server and the client cannot collude. The server is considered semi-honest. A potentially malicious server S tries to learn X while a compromised C wants to learn W and B. The solution does not protect auxiliary information such as the size of X, W and B and the function $f(\cdot)$. In their setting, C can use the prediction service provided by S as a black box to obtain an approximate or equivalent model. To solve this, S can set a limit for the prediction request from a given C.

Performance analysis Authors achieve an accuracy of 97.6% when using the MNIST dataset over a NN defined by [81]. Authors also compare their solution with the CryptoNets one and obtain an accuracy level of 98.5% with the CNN integrating the square function as the activation function and mean pooling replacing max pooling. Another dataset, namely CIFAR-10 (i.e., this dataset contains of 32×32 colour images in 10 classes, with 6000 images per class), is used to evaluate the performance of the solution using a NN with 17 layers. These experiments result in an accuracy of 81.61%. Table 4 shows the performance of MiniONN.

Table 4: Performance of MiniONN

	Latency (sec.)		Message Size (MB)		Accuracy %
	Offline	Online	Offline	Online	
ReLU / CNN / MNIST	3.58	5.74	20.9	636.6	99.0
ReLU / CNN / CIFAR-10	472	72	3046	6226	81.61

Summary & Suitability to PAPAYA MiniONN uses 2-party computation in order to ensure privacy preserving prediction with neural networks. The use of 2PC for privacy-preserving NN seems an interesting asset for PAPAYA. The suitability of these for a dedicated NN defined for the PAPAYA use cases can be evaluated.

3.2.2.2.2 SecureML [81] In [81], the aim is to develop a privacy-preserving training and classification of NNs using 2PC. This study is the first implementation of PP-system for training neural networks with high efficiency. Authors implement a fully connected NN with 2 hidden layers (use ReLU and square function for the activation function, and use an approximation of softmax function for the output layer).

ReLU is computed with Yao's sharing. The proposed system is a 2-server model where data owners encrypt their private data, and secretly share among 2 non-colluding servers in the set-up phase while in the computation phase, two servers can train various models on the joint data of clients without learning any information beyond the trained model. Moreover, a truncation technique is proposed to convert decimal arithmetic to an integer field.

Security Analysis The authors suggest a server-aided setting in which two untrusted and non-colluding servers compute the outsourced data given by the clients. The adversary is assumed to be semi-honest and is able to corrupt only one of two servers and a subset of the clients.

Performance Analysis The experiments described in the paper are executed on two different datasets: MNIST and Arcene (i.e., this dataset is related to a two-class classification problem on cancer)[9]. Authors achieve an accuracy of 97.6% when batch normalization is used during the training phase. An accuracy level of 93.4% is achieved using their approximation for softmax. The computation of ReLU with Yao sharing dominates the online training time. In order to reduce the time for training, authors use the square function instead of ReLU, and thus obtain an accuracy of 93.1%. The online phase is much faster, but this uses more multiplication shares and boosts the cost of the offline phase. Authors run their privacy-preserving training solution over a 3-layer NN with 266 neurons and the obtained runtime is 21000s. When using ReLU, the online phase of their protocol takes 4239.7s, and the offline phase with OT takes $2.9 \cdot 10^5 s$.

Summary & Suitability to PAPAYA Similarly to MiniONN, SecureML also uses 2-party computation to ensure data privacy for neural networks. The main novelty is that SecureML supports the training phase. Nevertheless, performance results still remain prohibitive and therefore, in PAPAYA, we do not consider to design a privacy preserving variant of a neural network training algorithm.

3.2.2.3 Hybrid Solutions

3.2.2.3.1 GAZELLE [66] GAZELLE is a library for secure neural network inference. In this work the authors implement a protocol which combines FHE and MPC (via Garbled Circuits) to compute neural network classifications privately. In similar fashion to Cryptonets, the authors deal only with the problem of private classification (without learning) and assume the neural network model is given to the server in the clear. The strength in GAZELLE's approach lies

[9] https://archive.ics.uci.edu/ml/datasets/Arcene

in the understanding that FHE and MPC have merits and shortcomings compared to one another in realizing private evaluation protocols of various functions, in terms of computation and communication complexity. Private computation of convolutional neural networks benefits from this approach as it features functions from both categories. Fully connected, convolutional and ReLU layers are considered.

FHE performs better than SMC in the following cases:

1. Computation of small multiplication depth (ideally multiplicative depth '0', i.e., linear functions)

2. The Boolean circuit has large size compared to the input size (i.e., quadratic the size of input size)

Therefore, fully connected and convolutional layers are computed via FHE by using specialized algorithms that utilize batching to accelerate computation. ReLU activation layers are computed via MPC. Transitions between FHE and MPC are performed by having each participant hold an additive secret sharing of the intermediate result. This allows to take FHE with very low noise capacity (which results in efficient computation). Transitions between FHE and MPC act effectively. Another feature of these transitions is that computation and communication costs grow only linearly with network depth. The above ideas, namely the transitions between FHE and MPC and the specialized algorithms for linear layers in FHE, could be applied regardless of the given cryptographic primitives which realize FHE and MPC. However, another avenue of innovation for GAZELLE lies in the FHE implementation and parameter choice. The authors pick the Brakerski-Fan-Vercauteren (BFV) scheme with parameters which allow efficient algorithms to be used. For example, they pick the plaintext modulus p in a way that allows Barret reduction, ciphertext modulus q that allows lazy modular reduction and cyclotomic polynomial order m that allows Cooley-Tukey style Number-theoretic transform, which allows homomorphic rotations and going back and forth from the coefficient representation (polynomial) of a plaintext to the Chinese-remainder representation (batched vector).

Security Analysis

The GAZELLE library is secure for semi-honest adversaries, that is, neither the server nor the client recovers any information if they proceed by the protocol. However, if either the server or the client act maliciously they may learn private information. The client may learn information from the FHE layers by encrypting the vector in a different format than required, and both participants may learn private information by "cheating" in the MPC protocol. In addition, using GAZELLE, the client learns how many layers a neural network has, and which layers are convolutional, fully connected or activations. This happens since the client must interact differently with the server for the computation of each layer. The protocol does not reveal the weights of each layer or their exact size – but the client does learn a bound on the size of each layer (which can be increased by computing "dummy neurons" at the cost of additional computation). This kind of security guarantee is weaker compared to the one promised in CryptoNets. In CryptoNets the neural network classification is done in a purely homomorphic manner (without requiring the client to encrypt the vector with redundancies in a specific format), and so

the client may not learn anything about the neural network model and weights. In addition, the server does not learn anything either.

Table 5: Performance comparison of classification for CIFAR-10 dataset

Framework	Runtime (sec.)			Communication (MB)		
	Offline	Online	Total	Offline	Online	Total
MiniONN	472	72	544	3046	6226	9272
GAZELLE	9.34	3.56	12.9	940	296	1236

Performance Analysis

The authors of GAZELLE compared their framework with different convolutional neural networks architectures as given in Table 6 found in the literature for classifying MNIST:

A) 3-FC layers with square activation [81]

B) 1-Conv and 2-FC with square activation from CryptoNets

C) 1-Conv and 2-FC with ReLU activation from DeepSecure

D) 2-Conv and 2-FC with ReLU activation and MaxPool [74]

In addition, the authors compared the performance of GAZELLE with MiniONN in terms of runtime and communication costs while classifying CIFAR-10 datasets (collection contains $600,000$ images of 32×32 color images) as given in Table 5:

Summary & Suitability to PAPAYA

GAZELLE provides a methodology for privately classifying neural networks which enjoys from very efficient computation and communication complexity. GAZELLE's main drawbacks are the fact that it relies on the semi-honest model and that the neural network model is assumed to be given to the server, excluding the task of private neural network training. GAZELLE is efficient and scalable and so even deep neural networks (such as RNNs) can be potentially privately computed, albeit not being considered by the authors. As PAPAYA aims for deep neural networks to be employed, assuming the training problem is overcome somehow, GAZELLE seems to be like a favorable choice.

3.2.3 Privacy preserving NN Classification in PAPAYA

In this section we describe three methods for privacy preserving NN classification that we will develop and deploy on the PAPAYA platform. The first method provides a solution for MLP neural networks and is tailored to the ECG classification task, to address requirements of privacy preserving arrhythmia detection use case (UC1) in the healthcare umbrella defined in

Table 6: Comparison of Convolutional Neural Networks

Architecture	Framework	Runtime (sec.)			Communication (MB)		
		Offline	Online	Total	Offline	Online	Total
A	SecureML	4.7	0.18	4.88	-	-	-
	MiniONN	0.9	0.14	1.04	3.8	12	47.6
	GAZELLE	0	0.03	0.03	0	0.5	0.5
B	CryptoNets	-	-	297.5	-	-	372.2
	MiniONN	0.88	0.4	1.28	3.6	44	15.8
	GAZELLE	0	0.03	0.03	0	0.5	0.5
C	DeepSecure	-	-	9.67	-	-	791
	GAZELLE	0.15	0.05	0.2	5.9	2.1	8
	MiniONN	3.58	5.74	9.32	20.9	636.6	657.7
D	ExPC	-	-	5.1	-	-	501
	GAZELLE	0.481	0.33	0.81	47.5	22.5	70

deliverable D2.1 [29]. The second and third approaches, however, are more generic and could be applied on other NN architectures as well.

3.2.3.1 Privacy preserving NN classification based on 2PC

In this section, we investigate the use of 2-party computation (2PC) for NN classification algorithms. We particularly focus on the privacy preserving arrhythmia detection use case (UC1) defined in deliverable D2.1 [29] and develop a dedicated solution to address privacy concerns raised by the analysis of the ECG data for arrhythmia classification. Since secure two-party computation protocols cannot efficiently support all kinds of operations, we propose to design a new and customized neural network model that can be executed to classify arrhythmia accurately, and this, without disclosing the input ECG data.

The design of the new model customized for the use of 2PC should not result in a significant decrease on the accuracy of the classification. We remind that the goal of a NN classification algorithm is to determine to which class a given input belongs to. In the particular UC1 [29], the class x refers to a particular type of arrhythmia. The different classes are listed in table 7. By definition, the accuracy of class x is the probability that the model predicts the class x knowing that the heart beat belongs to class x ($P(predict = x/class = x)$). The overall accuracy of

the model is computed using the following formula:

$$\sum_{x \in Classes} freq(x) \times P(predict = x/class = x)$$

where $freq(x)$ represents the frequency of the class x in the test dataset. We may also refer to the precision parameter to evaluate the system. The precision of class x is the probability that the heart beat belongs to class x knowing that the model predicted the class x ($P(class = x/predict = x)$). We propose to follow a privacy-by-design approach and consider privacy requirements at the early design phase of the neural network.

The proposed model

We describe the resulting NN model with an incremental approach. This model is built using the MIT-BIH arrhythmia dataset from the PhysioBank database[10]. We extract heart beats from the Electro-Cardiogram (ECG) signals. Each heart beat is composed of 180 samples with 90 samples before the R-peak (this is the peak that helps detect arrhythmia), 1 sample for the R-peak, and, 89 samples after the R-peak.

Once heart beats were extracted, we have performed various filtering operations to create an appropriate dataset to build the neural network model. The PhysioBank database contains 23 different annotations for the extracted heartbeats. We have decided to only consider heart beats annotated with labels representing meaningful arrhythmia classes that have significant number of instances in the dataset. The resulting number of classes is 16. Table 7 provides details about our dataset (including the different 16 classes and their frequencies) from the PhysioBank database. The dataset is split such that 80% of the heart beats are used to train the network and 20% of the heart beats are used to test the performance of the model. We propose a model with two fully connected layers involving matrix multiplications, one activation function and a final softmax function that would provide the resulting arrhythmia class.

The second parameter that affects the complexity of the NN model is the size of the input vector. This inherently reduces the dimension of the first matrix used for the Fully-Connected layer. The main tool to adequately reduce the number of neurons of the input layer is the principal component analysis technique (PCA) [64]. PCA uses orthogonal linear transformations to find a projection of all input data (ECG heart beats) into k dimensions which are defined as the principal components. Hence, PCA outputs the k features with the largest variance. The first k eigenvectors of the covariance matrix (of the dataset) are the target dimensions. Moreover, the PCA transformation has another advantage: It helps reduce the noise in the ECG signals hence improve the accuracy of the model. This is due to the fact that it discards dimensions with low variance noise. The efficiency of using PCA for the ECG analysis domain has also been proved in [21].

To identify the appropriate number of eigenvalues we run a simulation with 100 hidden neurons and change the value of the input size n starting from $n = 180$. The same simulation is executed using the ReLU and the square operations as for the activation functions. The choice of these two functions instead of more sophisticated functions such as the widely used sigmoid

[10]MIT-BIH Arrhythmia Database: `https://www.physionet.org/physiobank/database/mitdb/`

Table 7: Heart beats for Arrhythmia classification and their frequency

		Original Dataset		Our Dataset	
Arrhythmia Class	Symbol	frequency	percentage	frequency	percentage
Normal beat	N	14985	34.02%	1500	31.6
Left bundle branch block beat	L	6450	14.64%	600	12.6
Right bundle branch block beat	R	5794	13.15%	500	10.5
Premature ventricular contraction	V	5712	12.97%	500	10.5
Paced beat	/	5608	12.73%	500	10.5
Atrial premature beat	A	2042	4.64%	300	6.3
Rhythm change	+	1005	2.28%	238	5.02
Fusion of paced and normal beat	f	786	1.78%	196	5.02
Fusion of ventricular and normal beat	F	647	1.47%	155	3.2
Ventricular flutter wave	!	378	0.86%	94	1.98
Nodal (junctional) escape beat	j	184	0.42%	45	0.95
Non-conducted P-wave (blocked APB)	x	155	0.35%	37	0.78
Aberrated atrial premature beat	a	123	0.28%	27	0.57
Ventricular escape beat	E	85	0.19%	20	0.42
Nodal (junctional) premature beat	J	68	0.15%	15	0.32
Atrial escape beat	e	26	0.06%	5	0.13

Table 8: Simulation results for the prediction model with different input sizes and 100 neurons in the hidden layer

Activation	Input size	Accuracy train	Accuracy test
ReLU	180	98.10%	97.00%
ReLU	48	99.35%	97.32%
ReLU	16	98.63%	97.33%
ReLU	8	96.55%	96.10%
x^2	180	97.61%	96.98%
x^2	48	99.00%	96.70%
x^2	16	97.68%	96.80%
x^2	8	95.60%	95.00%

function is due to their simplicity and hence their easy integration. The results of the simulation are shown in table 8 ("Acc." stands for "Accuracy"). We observe that reducing the dimension of the input data can sometimes increase the accuracy of the prediction model. This is mainly due to the existence of low variance noise in the ECG heart beats. From this analysis, we choose to set the input size to 16, mainly because the resulting prediction model provides good accuracy with acceptable (low) overfitting (the difference between the accuracy during the training and the testing phase is small). Hence, the number of neurons of the input layer is now set to 16, i.e., $X = (x_1, x_2, \ldots, x_16)$.

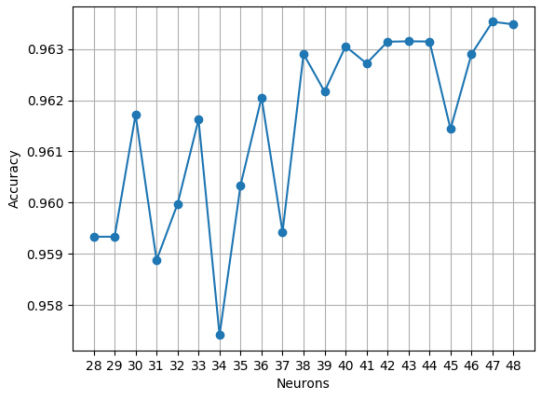


Figure 9: Accuracy with respect to different number of neurons in the hidden layer (input size:16, output size: 16)

In order to choose the appropriate number of neurons within the hidden FC layer, we have evaluated the accuracy of models whereby the number of neurons varies from 20 to 48. We not

only evaluate the overall accuracy but compute the confusion matrix that indicates the accuracy with respect to each arrhythmia class. We observe that although a model with more than 40 neurons in the hidden layer may show a better accuracy (see figure 9), 38 is a better choice as this implies less complexity in the model and the difference of the accuracy remains negligible (not more than 0.1%).

Hence, from figure 9, we observe that the accuracy of our model is 96.3% on the test data. We represent the model performance on the test dataset with the confusion matrix shown in figure 10. We also checked the precision of the model and we found that it shows a good precision (95.07%).

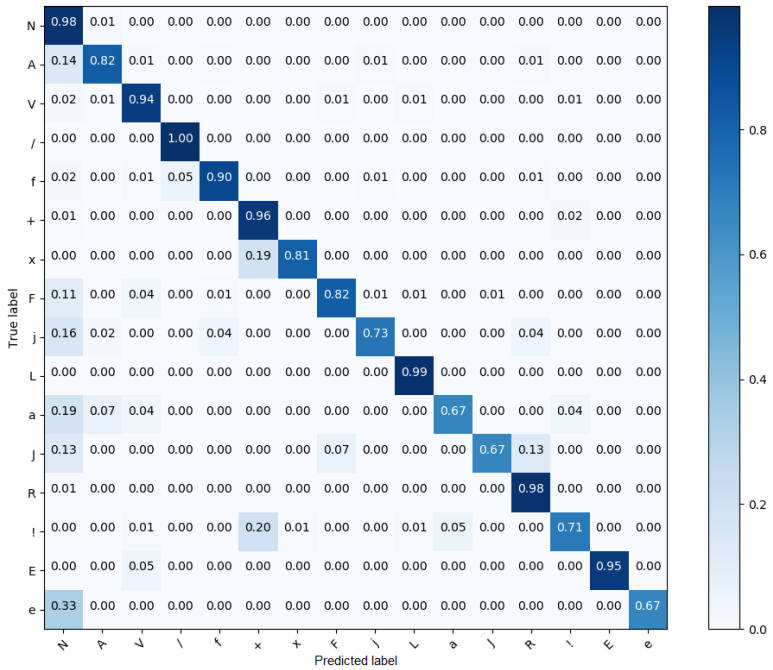


Figure 10: Confusion matrix showing the accuracy of the model for each class in the test dataset

Furthermore, in addition to the optimization of the number of neurons, we have to analyze the underlying operations. While linear layers such as convolutional or fully connected layers can easily be supported by 2PC, activation functions need to be replaced with adequate functions. We select the most appropriate activation function that cryptographic tools (in this case 2-party computation) can support. Although Table 8 shows better accuracy results when ReLU is used, we opt for the use of the square function mainly for performance reasons. Indeed, the ReLU function involves comparison operations that can incur higher overhead compared to the square function. Moreover, the resulting degradation is not very significant (0.53%).

To summarize, the developed model, compatible with the use of 2PC, consists of 2 fully connected layers, one activation layer implementing a square function and one softmax function. The architecture of the proposed neural network model is shown in Figure 11.

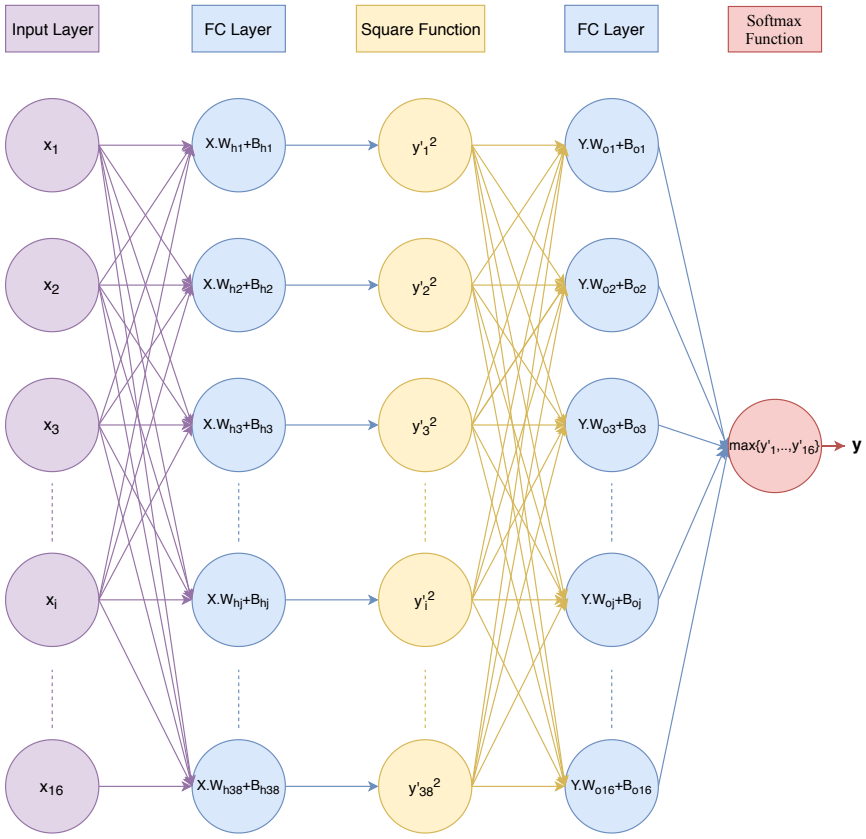


Figure 11: Diagram showing the architecture of the NN model

Privacy-preserving Arrhythmia classification with ABY

We propose to use ABY [34], a mixed-protocol framework that efficiently combines the use of Arithmetic shares, Boolean shares, and Yao's garbled circuits, to implement and evaluate the newly designed model. ABY supports many operations for Arithmetic, Boolean and Yao circuits and provides novel and highly efficient conversions between different shares.

The first solution translates the NN model regrouping the defined operations (e.g. matrix-vector multiplication, bias addition, etc.), into Boolean circuits. Both the input vector and the model are represented with matrices and vectors with floating points which values are represented as doubles (64 bits variables). Each Boolean share consists of 64 wires. When working with floating point numbers, ABY builds a specific circuit for each operational gate: For example, one floating point multiplication gate consists of 3034 XOR gates, 11065 AND gates and 3 MUX gates. Consequently, the total number of gates in the resulting circuit becomes 669,854 and the depth of the circuit is evaluated as 5,330.

The multiplication and addition of Boolean shares are much more time and bandwidth consuming than multiplication and addition of arithmetic shares. We, therefore, consider the use of arithmetic circuits only and represent real numbers with fixed-point numbers. As the multiplication of two fixed-point numbers can yield numbers with a number of bits higher than the two initial numbers, hence to an overflow, these numbers need to be truncated and/or rounded in order to ensure that all intermediate values can be represented in 64 bits. We mainly propose two truncation methods:

First truncation method: We discuss how we implement the first version of the truncation model. The precision of the inputs of the arithmetic circuits is in the order of the 6^{th} floating point: thus weights and input vector's values are multiplied by 2^{24} and the biases by 2^{48}. These are further converted to integers without having an impact on the precision of the value. Once the multiplication of the weight matrix and the input vector is executed using arithmetic gates, the intermediate values (the result of the addition of the bias vector and the multiplication of the weight matrix and the input vector) are amplified by a factor of 2^{48}. Thanks to the initial truncation, this following operation does not incur any overflow. However, moving to square function, the values need to be truncated once more. To make these truncations efficient, we propose to use a simple shift operation. Hence, shares of intermediate values are first transformed to Boolean shares and their bits are right-shifted by 24: This is equivalent to dividing the values by 2^{24}. To implement the shift function, the wires of position 2,3,...,40 (wire 1 represent the most significant bit (MSB) and wire 64 represent the least significant bit (LSB)) are moved to the position of the 40 most right wires (we do not move the first wire since it represent the sign bit). Then the wires of position 2,3,...,24 are set to the same value of the wire of position 1. This ensures correct binary representation of the values and it is compatible with negative numbers since it respects the 2's complement representation. Note that this method is also implemented in SIMD (Single Instruction Multiple Data) form as we perform the shift of the bits of all the values in the vector with a single operation. Once the truncation is applied, the Boolean share is re-converted to an arithmetic share and the arithmetic circuit corresponding to the activation function can be applied. Because of the multiplication operation, the resulting shares of the result of square function will again be amplified by a factor of 2^{48}. The same truncation method will thus be repeated. A truncation is not needed at the fourth stage as the

max operation only outputs the index and not the actual value. Finally, we convert the input of the max operation again to a Boolean SIMD share since comparison operations cannot be efficiently computed with arithmetic gates and implement the max function the same way it was implemented before.

We have also evaluated the complexity of this new model. With the truncation approach, the number of gates is reduced from 669,854 to 153,140 whereas the total depth is reduced from 5,330 to 1,104. The decrease in the number of gates and the depth of the circuit is due to the use of arithmetic gates instead of Boolean gates for multiplications and additions. Nevertheless, these remain significant because some Boolean gate are still used mainly for performing the shifting and max operations.

Second truncation method: The problem with the previous approach is that in addition to the input, intermediate values within the circuits are truncated and this had a significant impact on the accuracy of the prediction model. Furthermore, the previous truncation method also implies the modification of the actual circuit (such as adding shifters, translating arithmetic shares to Boolean shares, etc.). We consider a more simple truncation approach for which only the inputs to the circuits are truncated and this before the actual execution of the circuit. Thanks to this new approach, the actual circuit only consists of arithmetic gates except at the last stage where a max operation needs to be executed. We also propose to avoid this operation and instead, output the vector of the intermediate values to the client. Once the client obtains the last layer's values in clear, it can compute the maximum operation locally. In order to avoid overflows, all elements of the input vector, the weight matrix of the hidden layer and the weight matrix of the output layer are multiplied by 10^3, elements of the bias vector of the hidden layer are multiplied by 10^6 and elements of the bias vector of the output layer are multiplied by 10^{15}. We further truncate the fractional part afterwards. We observe that this method is safe as the maximum number a signed 64 bit integer variable can take is $9.223372037 \times 10^{18}$ and the upper bound for the intermediate values is $9,223$ and the lower bound is $-9,223$. We observe that the risk of overflow is very low.

Regarding the complexity of the circuit, thanks to the use of arithmetic gates only, the number of gates is reduced from 669,854 to 121 and the depth is reduced from 5,330 to 5. The diagram shown in figure 12 representing the arithmetic circuit easily shows that the actual depth (5) of the circuit.

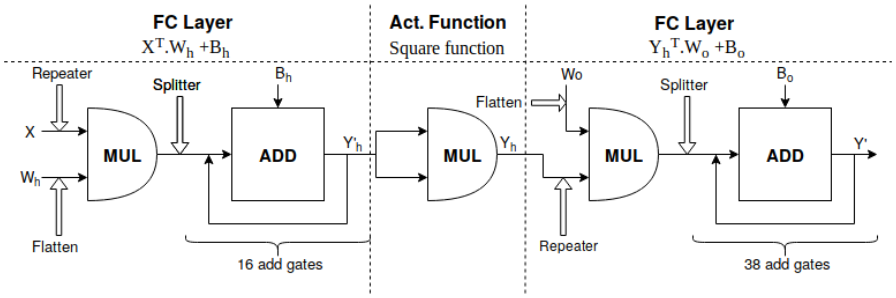


Figure 12: Simplified representation of the arithmetic circuit of the model with second truncation method

Status and next steps

The design of the solution is completed and also includes some optimizations with respect to batch classification using the SIMD approach. An alpha-version of the solution is implemented and will be tested and evaluated under UC1.

3.2.3.2 Privacy preserving NN classification based on PHE

We propose to provide privacy guarantees NN using an additively HE scheme, namely Paillier [84], with 2PC. In this report, we only introduce the preliminary design of the proposed scheme. Our main goal is to minimize the computations at the client side and the overall computational cost while maintaining privacy. In the proposed scenario, a client shares a private image with a server. The server, which holds the neural network model, computes the prediction result on the private image. The majority of computations are performed by the server, and the client is involved when intermediary decryptions are needed. The client encrypts an image with his/her public key and sends it to the server who computes the secret prediction result using the neural network parameters. Depending on the operation performed by the server, the client might be involved in the computations.

By definition, the Paillier cryptosystem's public key is the pair (N, g), where N is the product of two large primes p and q, and $g \in \mathbb{Z}^*_{N^2}$. The private key is (λ, μ), where $\lambda = lcm(p-1, q-1)$ and $\mu = (L(g^\lambda \mod N^2))^{-1} \mod N$. The encryption function of the Paillier cryptosystem is probabilistic. Encryption of a message $m \in \mathbb{Z}_N$ on modulus N is computed as $E(m) = g^m \cdot r^N \mod N^2$, where $r \in \mathbb{Z}^*_{N^2}$. An encrypted message $E(m)$ can be decrypted using the formula $m = L(c^\lambda \mod N^2) \cdot \mu \mod N$. As described in [33], the decryption function of the Paillier cryptosystem supports threshold decryption. In our paper, when necessary, we use a 2-out-of-2 variant of the threshold decryption which distributes the private key among two parties. Successful decryption requires both parties to compute the decryption function.

Using the Paillier cryptosystem, it is possible to perform addition and scalar multiplication over encrypted messages as shown in the following equations:

$$\begin{aligned} E(m_1) \times E(m_2) &= g^{m_1} \cdot r_1^N \times g^{m_2} \cdot r_2^N \mod N^2 \\ &= g^{m_1+m_2} \cdot (r_1 \cdot r_2)^N \mod N^2 \\ &= E(m_1 + m_2), \end{aligned} \tag{2}$$

$$\begin{aligned} E(m)^c &= g^{c\dot{m}} \cdot (r^c)^N \mod N^2 \\ &= E(c \cdot m). \end{aligned} \tag{3}$$

We use the following notation throughout this section: $[x]$ represents a Paillier ciphertext of a clear text x and $\langle x \rangle_i$ represents the party i's share x for 2PC operations.

We are planning to regroup all computations into two different phases: a non-interactive phase, where the operations are performed by the server without the client's involvement and an interactive phase which requires the collaboration of the server with the client for computations. In the non-interactive phase, we also provide computations for the linear layers such

as convolutional, fully connected and mean pool layers of NN. For convolutional and fully connected layers, we compute all related operations such as dot products as defined in Section . For the mean pool layer, we use linear approximation of the mean pooling operation. Following the approach in [36, 74], we compute scaled mean pool instead of the mean pool, where the summation is performed, but the division is omitted. The scaled mean pool can be computed by additive homomorphic property of the Paillier cryptosystem without interaction.

The interactive phase involves the computation of nonlinear functions that mainly are activation and pooling functions. Similarly to existing work, we propose to implement the square function as the activation function and use the Paillier encryption algorithm as shown in Protocol 1. In this protocol, to compute the square root of the encrypted input $[x]$, the server adds a random number $[r]$ to $[x]$ to perform a secure decryption, then sends it to the client. The client decrypts $[x_r]$ and takes the square of it. After square computation, the client encrypts and sends it back to the server. Then, the server subtracts the underlying random values in the protocol to get the square of $[x]$.

Protocol 1: Secure Square Protocol

Client (pk, sk)		**Server** (pk)
		$[x]$
		$r \in_R \mathbb{Z}_N$
		$[x_r] \leftarrow [x] \cdot [r]$ (this is equal to $[x+r]$)
$x_r \leftarrow \text{decr}([x_r])$		
$x_r^2 \leftarrow x_r \cdot x_r$	$\xrightarrow{[x_r^2]}$	
		$[x^2] \leftarrow [x_r^2] \cdot \left([r^2] \cdot [x]^{2r}\right)^{-1}$ (this is equal to $\left[x_r^2 - r^2 - 2xr\right]$)

Unlike the mean pool layer, we do not plan to use an approximation function for the computation of the max pool layer. Instead, we will implement the maximum pooling using the comparison gates under Boolean sharing. Then, we would like to perform the maximum pooling layer right after the activation layer to reduce the number of switching operations between 2PC and PHE. Since linear and nonlinear operations follow each other repetitively in NN, we need a secure switching mechanism between the two cryptographic techniques. We design a protocol for secure switching as shown in Protocol 2 and 3 to demonstrate the steps of switching from PHE to 2PC and 2PC to PHE, respectively.

Switching from PHE to 2PC (Protocol 2) requires to perform a secure decryption by masking the encrypted value with a random r. Once the client securely decrypts the masked value $x+r$, he creates the secret shares of it for himself and for the server as $< x+r >_c$ and $< x+r >_s$. In the mean time, the server creates the secret shares of the random r as $< r >_c$ and $< r >_s$ to remove the mask from the original value x. Finally both parties perform a local subtraction on their shares $< x+r >$ and $< r >$ to compute the secret shared value $< x >$ which is going to be used in 2PC computations.

Protocol 2: PHE to 2PC Secure Switching Protocol

Client (pk, sk)		**Server** (pk)
		$[x]$, $r \in_R \mathbb{Z}_N$
	$\xleftarrow{[x+r]}$	$[x+r] \leftarrow [x] \cdot [r]$
$x+r \leftarrow \text{decr}([x+r])$		
$x+r \rightarrow \langle x+r \rangle_c + \langle x+r \rangle_s$	$\xrightarrow{\langle x+r \rangle_s}$	
	$\xleftarrow{\langle r \rangle_c}$	$r \rightarrow \langle r \rangle_c + \langle r \rangle_s$
$\langle x \rangle_c \leftarrow \langle x+r \rangle_c - \langle r \rangle_c$		$\langle x \rangle_s \leftarrow \langle x+r \rangle_s - \langle r \rangle_s$

Protocol 3: 2PC to PHE Secure Switching Protocol

Client (pk, sk)		**Server** (pk)
$\langle x \rangle_c$		$\langle x \rangle_s$, $r' \in_R \mathbb{Z}_N$
	$\xleftarrow{\langle r' \rangle_c}$	$r' \rightarrow \langle r' \rangle_c + \langle r' \rangle_s$
$\langle x+r' \rangle_c \leftarrow \langle x \rangle_c + \langle r' \rangle_c$		
	$\xleftarrow{\langle x+r' \rangle_s}$	$\langle x+r' \rangle_s \leftarrow \langle x \rangle_s + \langle r' \rangle_s$
$x+r' \leftarrow \langle x+r' \rangle_c + \langle x+r' \rangle_s$		
$[x+r'] \leftarrow enc(x+r')$	$\xrightarrow{[x+r']}$	
		$[x] \leftarrow [x+r'] \cdot [r']^{-1}$

Switching from 2PC to PHE (Protocol 3) reverses the former procedure. It starts with a secret shared value $< x >$. Similar to the previous protocol, to prevent the leakage of the original value the parties reveal it after masking. Thus, the server generates a random mask r' and sends a secret share of the random $< r' >_c$ to the client. Both parties perform an addition operation to mask $< x >$, and then the server sends the masked value $< x + r' >_s$ to the client. The client reveals $x + r'$ by adding the two shares and encrypts it with his public key. In the final step, the server removes the random mask from $[x + r']$ with a homomorphic subtraction.

Security of the Proposed Model

In this approach, we assume that a semi-honest security model where the parties do not collude. Therefore, parties in the computation exactly follow the protocol steps. However, server is curious to obtain some information from outputs of the computations and intermediary messages. The client's goal is to hide the image content and the result of classification from the

server. On the other hand, the server does not want to reveal the model parameters used during computations to the client.

Status and next steps

Until now, we have designed a client-server model for Neural Network classification based on partial homomorphic encryption. Our main goals are on the one hand to reduce the computational cost of clients in the system and delegate this cost to the server side and on the other hand to provide the privacy for both the data and the classification results. For this respect, we will also investigate to minimize the computations at the client side and the overall computational cost for the proposed model. The complete specification and the implementation of the proposed model and its relevance to the PAPAYA usage scenarios will be described in D3.3.

3.2.3.3 FHE-based privacy preserving NN classification

Our aim in PAPAYA is to solve the so-called "privacy preserving classification" problem. Briefly, a client has his data X and a server has a trained deep NN model M. The client obtains the classification (label) of his data from the server, in such way that the server obtains nothing about the data X, and the client is not revealed to the model M. In addition, the solution must be efficient and accurate practically applicable.

As mentioned in the related work section, GAZELLE's method [66] combines traditional secure multiparty computation (specifically, garbled circuits), and homomorphic encryption in order to achieve maximum performance. In our implementation we follow the same approach, because is has the lowest latency among all other approaches. In particular, we implement the main ideas presented in GAZELLE with modifications. In following sections we provide a detailed analysis of GAZELLE framework as well as modifications we made in our implementation.

3.2.3.4 GAZELLE's framework

The framework of Gazelle consists of three parts:

1. GAZELLE's homomorphic encryption layer: The homomorphic encryption scheme that is used and implemented in GAZELLE framework is BFV [43]. The authors implemented the SIMD (single instruction multiple data) optimization. The SIMD optimization allows computing many data elements in a parallel manner. This optimization described in [94] can be applied on other homomorphic encryption schemes, specially the BGV. scheme [18].

2. GAZELLE's linear algebra kernel: This part consists of few optimizations:
   - Hosting: performing rotations is expensive and costs as much as performing number theory transform (NTT), plus the cost of $\Theta(log(q))$ inverse number theory transform NTT^{-1}. When the 'same' ciphertext is rotated more than once, then the cost

of overall rotations can be reduced. The idea is to 'hoist out' the part of the computation that would be common to these rotations.

- Input packing: usually, the number of plaintext slots is bigger than the input size, therefore, packing multiple copies of the input data into a single ciphertext allow better utilization of the ciphertexts slots (by utilization of SIMD optimization).
- Fast vector-matrix multiplication: assume given matrix with dimension of $n_i \times n_o$, and vector with length n, then the naive method of vector-matrix multiplication costs n_o SIMD multiplication, $n_o \cdot log_2(n_i)$ SIMD addition and $n_o \cdot log_2(n_i)$ rotations (automorphism). The authors develops a 'hybrid method' for vector-matrix multiplication that requires only $\frac{n_o \cdot n_i}{n}$ SIMD multiplication, $\frac{n_o \cdot n_i}{n} + log\frac{n}{n_o}$ SIMD addition, and $log\frac{n}{n_o}$, where n in the number of slots in the ciphertext (in our case, $n > n_o \cdot n_i$), see [66] for more details.
- Fast homomorphic convolution: because the convolution can be seen as a vector-matrix special case, the authors utilize and adapt the algorithms of vector-matrix multiplication to compute the convolution layer.

3. GAZELLE's inference framework: The third and final last layer, uses the combination of garbled circuits and linear algebra kernel to construct protocol for privacy preserving neural network classification. In particular, the authors implemented a switching protocol to convert between garbled circuits and homomorphic encodings without revealing any intermediate values to any party.

Our framework

As mentioned previously, we follow the approach presented in GAZELLE. Namely, the privacy achieved by computing the linear algebra (convolution, and vector-matrix multiplication) using homomorphic encryption and the non-linear part (activation functions) using secure multiparty computation. However, we use different primitives in our framework. In GAZELLE [66], the authors use an implementation of FV scheme [43] as a Homomorphic encryption library, and JustGarble [12] as a secure multiparty computation library. In our implementation, we use HElib [56] as a homomorphic encryption library, and ABY [34] as our secure multiparty library.

We chose HElib over other implementation of homomorphic encryption schemes due to the following reasons:

- HElib is a well-known, very maintained and tested library
- HElib support SIMD (single instruction multiple data), and rotations
- HElib support parallelism
- HElib implements the BGV scheme [18]. Based on [90], BGV scheme is more efficient for large plaint text space.

Similarly, we chose ABY due to the following:

- ABY is highly efficient, very maintained library

- ABY uses efficient implementation of OT extension, which allows faster activation function evaluation using MPC.
- ABY framework provides a simple API to write circuits

Status and next steps

Up until now, we have developed the convolution layer, vector-matrix multiplication algorithms, and the switching protocol, the same as GAZELLE.

Using aforementioned protocols and algorithms, we developed a prototype for image classification using CNN network, and evaluated it on MNIST data set. In spite of that we perform evaluation on CNN networks, our approach is generic and suitable for other NN architectures.

The MNIST basic image classification task: given 28×28 grayscale images of handwritten digits in range $[0-9]$, the goal is to predict the correct handwritten digit it represents. The neural network that used for the classification consists of 1-convolution layer and 2-fully connected layers with ReLU activation function. Currently, the overall classification time is 4.9 seconds (see table 9) which is greater than GAZELLE (their latency is $30ms$). However, our aim to continue to optimize the implementation.

Layer	Time (sec)
Conv-Layer	1.7
$2\times$ Garbled circuit	1.2
$2\times$ FC-Layer	2

Table 9: classification times

The flow of the prototype is as follows: the client encrypts an image and sends it to the server. The server computes the convolution layer (while the image is encrypted with HElib scheme). Then, the client and the server compute the activation function using garbled circuits. After that, the server computes two hidden layers.

Our next steps is to keep optimizing vector-matrix multiplication and the activation function part. Recently, HElib implemented fast linear algebra algorithms, and in particular fast vector-matrix multiplication, based on [55]. However, the algorithms were implemented to prove the concept, and the current implementation is not intended for general use. Therefore, we plan to extend HElib to ease the use of those algorithms.

After we improve performance of our method, we plan to evaluate in on RNN networks in order to figure out what kind of problems (addressed by RNN networks) could be solved in privacy preserving manner using our approach.

3.3 Collaborative setting

3.3.1 Privacy challenges

Privacy Issues in Collaborative Training

In order to build the most accurate model, we want the training set to be as biggest and heterogeneous as possible. When the data is distributed among multiple parties, the optimal solution is obtained when each party uploads the data to a central entity. The central entity will perform NN training on the entire data. When the training is finished, the central entity will send the trained model to each of the parties. However, whenever datasets contain sensitive information, the parties are not allowed to share the datasets due to data sensitivity and existing privacy regulations such as GDPR, ePrivacy, HIPAA etc.

Attacks on NN models

NN training on a distributed data has been getting a lot of attention that provided different methods of Collaborative Training. Similarly, it has also provided a number of attacks on the existing methods. We can split attacks on NN models into two main categories:

1. **White Box attacks** - The adversary has access to the model's description. For example: In case of NN, the adversary can see the model's architecture and weights.
2. **Black Box attacks** - The adversary can run prediction queries and see only the model's output. For example: In case of NN classification model, the adversary can see the classification result and the confidence score.

When addressing the attacks related to NN, we can split those attacks into the following categories:

1. **Models Inversion/Class Inference Attack** - this attack infers features that characterize each class. When all class' training data looks similar, the adversary may be able to generate representation of the class.
2. **Membership Inference** - Given an exact data sample and trained model, the adversary determines whether this sample was used in a model's training or not.
3. **Property Inference** - The adversary infers some property regarding the training data.
4. **Adversarial Attack** - The adversary exploits the margins between model's classification of different classes to generate input misclassified by the model.
5. **Backdoor injection into training data** - The adversary shifts the margins between model's classes, so that certain inputs are misclassified.

Most of attacks related to collaborative training assume adversary participant or adversary central server, thus most of the attacks belong to the white box attacks category. We will focus on detection and prevention of attacks relevant to our setup.

3.3.2 Related work

3.3.2.1 Privacy Preserving Collaborative Training

In order to allow multiple participants perform collaborative training while preserving the privacy of training data of all parties, we will employ and implement the method presented in [93]. The paper presents a method that enables multiple participants to perform CNN training collaboratively. Each participant performs independent local training on their own datasets. During the training, after each epoch, each participant adds Differential Privacy (DP) to the models' gradients and shares a portion of them with a central server. The server creates a joint model by aggregating the participants' uploads. The server allows the participants to download the joint model. The participants download a part of the joint model weights, replace the local model with them and continue to train the updated local model. The training is performed until the desired accuracy is achieved. Such training achieves higher accuracy than training on a local and rather smaller set of data and also preserves the privacy of it.

Privacy analysis

- Each participant performs collaborative NN training while keeping all the training data on the participant's premises.
- All participants learn the joint model and can use it privately and locally.
- Adding DP ensures that parameters' updates do not leak too much information about any individual point in the training dataset.
- Each participant fully controls which gradient to share/download and may decide not to share particular sensitive one.

3.3.2.2 Attacks on Collaborative Training

Since privacy preserving collaborative training protects privacy of participant's training data, attack on such training might be considered when the adversary participant succeeds to infer with any kind of information regarding the other participant's training data. Additional type of attacks is considered when the adversary succeeds to impact on the utility of the training process or to damage the accuracy of the NN. In the following section we will describe the most relevant and effective attacks that we would like to address and try to prevent.

3.3.2.2.1 Use of GAN for Class Inference attack [59]

Attack Description The paper presents inference attack on collaborative training. By using GAN (Generative Adversarial Networks) [54] the adversary participant is trying to generate samples which look like samples from training sets. This type of attack is considered white

box, with active adversary (not semi-honest). The adversary tries to generate samples which represent a certain targeted class. The adversary is trying to generate this class representation by training GAN model. In addition to generating a representation of a certain class, the adversary labels these generated samples with a different label and trains the local model with them. Uploading the gradients reduces the accuracy of the global model, and causes the victim participant to release more information or else the joint model will not convert.

Assumptions This attack assumes that all representatives of the targeted class look similar e.g. AT&T [1] MNIST [71].

Defense Effective countermeasures are unknown.

3.3.2.2.2 Inference attacks by observing joint model's parameters updates [80]

Attack Description The paper presents two types of attacks on Collaborative Training, Membership Inference and Property Inference. For both types of attacks the adversary needs auxiliary data. Membership Inference requires a specific data sample that the adversary wants to infer with. For Property Inference attacks the adversary needs data correctly labeled for property and for the main task. On each iteration i the adversary downloads the joint model and observes the gradient uploaded by other participants. He saves the snapshots of the joint model weights θ_t.
$\Delta\theta_t = \theta_t - \theta_{t-1} = \sum_k \theta_t^k$ - equals to aggregated updates from all participants.
$\Delta\theta_t - \Delta\theta_t^{adv}$ - equals to the aggregated updates from all participants other than the adversary.

- **Membership Inference Attack** - The adversary exploits the leakage from the **Embedding Layer**.
  The **Embedding Layer** is used to transform inputs into lower dimensional vector representation. During the training the embedding layer is updated only with the words that appear in the batch. The gradients of other words are zeros. The difference in this layer directly reveals which words occur in the training batches.

- **Property Inference Attack** - The paper presents two following phases of property inference attacks :
  - **Passive adversary** - The adversary leverages the global model to generate gradients based on data with and without property. This produces labeled batch gradients for targeted property. The adversary trains the batch property classifier with these gradients. The batch property classifier will determine whether the observed updates are based on a data with or without property.
  - **Active adversary** - The adversary extends his local copy of the collaboratively trained model with an augmented property classifier connected to the last layer. He trains this model to simultaneously perform well on the main task and recognize batch properties. During the training the adversary uploads the gradients based on

an augmented model, causing the joint model to learn separable representations for the data, with and without property. As a result, the gradients will be separable too. This enables the adversary to tell whether the training data has the property. The adversary is still "semi-honest" in the crypto parlance.

Assumptions The effectiveness of the attack depends on the data distribution and the number of participants. When the number of participants equals to 2, the adversary may discover information regarding the participant's data, directly. When the number of participants is greater than 2 and the adversary does not have any additional information regarding data distribution, the inferred information may not directly reveal the identity of the participant to whom the data belongs to and the adversary task becomes harder.

Defense The paper experimented the following defense technics:

- **Sharing fewer gradients** - Even when only 10% of gradients are shared, the adversary achieved high attack accuracy.
- **Differential Privacy** - In theory DP bounds the success of presented inference attacks. However applying of DP during the training stage, occasionally may prevent the convergence of the model.

3.3.2.2.3 Backdoor Attack [9] The paper presents attacks oriented to collaborative learning performed on users' devices. Such learning is also called Federated learning. Federated learning performed between thousands or millions of users. Due to the large number of participants the assumption that some of the devices might be compromised is reasonable. A compromised participant (adversary) submits a malicious model which is trained for both, main task and backdoor functionality (so that certain inputs are misclassified). The main idea is:
1. On each round t, the central server randomly selects a subset of m participants S_m and sends them the current global model G_t. (Choosing m involves a tradeoff between the efficiency and speed of training.)
2. Each selected participant updates this model to a new local model L^{t+1} by training on their private data
and sends the differences $L^{t+1} - G^t$ back to the server.
3. The central server averages the received updates to obtain the new global model:

$$G^{t+1} = G^t + \frac{n}{\eta} \sum_{i=1}^{m} (L_i^{t+1} - G^t)$$

where η denote the global learning rate.

Attack Description The adversary has full control over one or several participants. The adversary wants to produce a global model that converges and performs good on it's main task while also performing well on backdoor inputs (which should be misclassified). The attack focuses on **semantic backdoor**. Specifically, classification with semantic backdoor assigns

an attacker chosen label to all images with certain property or a backdoored word-prediction model suggests an attacker chosen word to complete certain sentences.
An additional adversary goal is to invade anomalous among the other participants updates for whatever definition of ”anomaly” is used by the central server.
When the training algorithm averages the uploaded gradients, the adversary has to scale local gradients in order for the backdoor to remain.
The papers present 2 approaches:
Model Replacement - In this approach the adversary continuously attempts to replace the global model with the malicious one X. Knowing the averaging algorithm, the adversary calculates the local model L^{t+1} such as:

$$X = G^t + \frac{n}{\eta}\sum_{i=1}^{m}(L_i^{t+1} - G^t)$$

As a model starts to converge the differences between participants' local models start to cancel out.

$$\sum_{i=1}^{m-1}(L_i^{t+1} - G^t) \approx 0$$

Thus:

$$L_{adv}^{t+1} = \frac{\eta}{n}(X - G^t) + G^t$$

Intuitively, this attack scales up the weights of the backdoored model X by $\frac{\eta}{n}$ to ensure that the backdoor survives the averaging and the global model is replaced by X.

Constrain and Scale - The adversary trains the model with Loss function which rewards for model accuracy and penalizes for deviation from ”normal” indicator (evasion of anomaly detection).

$$L_{model} = \alpha L_{classification} + (1 - \alpha)L_{anomaly}$$

while $L_{classification}$ is function for both tasks main and backdoor.

Assumptions In the Constrain and Scale method the assumption is that the adversary knows the anomaly detection algorithm.
The evasion of anomaly detection depends on the anomaly detector. For sophisticated anomaly detector, a single malicious participant may not be able to achieve high accuracy on a backdoor task.

Defense The paper focuses on the defenses specifically designed for federated learning (with non i.i.d data).

- **Anomaly Detection**.
  - **Clustering** - Usage of K-Means to cluster participant's gradients isn't effective. With the assumption of non i.i.d data the anomaly detector may discard many interesting cases.

  - **Accuracy auditing** - Because of the fact that the adversary model L^{t+1} is scaled, it may drawdown the main accuracy. This may be detected and classified as anomaly. Splitting the malicious model between multiple adversary participants may help evade the detection. However, the high accuracy on the backdoor task and the evasion of anomaly detection is not always possible and depends on the backdoor task and the dataset.
  - **Cosine similarity** - Random vectors in high-dimensional space are orthogonal. This anomaly detector measures the cosine similarity between the submitted updates. Thus, it may detect and discard those updates which are similar to each other. This technic may defeat a case when a backdoor update splits among multiple participants. However the adversary can evade it by decomposing the backdoor model into orthogonal vectors.

- **Participant-level differential privacy**
  Participant-level differential privacy can mitigate the effectiveness of the backdoor attack, but on the examined collaborative training setup it also degraded the model's accuracy or convergence.

Table 10: Summary of attacks on Collaborative Training

Adversarial Behavior	Attack Defense	Differential Privacy	Sharing Fewer gradients	Anomaly Detection	Our Research Direction
Semi-Honest	Membership Inference	By definition DP prevents the success of these attacks. Application of DP may lead to tradeoff between privacy and accuracy. Application of DP may lead to prevention of model's conversions. This depends on number of participants and users in data set	Not Effective	Not examined	TBD
Semi-Honest	Property Inference (Passive Adversary)	In theory DP bounds the success of property inference attacks. Application of DP may lead to prevention of model's conversions. (Depending on number of participants and users in data set)	Not Effective	Not examined	Force to forget uncorrelated features by learning with particular objective function
Semi-Honest	Property Inference (Active Adversary)		Not Effective	Not examined	Anomaly detection
Malicious	Class Inference GAN (using network)	Not Effective	Not Effective	Not examined	Anomaly detection on a client side. (Accuracy monitoring)
Malicious	Backdoor injection into training data	Can reduce the effectiveness of the backdoor attack and the accuracy of the model.	Not examined	With assumption of i.i.d data most of the anomaly technics are ineffective. Accuracy auditing – may be effective with assumption of single participant adversary, but such assumption not always possible.	Anomaly detection on a client side

3.3.3 Privacy preserving collaborative training in PAPAYA

In the PAPAYA project we employ the approach presented in [93]. While the presented approach was implemented and evaluated on CNN, we implemented and evaluated the presented method on RNN. We used two datasets: the IMDb dataset [77] and The Reuters dataset [96].

Reuters Dataset - Collaborative Training setup

The Reuters dataset contains 11,228 newswires from Reuters news agency, labeled over 46 topics. It is intended to serve as a benchmark for topics classification. 8,982 are used for training, while the other 2,246 are used for validation. We performed collaborative training with three clients and a central server. We equally distributed the dataset between all clients. All clients performed 50 training epochs and shared only 0.3 of most largest gradients and downloaded the whole joint model from the central server. We compared the validation accuracy between the training on a local data only (in orange in Figure: 13) and the collaborative training (in aqua in Figure: 13).

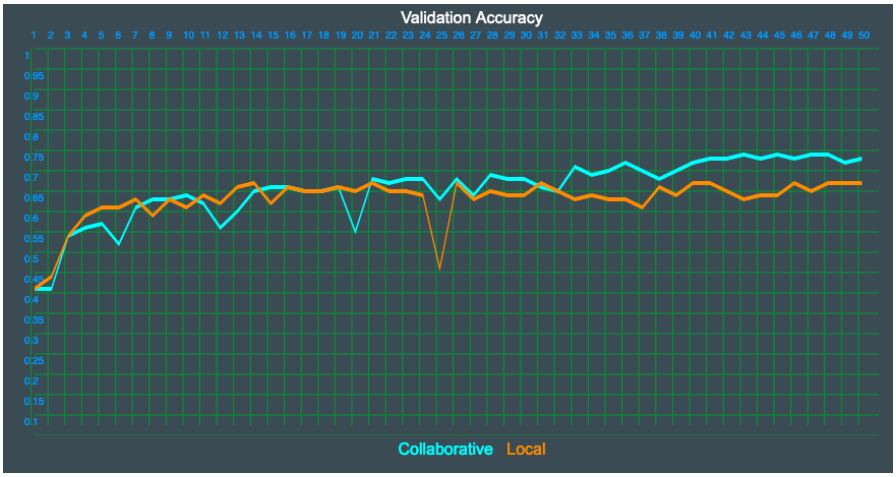


Figure 13: Validation accuracy of collaborative training vs. training on a local data.

We achieved similar results for all three clients. The figure 13 demonstrates that collaborative training achieves higher validation accuracy with all three clients. We figured out that the Collaborative Training method performs well also with RNN. From our experience with this approach we can expect that the method can be used for various NN architectures. However, as presented in 3.3.2.2 there are unresolved attacks that we plan to address in PAPAYA platform. The rest of this section is organized as follows: In Section 3.3.3.1, we describe the Algorithm as presented in [93]. Section 3.3.3.2 describes our research direction of defense against existing attacks on Collaborative Training.

In section 3.3.3.3 we describe the privacy preserving stress detection use case (UC2) in the healthcare umbrella defined in deliverable D2.1 [29], in which we will use Collaborative Training.

3.3.3.1 Privacy Preserving Collaborative Training Algorithm

We implemented the privacy preserving collaborative training algorithm as described in [93]

Algorithm 4: Privacy Preserving Collaborative Training Algorithm

Client Side: Each participant performs the following steps
repeat
for *each training ittteration* t **do**
Download a fraction of parameters from the central server and replace the corresponding parameters in a local model L_t Train a local model L on a private training data Compute local model changes (gradients)

$$\theta_t = L_t - L_{t-1}$$

Choose which gradient to upload to the central service
(The participant may apply DP noise to the selected gradients)

1. Upload θ_u fraction of most alternated gradients
2. Upload randomly sampled gradients which are above a certain threshold

end
until the desired accuracy is achieved, or when the accuracy is not improved anymore
The server performs the following steps:
On upload:

- Given a gradient vector θ, add the gradients to the global model
- $G = G + \theta$

On download

- Return the global model's parameters.

The gradients exchange may be performed without central server/model by using MPC between participants

3.3.3.2 Robustness against existing attack on Collaborative Training

We will research and examine the following direction:

- Anomaly Detection
  1. Anomaly detection on a participant side. We assume that when an adversary uploads the malicious model the honest participant may be able to detect it by observing the degradation of main task accuracy.

2. Anomaly detection on a central server. We assume that by observing the anomaly in gradients uploaded by participants, we may detect the adversary participant.

- Learning with precise Objective Function
  1. We will attempt to define the learning objective function in such way that will learn the main task accurately and will force the model to forget sensitive uncorrelated feature simultaneously.

3.3.3.3 Privacy Preserving Collaborative training for UC2

Our next step with respect to Collaborative Training will be to address requirements of UC2 [29]. In particular, the goal is to develop and train collaboratively NN for identification of upcoming stress and anxiety levels. The employees of several companies will wear a special MCI's T-shirt which measures and collects health-related data, in addition the employees will use MCI's app to label the collected data. The employees' data will be stored on companies trusted area. In order to achieve high prediction accuracy of NN and to preserve employees' privacy, each company will participate in privacy preserving collaborative training. Such training will be provided as a service on PAPAYA platform.

4 Privacy preserving Clustering

This section focuses on mechanisms for data clustering in a privacy-preserving fashion. Clustering is the task of partitioning a dataset into subsets (clusters) where in each cluster, the elements are "similar" according to some clustering models. k-means [78] and DBSCAN [42] are the most popular clustering algorithms nowadays and stand out for their simplicity, efficiency and accuracy. Both apply different clustering models. k-means' model is based on centroids: each cluster is represented by a single point computed as the mean of all the elements in that cluster. DBSCAN's model relies on the concept of density: clusters are defined as dense regions whose elements are connected to each other. Preserving privacy of users and confidentiality of data in these two algorithms has been first investigated through the lens of MPC. Nowadays, with the advent of homomorphic encryption, and especially FHE, more and more solutions have integrated this building block, essentially to avoid heavy interactions between data owners and the server. Before specifying the privacy challenges specific to clustering algorithms and analyzing the related works for privacy-preserving clustering, we outline below the two algorithms of k-means and DBSCAN. We also give an overview of TRACLUS [72], a clustering algorithm, based on DBSCAN, tailored to the peculiar case of trajectory data, and relevant for one of the PAPAYA use cases defined in deliverable D2.1 [29].

4.1 Definition

This section describes two popular clustering algorithms used in practice: k-means and DBSCAN. Because of this popularity, crypto researchers naturally focused on the privacy-preserving versions of these two algorithms. In addition, this section introduces the TRACLUS algorithm

that is used in the privacy-preserving mobility analytics use case (UC3) in the mobile and phone usage umbrella defined in deliverable D2.1 [29]. To give an overview of these three clustering algorithms, we will briefly explain the intuition behind their execution and based on illustrations, we will sketch the key steps of the algorithms.

4.1.1 k-means

k-means algorithm [78] is one of the simplest and one of the mostly used unsupervised learning algorithms that label a given data set into a certain number of clusters. The pseudocode of the algorithm is depicted in Algorithm 5.

Algorithm 5: Basic k-means algorithm

Input : Dataset D, k
Output: set of clusters $\{C_1, \ldots, C_k\}$
set of centroids $\{\mu_1, \ldots, \mu_k\}$
Initialisation: select k initial centroids
repeat
for *each point* $P \in D$ **do**
  Find the closest centroid μ_j
  Assign P to cluster C_j
end
for *each cluster* C_j **do**
  Update the centroid μ_j
end
until stop criterion

In k-means, the number of clusters is a priori fixed and defined by the parameter k. In a nutshell, the algorithm initially defines k centroids, one for each cluster. These centroids can be either randomly initialized or carefully computed so that the algorithm converges quickly to an optimal partition [17] (indeed, different locations for the initial centroids can yield different resulting clusters). It is commonly thought that placing them as much as possible far away from each other is a good choice. After initialization, the algorithm iteratively performs two main steps: (i) it assigns each data point to the closest centroid (according to a certain distance) and (ii) it updates the location of the centroids for each cluster. In step (i), an early clustering of the data is computed. However, after step (ii), because the location of the centroids has changed, the early clustering may be not optimal. This is the reason why the k-means algorithm operates several iterations, until the centroids no longer move or when the differences between the current centroids and the centroids in the last iteration is below a threshold. The Euclidean distance is a possible candidate for the metric to compute the nearest centroid in step (i); however, depending on the application, any other distance can be applied instead. The complexity of k-means is $\mathcal{O}(kndI)$ where k is the number of clusters, n the number of points, d the number of attributes (the dimension of the points in the dataset) and I the number of iterations until the algorithm stops.

Figure 14 depicts the several iterations of an application of k-means on a set of points (the black dots in the figure) for k=3:

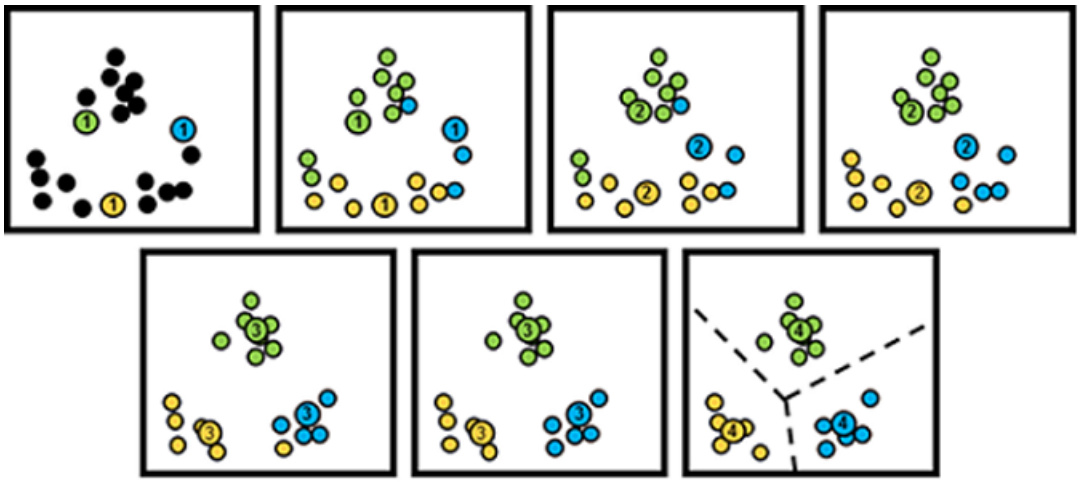


Figure 14: k-means algorithm

- Initialization: we (randomly) select 3 centroids defining 3 clusters (labelled as “green”, “blue” and “yellow”);
- Iteration 1: each dot is assigned a label (step (i))
- Iteration 2: we update the centroids (step (ii)) and reassign new label if necessary (step (i))
- Iteration 3: same as in iteration 2
- Iteration 4: centroids are updated again and no further move of the centroids or no further assignment is possible (the clustering algorithm converges).

As shown in the last panel in Figure 14, the result of k-means algorithm partitions the set of points into Voronoi cells.

4.1.2 DBSCAN

DBSCAN algorithm [42] is another very popular clustering algorithm and unlike k-means, it best mimics the human intuition of clustering a dataset. Besides, it does not need the parameter k (the number of clusters we are trying to find) and is able to identify clusters of any shape (whereas k-means only partitions the data in a Voronoi diagram). DBSCAN stands for Density-Based Spatial Clustering of Applications with Noise and, as its name implies, relies on the concept of density, that is, the number of neighbours of a point within a predefined radius. Because of this notion, some elements in the data may not be assigned to any cluster (hence, they are labeled as “noise” points).

The intuition behind DBSCAN is illustrated in Figure 15. Let us consider a dataset of points D. DBSCAN defines two parameters: a radius ϵ and a threshold Min_{pts}. For each point $P \in D$, the algorithm considers the neighbourhood around P as a sphere of radius ϵ and centered

in $P : N_\epsilon(P) = \{Q \in D | dist(P, Q) \leq \epsilon\}$. Then it checks whether $|N_\epsilon(P)| \geq Min_{pts}$. If so, P is labelled as a core point of a cluster C. Points in $N_\epsilon(P)$ are also added to C. The algorithm then expands cluster C by recursively checking if $|N_\epsilon(P')| \geq Min_{pts}$ for all $P' \in N_\epsilon(P)$. If the criterion on Min_{pts} is not verified for some point P, the point is ignored and the algorithm proceeds to a next element in the dataset. If the dataset contains n elements, the worst case complexity is $\mathcal{O}(n^2)$.

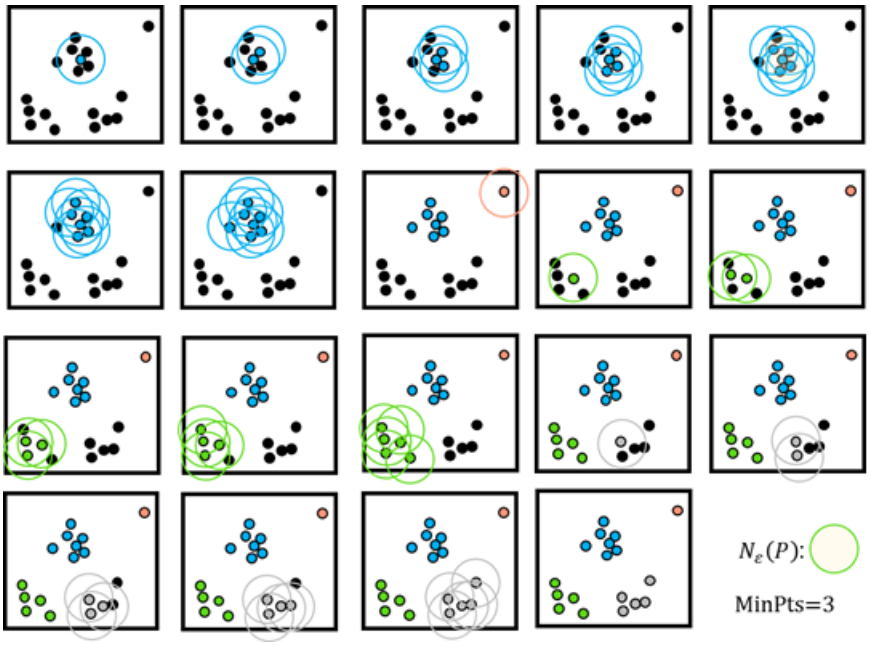


Figure 15: DBSCAN Intuition

More formally, the authors in [42] define clusters from two notions: density-reachability and density-connectivity. A point P is density-reachable from a point Q with respect to radius ϵ and threshold Min_{pts} if there is a chain of points $P_1, P_2, \ldots, P_m$ where $P_1 = Q$, $P_m = P$ and such that $P_{i+1} \in N_\epsilon(P_i)$ and $|N_\epsilon(P_i)| \geq Min_{pts}$. A point P is density-connected to a point Q with respect to ϵ and Min_{pts} if there is a point O such that both P and Q are density-reachable from O (with respect to ϵ and Min_{pts}). Hence a cluster (with respect to ϵ and Min_{pts}) is defined as a non-empty subset of D which satisfies the two conditions:

- Maximality: for all $P, Q \in D$, if $P \in C$ (a cluster) and Q is density-reachable from P with respect to ϵ and Min_{pts}, then $Q \in C$;
- Connectivity: for all $P, Q \in C$, the point P is density-connected to Q, with respect to ϵ and Min_{pts}.

A point $P \in D$ is a core point if $|N_\epsilon(P)| \geq Min_{pts}$. A point $P \in D$ is a border point if $|N_\epsilon(P)| < Min_{pts}$ but $P \in N_\epsilon(Q)$ where Q is a core point. A noise point is a point that is neither a core nor a border point.

4.1.3 TRACLUS

One of the PAPAYA use cases studies the mobility patterns of users of Orange's mobile network. Based on probe data, this use case aims at identifying people's trajectories between two areas of observation using a clustering algorithm. Trajectory data (sequences of spatial points together with a timestamp that are extracted from raw probe data) can be analysed thanks to a specific clustering algorithm named TRACLUS [72]. The high-level idea behind TRACLUS (for TRAjectory CLUStering) is to segment trajectories into small pieces (segments) which are then treated as points so as to apply the DBSCAN algorithm on these segments.

In order to find the clusters, the TRACLUS algorithm operates in three steps:

- Step 1: Trajectory segmentation. It basically consists in a (information-theoric) problem size reduction by approximating trajectories into small sets of segments that minimize the difference between the approximation and the actual trajectory.
- Step 2: Segment clustering. This step applies the DBSCAN algorithm to the set of segments. In this case, Min_{pts} is renamed Min_{lns} (for line segments). Besides, $dist$, the distance function used in DBSCAN, is a metric defined in [72] and depicted in Figure 16. The distance between two segments $dist(L_i, L_j)$ is defined as a weighted sum of three components $dist(L_i, L_j) = w_\perp.d_\perp + w_\parallel.d_\parallel + w_\theta.d_\theta$. The weights depend on the application but, for sake of simplicity, they can be set to 1. At the end of this phase, trajectory segments are assigned to several clusters.

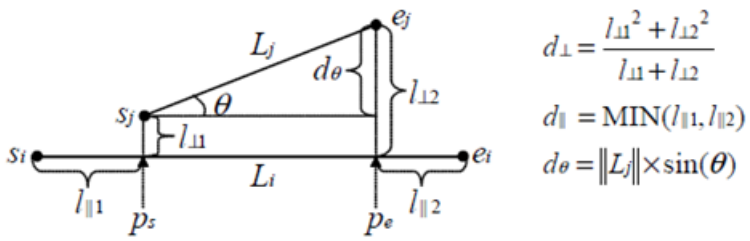


Figure 16: Line segment distance function in TRACLUS [72]

- Step 3: Representative trajectory definition. This final phase generates for each cluster identified in step 2 a representative trajectory that translates the main behavior of the trajectory segments belonging to the considered cluster.

4.2 Privacy Challenges & relevant PETs

In PAPAYA, we consider several privacy settings. Namely we distinguish the case where the data to be analysed is coming from a single data owner or from multiple data owners. We also examine for both cases the scenario where the analytics query stems from a third-party querier. In all these settings, the clustering task is outsourced to an *honest-but-curious* (cloud) server that owns the computational resources to perform the operations in an efficient way. Similarly, in the third-party querier scenario, the third party is also assumed to be honest-but-curious as

she should learn nothing but the results of clustering. Last but not least, data owners are also honest-but-curious, since they must not infer any information from other owners' data. Based on this threat model, the desirable properties for a privacy-preserving clustering algorithm are manifold. We use the following notations: O_i is the owner of dataset D_i, S designates the server while Q denotes the third-party querier.

- Data confidentiality: no information about the datasets D_i's should be leaked during the clustering operation to S and Q. Besides, owner O_i should not infer information about other datasets $D_{j\neq i}$.
- Cluster confidentiality: neither the O_i's nor server S should learn the result of the clustering. More specifically, the cluster characteristics such as the total number of clusters, the size of the clusters, the centroids (in the case of k-means), and the access patterns, that is the data assigned to each cluster, should remain unknown to all but the third party Q.
- Clustering function privacy: intermediate operations and results during the clustering task (such as the computation of distances dist in both k-means and DBSCAN) should remain blind to server S.

The first property, aka data confidentiality, is a mandatory requirement for a privacy-preserving data clustering algorithm. On the other hand, the two latter properties are more optional: some of the research work we analysed in this state of the art analysis, allow some amount of leakage to design efficient protocols in terms of communication and computation complexities.

The solutions we present in the next sections resort to two types of cryptographic building blocks: secure multiparty computation and (fully) homomorphic encryption or a combination of both. In MPC-based solutions, the challenge is to keep the number of interactions between parties at minimum. These solutions also often resort to other building blocks such as secret-sharing-based protocols or secure dot product algorithms to perform comparisons or to compute the distance between points (DBSCAN) or between points and cluster centroids (k-means). In (F)HE-based solutions, some considerations have to be taken into account to make the clustering algorithms compatible with encryption. For instance, both k-means and DBSCAN require the computation of a distance. Computing the Euclidean distance in the homomorphic domain is hard, since it involves the evaluation of a square root, which has not been implemented yet, to the best of our knowledge. Hence, either only the squared Euclidean distance is considered or a more FHE-friendly distance should be employed instead (such as the Manhattan distance, or L^1 distance), yielding possible accuracy loss. Another challenge with respect to FHE is the evaluation of comparisons (which appears both in k-means and DBSCAN). Even though recent advances in implementation of FHE algorithms show that FHE is close to be practical, homomorphic comparisons are still far from being efficient. Current FHE implementations (TFHE [24] , HElib [56], ...) require to perform bitwise (or digitwise) comparisons to compare two ciphertexts, which yields prohibitive costs. Finally, k-means involves a division step to update the centroids by computing the mean of the points assigned to them in the current iteration. This operation requires to divide a weighted sum of the points by the

number of points in a specific cluster (this number should remain unknown to the server according to our abovementioned properties). Performing the division without knowing the divisor led researchers to devise specific MPC-based protocols that enable several parties to jointly compute the result. Besides, division of ciphertexts in FHE is an operation that has not yet been implemented. Hence, FHE-based privacy-preserving k-means necessitates some tricks to avoid or approximate this division operation.

To summarize, the challenges in privacy-preserving k-means or DBSCAN lie in determining, in a secure way, whether two data points are close to each other. This involves the computation of a distance and a comparison of the obtained value with a (possibly secret) value. In k-means, performing secure division in order to update the centroids constitutes an additional challenge.

4.3 Related Work

Generally speaking, the problem of privacy-preserving clustering has first been tackled by the data mining community, in the setting where data is partitioned between several parties who collaborate to compute the clusters. The crypto research community then started to work on the topic, given the recent advances in cloud computing, FHE and MPC.

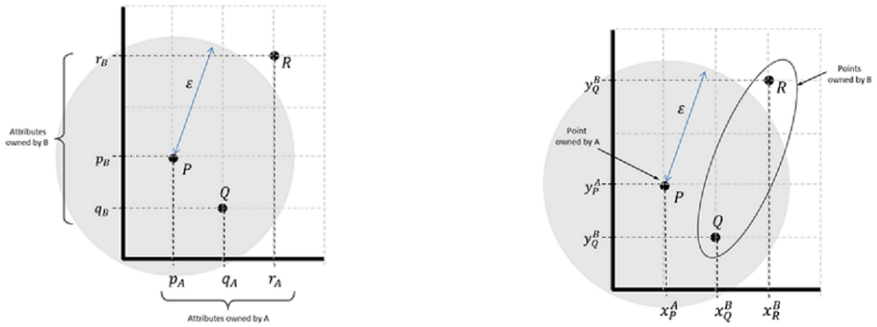
4.3.1 DBSCAN

For sake of simplicity, to describe the existing solutions for privacy-preserving DBSCAN clustering, we define the following operation (of traditional DBSCAN):

- $regionQuery(P, \epsilon)$: on input of a point P and radius ϵ, it returns the set of points $N_\epsilon(P)$, that is, all the points Q such that $dist(P, Q) \leq \epsilon$. This operation can be divided into two atomic operations, namely:
  - $dist(P, Q)$ which computes the (squared) Euclidean distance between two points P and Q;
  - $compare(d, \epsilon)$ which compares a value d (typically the distance computed via $dist$) and ϵ.

The research work we list in this section aims at adapting these operations into the privacy-preserving setting, such that they do not leak any information to unauthorized parties.

We review a total of five papers [100, 70, 75, 87, 6] that propose cryptographic solutions to the problem of privacy-preserving DBSCAN. To compare and analyse these solutions, we consider several parameters: the partitioning of the dataset (whether it is vertically or horizontally partitioned), the computation setting (whether the clustering algorithm is executed in a collaborative manner or is outsourced to a cloud server), the cryptographic techniques employed, the amount of offline (local) and online computation and the amount of leaked information. To illustrate our analysis, we consider a dataset D of three points $D = \{P, Q, R\}$ and a DBSCAN instance with parameters $\epsilon = 2,5$ and $Min_{pts} = 2$.



(a) Vertical partition

(b) Horizontal partition

Figure 17: Illustration for DBSCAN

Partitioning

In a multiple data owners setting, the considered dataset is often partitioned according to three types of division. Figure 17 illustrates two of these types: vertical partition (Figure 17a) and horizontal partition (Figure 17b).

In vertical partitioning, each user possesses all the data records but only a subset of their attributes. In other terms, each record is distributed among the parties. This approach is depicted in Figure 17a: the considered dataset D contains three points $D = \{P, Q, R\}$ and is vertically partitioned between Alice and Bob. Alice (resp. Bob) stores the x-axis (resp y-axis) attributes $D_A = \{p_A, q_A, r_A\}$ (resp. $D_B = \{p_B, q_B, r_B\}$). The vertical partitioning setting was considered in [100, 70, 75].

In horizontal partitioning, each user possesses a subset of the data records but all their attributes. As depicted in Figure 17b, Alice stores $\{P = (x_P^A, y_P^A)\}$ whereas Bob stores $\{Q = (x_Q^B, y_Q^B), R = (x_R^B, y_R^B)\}$. the horizontal partitioning setting was studied in [70, 75, 87, 6].

The third approach, arbitrary partitioning , combines the two previous approaches and was considered in [75].

Collaborative vs. outsourced DBSCAN

Two main approaches are considered in the papers we reviewed. The first approach deals with a joint execution of the DBSCAN algorithm [100, 70, 75]. In such a setting (mainly involving two parties), each party holds a subset of the dataset (partitioned according to one of the splitting approaches previously described) and participates in the execution of the privacy-enhanced $regionQuery$ functionality. The data of each party must be kept private to the others.

In the second approach, considered in [87, 6], algorithm $regionQuery$ is outsourced to an untrusted third party such as a cloud service provider. Several data owners store their data and delegate the DBSCAN algorithm to this cloud provider. As the cloud is deemed untrusted, it should learn no information about the data and the resulting clusters.

Cryptographic building blocks

Most of the studied related work [100, 70, 75] rely on MPC techniques for both the functionalities $dist$ and $compare$. Some of these MPC techniques rely on Paillier's additively homomorphic encryption scheme [84]. The two remaining articles [87, 6] are only invoking homomorphic encryption solutions (either Paillier or an exotic scheme [73]). The choice of either MPC or HE induces a tradeoff between offline (local) computations and online computations.

In the MPC-based techniques, the $dist$ functionality is computed in privacy-preserving manner in two phases. In the first "offline" phase, parties compute the local distances of the data records (or attributes) they own. In the second "online" phase, parties jointly compute the global distances (between all the data records with all their attributes) using solutions such as a secure scalar product protocol [70], a secure two-party multiplication algorithm [75] in the case of horizontal partition or Yao's millionaire protocol [75] in the case of vertical partition. On the other hand, the function $compare$ is evaluated between parties using Yao's protocol [106] in all the three MPC-based schemes [100, 70, 75].

MPC-based solutions. In [100], where the data is vertically partitioned, each party locally and independently executes $regionQuery$ over their local data to find the local ϵ-neighbourhood of a point P in the dataset. Then the global ϵ-neighbourhood can be computed as the intersection of the local ones. To preserve the privacy of the local ϵ-neighbourhoods, the solution resorts to a secure set intersection protocol based on RSA encryption [89], which outputs the intersection in clear. However, the resulting intersection may contain points that are local but not global neighbours. Hence, to exclude those points, the parties engage in a Yao's millionaire protocol [106] to determine whether the global distance (the sum of the local distance) is below ϵ. Let us consider the example in Figure 17a. Alice locally computes $N_\epsilon(P)^A = \{Q, R\}$. Indeed, $dist(p_A, q_A) = 1 \leq \epsilon$ and $dist(p_A, r_A) = 2 \leq \epsilon$. Similarly Bob locally computes $N_\epsilon(P)^B = \{Q, R\}$. Then Alice and Bob use the secure set intersection to compute $I = N_\epsilon(P)^A \cap N_\epsilon(P)^B = \{Q, R\}$. Now, to exclude those points which are local but not global neighbours, for each point $P' \in I$, Alice and Bob engage in Yao's millionaire protocol with inputs $dist^2(p_A, p'_A)$ (only known by Alice) and $\epsilon^2 - dist^2(p_B, p'_B)$ (only known by Bob[11]). In our example, R will not be considered as a global neighbour of P since $(p_A - r_A)^2 = 4$ and $(p_B - r_B)^2 = 4$ and $(p_A - r_A)^2 \geq \epsilon^2 - (p_B - r_B)^2$. Hence $N_\epsilon(P) = \{Q\}$.

The authors in [70] follow the same approach as in [100]. Each party computes the local distances from a point P. For instance, if we consider the example in Figure 17a for vertically partitioned datasets, Alice locally computes the distances between p_A and q_A and between p_A and r_A. Bob similarly performs the same computations. To determine the global ϵ-neighbourhood of P, Alice and Bob jointly compute the distances global $dist(P, Q)$ and $dist(P, R)$ using their local distances and compare it to ϵ (or ϵ^2, to get rid of the square root operation in the Euclidean distance). The solution resorts to a secure scalar product protocol [51] based on Paillier's encryption [84]. The result of the scalar product is then passed as input to a Yao's millionaire protocol to test whether $dist(P, \ldots) \leq \epsilon$. To be precise, to compute $dist(P, Q)$, the secure scalar product takes as inputs the vector $A = (\frac{(p_A - q_A)^2}{\epsilon^2}, 1)$ (owned

[11] Distances and ϵ are squared to avoid computing the square root.

by Alice) and $B = (\alpha, (\frac{\alpha(p_B - q_B)^2}{\epsilon^2})$ with α being random (owned by Bob). Only Alice gets $\beta = \alpha.\frac{(p_A - q_A)^2 + (p_B - q_B)^2}{\epsilon^2}$ and both Alice and Bob execute the Yao protocol with inputs α and β to decide whether $\alpha \leq \beta$.

The authors in [75] also follow the same approach as in [70]. They focus on two scenarios where the data is either vertically or horizontally partitioned. For the vertically-partitioned case, the authors propose a similar but simplified protocol to [70]. They observe that to compute $(dist(P,Q)^2)$ of two points as in Figure 17a, it suffices for Alice to compute $d_A = (p_A - q_A)^2$ and for Bob to compute $\epsilon^2 - d_B = \epsilon^2 - (p_B - q_B)^2$. Then they both use Yao's protocol on these two values to decide whether $d_A \leq \epsilon^2 - d_B$, which allows to learn whether $dist(P,Q) \leq \epsilon$.

In the case of horizontally-partitioned data, the authors in [75] propose to compute the distance in a two-party fashion using a secure two-party multiplication protocol based on Paillier encryption [84] which on inputs of a (owned by Alice) and b (held by Bob), outputs to Alice the value $u = ab + v$, where v is a random number owned by Bob. Hence, if we consider the example from Figure 17b, $(dist(P,Q)^2)$ can be computed as $(dist(P,Q))^2 = d_A + d_B - 2d_{AB}$ where $d_A = (x_P^A)^2 + (y_P^A)^2$, $d_B = (x_Q^B)^2 + (y_Q^B)^2$ and $d_{AB} = x_P^A \cdot x_Q^B + y_P^A \cdot y_Q^B$. Alice (resp. Bob) can compute d_A (resp. d_B). To compute d_{AB}, they both engage in the secure multiplication protocol for each of the terms of the sum. Afterwards, as in the vertically-partition case, Alice and Bob execute the Yao's millionaire protocol to decide whether $dist(P,Q) \leq \epsilon$.

FHE-based solutions. The paper [87] is only an abstract paper, so many details are not available. Unlike the research work described so far, the solution presented in [87] focuses on the scenario where multiple data owners jointly compute DBSCAN algorithm by outsourcing the computational overheard to a cloud server $\mathcal{S}$. To allow $\mathcal{S}$ to execute operation $regionQuery$, the data owners encrypt their data using an FHE algorithm that has the special property of key-homomorphism: if for all i, party P_i has key pair (sk_i, pk_i), then this property allows to combine the pk_i's into a combined public key pk and simultaneously combine the corresponding secret key into a combined secret key sk. In other terms, sk_i is a secret share of the common secret sk. Hence decryption of a ciphertext encrypted with pk can be jointly performed by the data owners using their own sk_i. The authors in [87] leverage the construction in [76] for such an FHE algorithm. Thus, in the privacy-preserving DBSCAN solution introduced in [87], each data owner encrypts her data point using pk and sends it to S. In turn, the server selects any (encrypted) point P and executes $regionQuery$ as follows: for each other (encrypted) point Q, S computes the encrypted Euclidean distance $d = Enc_{pk}(dist(P,Q))$ thanks to the homomorphic property of the encryption algorithm. It also computes $d' = d + Enc_{pk}(-\epsilon) = Enc_{pk}(dist(P,Q) - \epsilon)$. Server $\mathcal{S}$ sends d' to the data owners who jointly run the decryption algorithm and test whether $dist(P,Q) \leq \epsilon$. The result of this test is sent to $\mathcal{S}$ who pursues the rest of the DBSCAN algorithm. Hence, in summary, the data owners' local computational cost consists of encryption, decryption and comparison $(compare)$ operations, while server's computational cost only consists of operation $dist$.

On the other hand, the work described in [6] is the only one so far that allows a cloud server $\mathcal{S}$ to perform the DBSCAN algorithm without the participation of data owners in the execution of $regionQuery$ to test whether $dist(P,Q) \leq \epsilon$. To enable such an interesting property, the authors propose to encode and encrypt all distances in special data structures called Chain

Distance Matrices (CDM), generated by individual data owners. Prior to the CDM generation, the dataset D is ordered according to a specific attribute. A CDM is then a 2D-matrix that stores the distances between each attribute value of a point of the data and the corresponding attribute in the following point. Each data owner generates her CDM with the attributes she owns and encrypts it using a special encryption algorithm called MUOPE (for Multi User Order Preserving Encryption) to obtain a secure CDM (SCDM). The MUOPE proposed in [6] is based on Paillier encryption scheme [84] and gives access to the server to the ordering of distances between records in the SCDM. Then all SCDMs (from each data owner) are combined into a super SCDM using a "binding" operation (a simple concatenation of SCDMs in the case of vertically-partitioned data or a more tricky combination in the case of horizontal partitions). The function $regionQuery$ is then applied to the resulting super SCDM. The authors in [6] suggest a similarity function Sim that allows to compute the MUOPE-encrypted distance between points from the encrypted distances stored in the super SCDM. This encrypted distance is then compared with ϵ', the MUOPE-encrypted value of ϵ. Since MUOPE preserves order, the comparison of the encrypted values gives the expected test $dist(P, Q) \leq \epsilon$.

Leakage

The solutions described above all allow a certain amount of information leakage. In particular, [100] leaks each local ϵ-neighbourhood's cardinality, the intersection cardinality and the output of Yao's protocol to all parties. Similarly, the algorithm proposed in [70] discloses the final clusters' sizes whereas in [75] both protocols for vertically and horizontally-partitioned datasets reveals the size of ϵ-neighbourhoods in each step of the DBSCAN algorithm. In [87] and [6], the server $\mathcal{S}$ also learns the size of final clusters, but does not learn the content of each cluster.

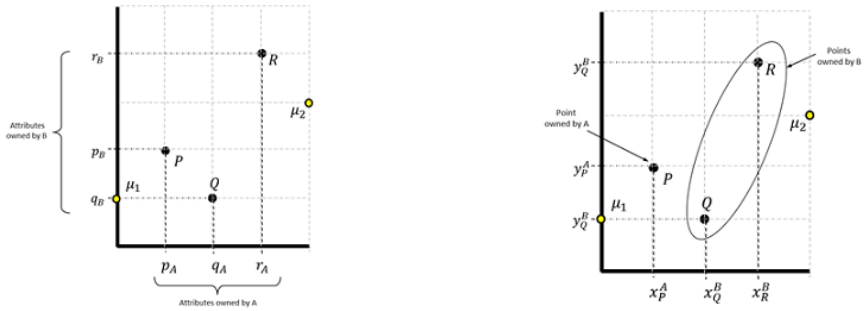
Summary

The research on privacy-preserving DBSCAN algorithm is relatively poor, with solutions relying either on MPC or FHE with a heavy computation load at the users. Nevertheless, this existing work shows that privacy-preserving DBSCAN is possible and that there is room for improvement.

4.3.2 k-means

For the sake of clarity, we define several operations that are key steps in the execution of the k-means algorithm. For each of these operations, the protocols described in this literature review suggest tailored solutions to allow privacy-preserving k-means.

- $Initialisation(k)$: this operation generates the k initial centroids.
- $FindClosestCluster(P, \{\mu_1, \ldots, \mu_k\})$: it takes as inputs a point $P \in D$ and the current list of k centroids and outputs the index $i \in [1, k]$ for which $dist(P, \mu_i) \leq dist(P, \mu_{j \neq i})$ for all $j \in [1, k]$. In other terms, $dist(P, \mu_i)$ is the minimum among all the k centroids. Hence, we also define the two atomic operations: $FCCdist$ which computes the distance between a point and a centroid and $FCCmin$ which finds the minimum.



(a) Vertical partition (b) Horizontal partition

Figure 18: Illustration for k-means

- $UpdateCentroids(D, L)$: this function takes as inputs the dataset D and the current cluster labels L (that is, point D_j is assigned to cluster $L_j \in [1, k]$ and $\|L\| = \|D\|$). It computes the new centroids as $\mu_i = \frac{\text{“sum” of points in cluster i}}{\text{number of points in cluster i}}$ for all $i \in [1, k]$. This functionality is then split into two atomic operations: $UC - sum$ which computes the sum of points in the same cluster and $UCdiv$ which computes the division.
- $CkeckStop(\{\mu_1, \ldots, \mu_k\}, \{\delta_1, \ldots, \delta_k\}, \theta)$: this operation checks whether the criterion to stop the k-means algorithm is verified. It computes the difference between the current list of centroids (the μ_i's) and the list of centroids of the previous iteration (the δ_i's) and checks whether it is below θ (θ can be equal to 0).

We review a total of nine papers [98, 35, 62, 63, 92, 20, 97, 88, 65] that tackled the problem of privacy-preserving k-means with cryptographic techniques. For illustration purposes, we consider a k-means instance with parameter $k = 2$ (see Figure 18). Two centroids (μ_1, μ_2) are arbitrarily set to $\mu_1 = (0, 1)$ and $\mu_2 = (4, 3)$.

As in section 4.3.1, we compare these solutions using several criterion.

Partitioning

As in the case of DBSCAN, the solutions outlined in this section consider different data partitioning models. Figure 18a depicts a dataset D which contains three points $D = \{P, Q, R\}$ that are vertically partitioned between Alice and Bob. The research work described in [98, 35, 92] propose solutions for this setting. In Figure 18b, D is horizontally partitioned between Alice and Bob. This environment has been studied in [63, 92, 88, 65]. Finally, the rest of the papers we reviewed [62, 20, 97] focuses on the scenario of arbitrary partitioned data, which consists in a combination of vertical and horizontal splitting of the dataset.

Collaborative vs. outsourced k-means

Privacy-preserving collaborative k-means is considered in [98, 35, 62, 63, 92, 20]. In this approach, several parties cooperatively execute the k-means algorithm on their joint datasets. Each party holds a subset of the data, partitioned according to one of the splitting approaches

previously described. The data of each party must not be revealed to other parties, in order to preserve confidentiality of the data and privacy of the users. Additionally, the data assignment to clusters should also remain private.

The second approach adopted by the authors in [97, 88, 65] delegates the computational burden of k-means over a dataset owned by a single or multiple parties to an untrusted cloud server. In this case, the cloud should not learn any information about the data and the clusters.

Cryptographic building blocks

Most of the solutions studied in this literature review [98, 35, 62, 63, 92, 20, 97, 88] resort to MPC techniques such as Yao's protocol and special secure function bricks (for addition, multiplication, division). Some of these MPC techniques invoke Paillier's homomorphic scheme. Only one work proposes a pure FHE-based solution [65]. One possible explanation of such a discrepancy is that, in the current status of research and implementation of FHE solutions, the comparison operation is not FHE-friendly since it relies on encoding the plaintext data into binary encoding and comparing the data bit by bit, which may lead to prohibitive computation.

As it was the case for DBSCAN algorithm, the MPC-based techniques operate the privacy-preserving k-means in two phases: an offline local phase and an online phase.

MPC-based solutions. To compare all the solutions, we consider each of the four functionalities defined above.

$Initialisation$**.** Not all the papers [98, 35, 63, 92] address the problem of executing the $Initialisation$ functionality, which generates the k initial cluster centroids. In [62, 88, 97, 65], the k initial centroids are data records randomly selected from the dataset. More particularly, in [62], which considers the collaborative setting, the k initial centroids are either the k first data points, or chosen randomly with randomness introduced by the parties. In [88], where the k-means algorithm is outsourced to a cloud provider consisting of two servers S_1 and S_2, server S_1 selects the k initial centroids randomly from the data encrypted by the parties (using Paillier's HE). In [97], also in the outsourced setting, where the cloud consists in several servers, the initial centroids are also selected at random and then secret-shared among the servers. The authors in [65] propose either to select the initial k centroids at random from the data records or k random values (in the range of the data). Finally, the authors in [20] propose a solution for two-party random value selection, based on secret sharing, that is used in the $Initialisation$ step.

$FindClosestCluster$**.** This routine, and its two subroutines $FCCdist$ and $FCCmin$, have received much more attention, and a variety of solutions has been proposed to make this functionality privacy-aware.

In the scenario of **horizontally-partitioned** data, in the collaborative model, considered in [63, 92], the entire $FindClosestCluster$ operation can be executed locally and independently by each party, since the data they own have all their attributes, and hence they can compute the distance to the centroids ($FCCdist$) and find the centroid that minimizes this distance ($FCCmin$). In [88], the data is also horizontally partitioned, but unlike [63, 92], the data is encrypted using Pailler's encryption scheme under server S_2's public key and outsourced to a server (S_1). To execute a private version of $FindClosestCluster$, the authors design a secure

two-party squared order-preserving Euclidean Distance and a secure two-party minimum-out-of-k-numbers protocols which manipulate ciphertexts encrypted via the Paillier homomorphic encryption scheme. These two subprotocols are run by the two cloud servers S_1 and S_2.

In the case of **vertically-partitioned data**, as in [98, 35, 92], $FCCdist$ can be computed locally and independently based on the attributes each party possesses. Indeed, each of the parties can compute a part of the distances between the points and the centroids ("local" distances). Note that the data holders do not need to know the centroids, but only their projection to the set of attributes they own. For example, if we consider the example in Figure 18a, Alice and Bob can compute the local distances of point P to centroid μ_1 as $dist^A(P, \mu_1) = \mu_{1A} - p_A$ (resp. $dist^B(P, \mu_1) = \mu_{1B} - p_B$). However, as these "local" distances contain private information, they cannot be revealed to all parties to compute the global distance normally output by $FCCdist$. Hence the authors in [98] suggest to use a secure add-and-permute protocol based on the algorithm proposed by Du and Atallah [37], which is based on Paillier's homomorphic encryption scheme [84]. In [35], the secure add-and-permute protocol relies on an additive secret-sharing scheme instead. Then , to securely find the cluster centroid that minimizes the distance, or in other words, to securely compare distance values, the authors in [98, 35] resort to a secure comparison protocol based on Yao's circuit evaluation for comparisons [107]. In [92], the authors suggest to use the secure sum algorithm provided in [30] to both $FCCdist$ and $FCCmin$.

In the case of **arbitrary-partitioned data**, as in the work in [62, 20, 97], several strategies are applied. The key idea of the solution presented in [62] is to compute random shares of the centroids: let μ_j^A (resp. μ_j^B) be Alice's (resp. Bob's) share of μ_j the centroid of the cluster indexed by $j \in [1, k]$, then $\mu_j = \mu_j^A + \mu_j^B$. Hence the idea is to run $FindClosestCluster$ on the random shares. In $FindClosestCluster$, to compute for each point P and for each cluster centroid μ_j the shares α^A and α^B of the distance $(dist(P, \mu_j))^2 = \alpha^A + \alpha^B$, the authors suggest to use the secure scalar product protocol described in [51]. Then to find the index j that minimizes $dist(P, \mu_j)$ among all the centroids (i.e. executing $FCCmin$), they propose to resort to Yao's protocol [107]. At the end of the functionality, Alice and Bob learn their respective shares of the centroids and nothing else. The authors in [20] extend this solution, except that instead of invoking Yao's protocol, they devise a XOR-based find-the-minimum secure protocol $FCCmin$.

In [97], the authors consider several parties that share an arbitrarily-partitioned data and which outsource the k-means clustering algorithm to a cloud that consists of $R > 2$ non-colluding servers. Following the same idea as the previously-described solutions, the solution presented in [97] resort to random shares of the data. Nevertheless, this work innovates by choosing a Chinese Remainder Theorem (CRT) based secret sharing. Accordingly, the authors define a "shatter" function that secret-shares a data point x into R parts $x_1, \ldots, x_R$ as $x_i = \phi(x, p_i) = x.S + \nu \bmod p_i$, where S is a scale factor, ν a random number and p_i a prime. To reconstruct the point, the authors define a "merge" function that can recover x by solving a system of congruence: $\Psi = (x_i, p_i) = \frac{CRT(x_i, p_i)}{S}$. Given these two functions, k-means can be applied to the CRT-based shares stored at the R servers. In the $FindClosestCluster$ functionality, the servers can independently compute the secret shares of the distances of any point P to the k centroids. On the other hand, the R servers must jointly execute $FCCmin$. To do

so, each of the servers computes the share of the difference between two distances, applies a layer of randomization (to prevent the leakage of the distance information) and send the obtained result to an untrusted server (called the thresholder) for reconstruction and comparison. This thresholder invokes the "merge" function to reconstruct the randomized difference of two distances, compares it with 0 and sends the result to the R servers, who can then find the minimum distance.

$UpdateCentroids$**.** In the case of **vertically-partitioned** data, in the collaborative model, the functionality $UpdateCentroids$ can be executed locally and independently, because each party has all the data records and hence can compute the components of the new centroids based on the attributes they own. If we consider the example of Figure 18a, Alice can compute the update centroid $\mu'_{1A} = (p_A + q_A)/2$.

In the case of **horizontally-partitioned** data, more elaborated computations are needed, since evaluating $UCsum$ and $UCdiv$ is not straighforward without having access to all the data points in the shared dataset. Practically, if we consider the example of Figure 18b, and if the k-means algorithm wants to update centroid μ_1, based on points P and Q, then Alice and Bob must cooperate to compute the barycentre of P and Q, without Alice having access to Q and without Bob learning P. In [63], updating the centroids requires to compute a division where the divisor is shared between two parties, hence unknown to both parties. Therefore, the authors in [63] introduce the weighted average problem (WAP), that aims at computing in a privacy-preserving manner the division $\frac{a+b}{n+m}$, where Alice only knows the private values a and n and Bob only has the private numbers b and m. Both parties learn the result of the division. To solve WAP, the authors propose two solutions. The first solution relies on oblivious polynomial evaluation (OPEv). An OPEv protocol such as the proposed by Naor and Pinkas [82] solves the following problem: Alice has a polynomial P, Bob has a point x and wants to compute $P(x)$ such that Alice learns nothing. Then an OPEv solution can solve the following other problem, denoted the private rational polynomial evaluation problem (PRPE): Alice has polynomials P, Q, Bob has points α, β and both parties want to compute $\frac{P(\alpha)}{Q(\beta)}$ in a private matter. This problem can be solved as follows: Alice blinds her polynomials with a random element z as zP and zQ. Bob computes $zP(\alpha)$ and $zQ(\beta)$ by invoking the OPEv and sends $\frac{zP(\alpha)}{zQ(\beta)} = \frac{P(\alpha)}{Q(\beta)}$ to Alice. Finally, to solve WAP, Alice defines $P(X) = X + a$ and $Q(X) = X + n$ whereas Bob sets $\alpha = b$ and $\beta = m$. The second solution for WAP is based on a homomorphic encryption scheme (E) such as Paillier's [84]: Alice encrypts $a' = E(a)$ and $n' = E(n)$ and sends them to Bob. Bob generates a random message z and encrypts $b' = E(zb)$ and $m' = E(zm)$. Thanks to the homomorphism of E, Bob can compute $c_1 = a'z * b' = E(za + zb)$ and $c_2 = n'z * m' = E(zn + zm)$ and sends them to Alice. Alice decrypts c_1 and c_2, obtains $z(a + b)$ and $z(n + m)$ and finally compute $\frac{a+b}{n+m}$. The authors in [92] extends the work of [63] to the scenario where more than two parties can be involved in the computation of the new centroid. They propose a new secure multiparty division protocol, based on a secure multiparty addition algorithm [91, 105], itself based on secure dot product scheme. In a nutshell, the secure multiparty addition algorithm converts a sum of private values into a product of random shares: $\sum_{i=1}^{N} x_i = \prod_{i=1}^{N} r_i$, without revealing the x_i's and the r_i's. If $N = 2$, Alice randomly select r_A and creates the vector $u_A = (\frac{x_A}{r_A}, \frac{1}{r_A})$ whereas Bob creates $u_B = (1, x_B)$. Alice and Bob run a secure dot product protocol so that Bob obtains $r_B = u_A \cdot u_B = \frac{x_A + x_B}{r_A}$.

Hence, $r_A r_B = x_A + x_B$. Then to perform the secure division $\frac{x_A+x_B}{a+b}$ where a is owned by Alice and b by Bob, they both apply the above secure addition protocol to separately compute r_A, r_B, s_A, s_B such that $x_A + x_B = r_A r_B$ and $a + b = s_A s_B$. Then Alice receives $\frac{r_B}{s_B}$ from Bob and computes $\frac{(r_A/s_A)}{(r_B/s_B)}$.

In the **arbitrary partition** setting, several approaches are followed. In [62], the authors simply suggest to invoke Yao's protocol to perform the secure $UCdiv$. Bunn and Ostrovsky [20] replace Yao with a dedicated secure two-party division algorithm, which is not detailed here because it is hard to summarize. Finally, in the work [97], which, we recall, adopts the outsourcing model to a cloud of R servers, each server can independently compute $UCsum$, the sum of the secret shares of the points assigned to each cluster. However, they need to collaborate to execute $UCdiv$, i.e. the division of this sum with the number of data points belonging to the considered cluster. Hence these servers together with the thresholder (see above) apply a privacy-preserving division protocol introduced in [97] which, similarly to the computation of $FCCmin$, randomizes the shares of the sum computed by each of the R servers so that the thresholder reconstructs the (randomized) sum from the shares and perform the required division to update the centroids.

$CheckStop$**.** Yao's protocol is used in several solutions [98, 35, 62] to check the stopping criterion of the privacy-preserving k-means algorithm. The work in [20] and [88] design dedicated secure two-party evaluation of termination condition. Finally, the solutions in [63, 92, 97] do not present a technique for a private $CheckStop$ functionality.

FHE-based solutions. Only Jäschke and Armknecht's work [65] proposes a pure FHE-based solution to privacy-preserving k-means clustering. Their solution relies on the choice of TFHE [24] as the underlying FHE algorithm. This choice is justified by the fact that TFHE since the scheme allows to perform comparison of ciphertexts (thanks to a custom comparison boolean circuit). The authors actually propose three approaches, that are gradually more efficient, by means of a series of approximations. In the first approach, the authors implement the exact k-means algorithm on encrypted data. Instead of using the Euclidean distance in the $FCCdist$ functionality, they resort to the L_1-norm (or Manhattan distance) defined as $dist(P, Q) = |x_p - x_q| + |y_p - y_q|$, without degrading accuracy of the algorithm. $FCCmin$ is implemented via the binary encoding of TFHE and then uses encrypted bit additions and multiplications. To operate algorithm $UpdateCentroids$ in the encrypted domain, the authors devise a fractional encoding which encodes a (bitwise-encoded) number a into two values (a_n, a_d), such that $a_n = \lfloor a \cdot a_d \rceil$, and which allows to divide two numbers only using the homomorphic multiplication of TFHE. While theoretically possible, this encrypted division incurs prohibitive costs. The second approach tries to solve this pitfall by avoid the division of ciphertexts in the $UCdiv$ functionality. The authors observe that while FHE schemes do not allow homomorphic division (i.e. the division of ciphertexts), they nevertheless support the division of a ciphertext by a constant. Hence, they introduce a trick such that $UCdiv$ requires only division by a constant. The last approach introduces an approximation of the comparison operations in $FCCmin$, such that it runs faster than in the first approach.

Leakage

All the solutions described in this section allow a certain amount of leakage, especially the results of intermediate operations at the end of each iteration of the k-means algorithm. For instance, in the solution presented in [92] and [63], the parties know the intermediate centroids. Having such a knowledge can be problematic since information about other parties' data may be derived from the centroids.

Summary

In comparison to the work on privacy-preserving DBSCAN, the k-means algorithm has received more attention by the cryptography research community. The existing work first focused on the scenario of collaboration of multiple parties to jointly compute the k clusters. This scenario naturally leads to the choice of MPC techniques to preserve privacy of the users and confidentiality of the data while executing the k-means algorithm. Nevertheless, even in the case where the computational burden of the clustering algorithm is offloaded to a third-party cloud server, MPC techniques are preferred over homomorphic encryption since the operations in k-means involve comparison operations that are today still not well supported by homomorphic encryption schemes. However, as interest in homomorphic encryption has been recently growing and its performance improvement is fast, we can conjecture that more and more solutions leveraging this encryption technique will be designed in the near future. To solve the pitfall with respect to comparison operations, homomorphic encryption will have to be combined with MPC or to involve the data owner(s) in the computation of the $FindClosestCluster$ functionality.

4.4 Privacy preserving trajectory clustering based on PHE (preliminary design)

In this subsection, we introduce a preliminary design of a privacy preserving trajectory clustering based on the additively homomorphic Paillier encryption scheme [84]. As introduced in Section 4.1, the goal of trajectory clustering is to find similar traveling routes in a set of trajectories. This requires to cluster trajectories based on location information. Since this can be a computationally expensive operation, we consider a scenario where a data owner having collected multiple trajectories, wishes to delegate the execution of the actual trajectory clustering algorithm (TRACLUS) to an untrusted but powerful server such as the PAPAYA platform. Therefore, all trajectory information need to be protected before their outsourcing to the cloud server. TRACLUS mainly consists of a number of distance calculations. In order to perform these distance computations with the Paillier encryption scheme, some operations need to be either transformed or approximated. Indeed, the Paillier encryption scheme supports only the addition of encrypted numbers and scalar multiplication whereby the scalar is in cleartext. We, therefore, propose to simplify the original distance metrics defined in Section 4.1 and replace them with the simple squared Euclidean distance as in Equation 4 since the positional difference and partial directional difference are included in the Euclidean distance. The computation of the Euclidean distance involves some multiplication operations that cannot be inherently supported by Paillier.

$$ED^2(p,q) = \sum_{r=1}^{N} p_r^2 + 2\prod_{r=1}^{N} p_r \cdot q_r + \sum_{r=1}^{N} q_r^2 \tag{4}$$

where p_i and q_i, $i = 1, \ldots, N$ are the components of p and q.

Therefore, we propose to support multiplications with a newly designed protocol executed between the cloud server and an additional third party that we name EDServer. The newly proposed solution is illustrated in figure 19.

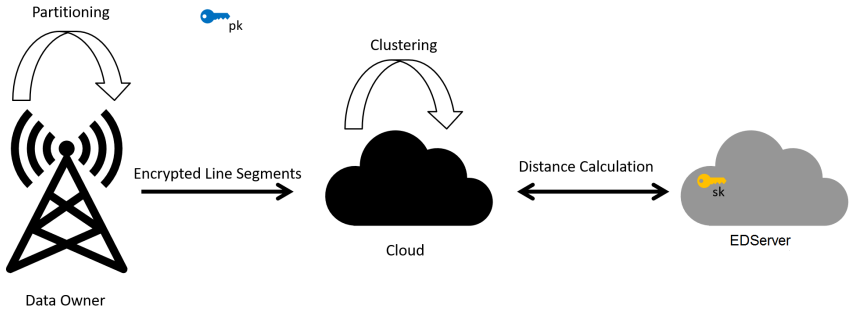


Figure 19: The preliminary design setting of privacy-preserving TRACLUS

In this scenario, the data owner encrypts all line segments with the Paillier encryption scheme and sends them to the cloud. The encryption is performed using the public key of EDServer. Then the cloud starts to compute Euclidean distances between these segments. Whenever multiplication between two encrypted numbers are needed, the cloud interacts with EDServer as follows: Let $E(a)$ and $E(b)$ be two encrypted numbers using EDServer's public key. In order to prevent EDServer from accessing the cleartext numbers, the cloud will perform a scalar multiplication with some random numbers and send the results to EDServer. When these newly computed values are received, EDServer can decrypt them, perform the multiplication and re-encrypt the result. The cloud server can easily recover $E(ab)$ thanks to its knowledge of the random numbers, $E(a)$ and $E(b)$. After having calculated all distances between one line segment and all other not clustered line segments, the cloud sends an array with all encrypted distances to EDServer who can decrypt and determine which distances are smaller than a given threshold ϵ. As EDServer does not know which distance belongs to which pair of line segments, it only learns the number of not clustered line segments and how many elements belong to one cluster. We are currently finalizing the design of the solution. More details will be provided in deliverable D3.3.

5 Privacy preserving Counting

In this section, we introduce basic privacy preserving counting mechanisms. The main idea is to encrypt one or several sets of data related to some individuals, optionally perform some basic operations on them (such as union or intersection) and finally count the number of individuals in

the set. As we will see, we will investigate two different process within PAPAYA: using encrypted Bloom filters and using inner-product functional encryption.

In the sequel of this section, we first give some general definitions for Bloom filters and inner product, then precise the privacy challenges in which we will focus. After that, we are also planning to focus on the related work on the subject. We finally give the way we plan to use inner-product for privacy-preserving counting, and the way it is possible to perform set intersection, set union and counting on encrypted Bloom filters.

On this aspect, this deliverable only give a high-level overview of the way we plan to proceed. The complete specification and implementation will be detailed in D3.3.

5.1 Definition

Let $\mathcal{D} = \{d_1, \cdots, d_i, \cdots, d_\ell\}$ be a finite set of some entries denoted by d_i. As the size ℓ of the set can be very large, we assume that the set is represented by a counting table denoted $\mathcal{T}$. The first option that we will consider is to make use of Bloom filters. The second way to perform some simple statistics (such as a mean) is to make use of an inner product.

Bloom filters. More precisely, a Bloom filter is used to store a set of data of variable size into a bit-string of fixed size. It is then possible, with a simple test, to estimate the probability that an element is in the set of data (which depends on the size of the result bit-string and on the number of data). This probability is equal to 0 if the output of the test is "no". More precisely, the result is an m-bit string named T. We note $T = t_{m-1} \cdots t_1 t_0$ where each $t_i \in \{0, 1\}$. Initially, T is set to $0 \cdots 00$. We have then ℓ elements $d_1, d_2, \cdots d_\ell \in \mathcal{D}$ of various size. Moreover, let us define q hash functions $\mathcal{H}_1, \cdots, \mathcal{H}_q$ where each $\mathcal{H}_i : \{0,1\}^* \longrightarrow \{0,1\}^c$ with $m = 2^c$.

For $j \in [1, \ell]$, the Bloom filter creation consists of computing $\mathcal{H}_1(d_j), \cdots, \mathcal{H}_q(d_j)$. For every $v \in [1, q]$, we assign $t_i = 1$ where $i = \mathcal{H}_v(d_j)$. If one wants to know the element $\tilde{d}$ is in the set $\mathcal{D}$ or not by using the created Bloom filter, one has to compute $\tilde{i}_v = \mathcal{H}_v(\tilde{d})$ for every $v \in [1, q]$. If there is an element $v_0 \in [1, q]$ such that $t_{\tilde{i}_{v_0}} = 0$ then, $\tilde{d} \notin \mathcal{D}$, otherwise, $\tilde{d} \in \mathcal{D}$. Such method is probabilistic in the sense that if the above testing procedure outputs that $\tilde{d} \in \mathcal{D}$, with an error probability of about $\left(1 - e^{\frac{-\ell q}{m}}\right)^q$.

The idea of a privacy-preserving counting with Bloom filters is to perform some basic operations on one or several such filters that are given in an encrypted form. We consider that the filling of a Bloom filter is realized in clear text. Later, it is possible to use one or several Bloom filters for counting, as given below:

- for one Bloom filter T (related to the set $\mathcal{D}$), determine the size ℓ of $\mathcal{D}$;
- for two Bloom filters T_1 (related to the set $\mathcal{D}_1$) and T_2 (related to the set $\mathcal{D}_2$), determine the size of $\mathcal{D}_1 \cup \mathcal{D}_2$;
- for two Bloom filters T_1 (related to the set $\mathcal{D}_1$) and T_2 (related to the set $\mathcal{D}_2$), determine the size of $\mathcal{D}_1 \cap \mathcal{D}_2$.

Inner products. Let the set $\mathcal{D}$ be a vector $V = (v_1, \cdots, v_\ell)$ of ℓ values. Considering two vectors V and $W = (w_1, \cdots, w_\ell)$, the inner product of V and W, denoted $\langle V, W \rangle$ is computed

as follows

$$\langle V, W \rangle = \sum_{i=1}^{\ell} v_i w_i.$$

Such a mathematical tool can be used, in an easy way, to compute:

- a mean of a set represented as a vector V, by computing $\langle V, \mathbf{1} \rangle / \ell$ where $\mathbf{1} = (1, \cdots, 1)$;
- an weighted mean of a set represented as a vector V with weights represented as $W = (w_1, \cdots, w_\ell)$, by computing $\langle V, W \rangle$.

5.2 Privacy Challenges

The main idea of the PAPAYA project is to protect the individual's data that are used to perform some basic operations.

When considering counting, intersection and union operations using Bloom filters, such filter should be manipulated in an encrypted form. There are two steps in a Bloom filter process: to fill the filters and then to test whether an entry is in the filter or not. Within PAPAYA, our main focus will be on the second step, considering that the first one will be realized by using known plain data. This case is related to the privacy preserving mobility analytics use case (UC3) in the mobile and phone usage umbrella defined in D2.1 [29]. This one consists of filling a Bloom filter, using probe data, during some study period (defined by the third party requester). The input will be the hash of the MSISDN (unique identifer of a mobile network user). After such study period, the resulting Bloom filter is encrypted so that nobody can retrieve the unencrypted form of the Bloom filter. Then, a mobile network operator will be able to make some basic statistics on given encrypted Bloom filters such as counting the number of individuals or comparing two Bloom filters by using set union or set intersection in the encrypted domain.

For the computation of a mean by using an inner product, the idea is quite similar. We consider that the input vector is given in an encrypted form and we want to compute the inner product in the encrypted domain. This is related to the mobile usage analytics use case (UC4) in the mobile and phone usage umbrella defined in D2.1 [29] in which the inputs come from individuals, are aggregated by a designated entity (called the Aggregator) and then proceeded to obtain the requested mean. Depending on the type of mean we want to compute, either the input of a unique individual will be a vector, or a unique value. In the latter case, the Aggregator can create a vector using the inputs of several individuals. Our aim is then to find a way to compute an inner product in the encrypted domain, that is where one vector is encrypted, and the other not.

5.3 Related Work

Encrypted Bloom filters. Several papers have proposed to make a link between Bloom filters and encryption schemes. In [52, 13], Bloom filters are used for checking whether a document contains certain keywords without disclosing all of them. The proposed mechanism consists of computing the Bloom filter hash function as a cryptographic pseudo-random function. Similarly, an interactive protocol for securely checking set inclusion via Bloom filters is described in [83].

Their proposal permits to hide both the Bloom filter content and the element to be checked for inclusion. Finally, in [67], Kerschbaum proposed a way to both add a new element into the Bloom filter and test whether a given data is in the set represented by an encrypted Bloom filter or not.

To the best of our knowledge, there exists no study for the following operations:

1. fill in one or several Bloom filters with plain data;
2. encrypt the Bloom filter(s);
3. perform operations (counting, union, intersection) on the encrypted Bloom filter(s).

Inner-product functional encryption. There are several papers proposing a functional encryption scheme for the inner product [2, 5, 15, 3]. The idea is that the key generation gives one vector $\mathbf{y}$, the encryption gives a second vector $\mathbf{x}$ and the decryption permits to output the inner product of both: $\langle \mathbf{x}, \mathbf{y} \rangle$. In this section, we focus on the construction given by Agrawal, Libert and Stehlé [5], based on the Diffie-Hellman problem. This is mainly because of its good efficiency.

- $Setup(1^\lambda, 1^\ell)$: choose a cyclic group $\mathbb{G}$ of prime order $q > 2^\lambda$ with generators $g, h \in \mathbb{G}$. Then, for each $i \in \{1, \cdots, \ell\}$, sample $s_i, t_i \in \mathbb{Z}_q$ and compute $h_i = g^{s_i} \cdot h^{t_i}$. Define $\mathbf{s} = (s_1, \cdots, s_\ell)$, $\mathbf{t} = (t_1, \cdots, t_\ell)$ and $msk := (\mathbf{s}, \mathbf{t})$ and $mpk := \big(\mathbb{G}, g, h, \{h_i\}_{i=1}^\ell\big)$.
- $KeyGen(msk, x)$: to generate a key for the vector $\mathbf{x} = (x_1, \cdots, x_\ell) \in \mathbb{Z}_q^\ell$, compute $sk_x = (s_x, t_x) = (\sum_{i=1}^\ell s_i \cdot x_i, \sum_{i=1}^\ell t_i \cdot x_i) = (\langle \mathbf{s}, \mathbf{x} \rangle, \langle \mathbf{t}, \mathbf{x} \rangle)$.
- $Encrypt(mpk, y)$: To encrypt a vector $\mathbf{y} = (y_1, \cdots, y_\ell) \in \mathbb{Z}_q^\ell$, sample $r \in \mathbb{Z}_q$ and compute
$$C = g^r, D = h^r, \{E_i = g^{y_i} \cdot h_i^r\}_{i=1}^\ell.$$
Return $C_\mathbf{y} = (C, D, E_1, \cdots, E_\ell)$.
- $Decrypt(mpk, sk_x, C_y)$: Given $sk_x = (s_x, t_x)$, compute
$$E_x = \Big(\prod_{i=1}^\ell E_i^{x_i}\Big)/(C^{s_x} \cdot D^{t_x}).$$
Then, compute and output $\log_g(E_x)$.

We remark that such a scheme has the interesting property of being additively homomorphic. Given two ciphertexts $C_\mathbf{y} = (C, D, E_1, \cdots, E_\ell)$ (resp. $C_{\tilde{\mathbf{y}}} = (\tilde{C}, \tilde{D}, \tilde{E}_1, \cdots, \tilde{E}_\ell)$) on two messages y (resp. $\tilde{y}$), one can compute $C_{\bar{\mathbf{y}}} = (\bar{C}, \bar{D}, \bar{E}_1, \cdots, \bar{E}_\ell) = (C\tilde{C}, D\tilde{D}, E_1\tilde{E}_1, \cdots, E_\ell\tilde{E}_\ell)$ on $\bar{y} = y + \tilde{y}$.

5.4 Privacy preserving Counting based on FE

As shown in PAPAYA's cryptographic techniques, functional encryption is a way to perform some operations in the encrypted domain. In our case, we consider using such a cryptographic building block to compute (weighted) means using an inner-product. Our idea is to start from the practical construction given in the related work section, but as the use case necessitates more functional requirements, we need to transform it to suit our needs. We may then later consider another more suited concrete construction.

More precisely, the main idea is to proceed as follows.

- The Aggregator obtains an evaluation key dk_{IP} for the inner product related to the vector $\mathbf{1} = (1, \cdots, 1)$.
- The i-th individual encrypts its data d_i using the $Encrypt$ algorithm of the inner-product functional encryption scheme with the message $(0, \cdots, 0, d_i, 0, \cdots, 0)$ where d_i is put at the i-th position of the input vector. The resulting ciphertext is sent to the Aggregator.
- The Aggregator makes use of the homomorphic property of the above ciphertext to obtain an encryption of the vector $\mathbf{d} = (d_1, \cdots, d_\ell)$. It then uses dk_{IP} and the $Decrypt$ procedure to obtain $\langle \mathbf{d}, \mathbf{1} \rangle / \ell$, which exactly corresponds to the desired mean.

We note that a weighted mean can easily be obtained in the same way by replacing the vector $\mathbf{1}$ for the relevant one. Moreover, this "simple" scheme does not really suit our needs. In fact, the following requirements are necessary to really protect individual's privacy in the mobile usage analytics use case (UC4) in the mobile and phone usage umbrella defined in D2.1 [29]:

1. no possibility to decrypt individual's data;
2. no possibility to obtain the analytics on a single individual (analytics cannot be done before aggregation);
3. no possibility to perform any other analytics (analytics should be done according to a key, and no possibility to derive another key); and
4. no possibility to collect individual's data without his/her consent.

This is easy to see that the above sketch of scheme does not fulfill those requirements as the Aggregator can easily obtain the value of a single individual by applying the $Decrypt$. We have moreover not given any details on the way the Aggregator can obtain the evaluation key on the vector $\mathbf{1} = (1, \cdots, 1)$.

The privacy-preserving mobile usage analytics use case will however necessitate to modify such inner-product functional encryption to fit all the requirements. We need to perform some more work on that before being ready for a true specification. Indeed, there are several options in which we will focus in the next months. At first, the way the data are aggregated in the above scheme poses several problem and a better direction is to consider multi-client inner product construction, such as the one in [27]. Another issue is that if one individual does not send his own entry, a dishonest aggregator can decrypt partial ciphertexts and obtain information about individual's data. The idea is then to add a secret sharing layer, as proposed in [28],

so that until all individuals have participated, the decryption does not work. However, as all registered individuals may not really participate, we furthermore need to replace the secret sharing technique by a threshold-based one, using Lagrangian interpolation. All the details will be given in the next WP3 deliverable.

5.5 Privacy preserving Set Intersection & Set Union

Let us consider a Bloom filter T of length m, that has been filled in using q hash functions for the set $\mathcal{D}$. Let n be the number of 1 in T. In [95], Swamidass and Baldi give a formula permitting to approximate the number of items $\tilde{\ell}$ in the set $\mathcal{D}$. This is given by

$$\tilde{\ell} = -\frac{m}{q} \ln \left(1 - \frac{n}{m}\right)$$

Now considering that the Bloom filter is encrypted, it is sufficient to count the number of 1 in the encrypted filter to obtain such a value. In fact, it suffices to encrypt the Bloom filter bitwise with an additively homomorphic encryption scheme to count the number of items in the set $\mathcal{D}$. More precisely, we can proceed as follows for a given Bloom filter $T = t_{m-1} \cdots t_1 t_0$:

1. for all $i \in [0, m-1]$, encrypt t_i with some key k, as $c_i = Encrypt(t_i, k)$;
2. compute $C_i = Encrypt(\sum_{i=0}^{m-1} t_i, k)$ using the homomorphic property of the encryption scheme;
3. decrypt C_i to obtain $n = \sum_{i=0}^{m-1} t_i$ and finally compute $\tilde{\ell}$ using the above formula (in the plain domain).

Let us now consider set intersection and union. In fact, in [95], the size $\ell_\cup$ of the union of two sets $\mathcal{D}_0$ and $\mathcal{D}_1$, represented by two Bloom filters T_0 and T_1 (defined again with size m and q hash functions) can be estimated by the following formula:

$$\tilde{\ell_\cup} = -\frac{m}{k} \ln \left(1 - \frac{n_\cup}{m}\right)$$

where $n_\cup$ is the number of bits set to one in either of (or both) the two Bloom filters.

In the encrypted domain, the above strategy can easily be derived to the case of two Bloom filters. To compute $\tilde{\ell_\cup}$, it remains to compute $n_\cup$, which can be done using the appropriate homomorphic encryption which permits to compute $T_0 + T_1$ (OR operation). Considering that we have an encryption scheme with both additive (modulo 2, denoted $\oplus$) and multiplicative ($\times$) homomorphism, one can easily compute such a relation (considering that $T_0 + T_1 = T_0 \oplus T_1 \oplus (T_0 \times T_1)$), and then design a solution to our problem. Given the bitwise encryption E_0 (resp. E_1) of the Blum filter T_0 (resp. T_1), one can homomorphically compute the bitwise encryption E_+ of $T_0 + T_1$ (by using the above formula), and then proceed as for the counting (see above).

Finally, the intersection of two Bloom filters can be estimated as

$$\tilde{\ell_\cap} = n_0 + n_1 - n_\cup,$$

which can easily be solved using the above results.

The encryption scheme that we will use in single source architecture for mobility analytics use case in the mobile and phone usage umbrella defined in D2.1 [29] depend on the exact way such encryption scheme will be used. Several options are possible, and in particular homomorphic encryption, functional encryption and multi-party computation. This will mainly depend on the entity having the right to decrypt the value n.

6 Conclusions

In this deliverable, we have analyzed the PAPAYA use cases in [29] to address the particular Privacy Preserving (PP) data analytics operations, namely PP Neural Network (PP-NN) classification, PP collaborative training, PP clustering and PP counting models. The presented solutions can be categorized as follows:

- For PP-NN, first, we have presented the background and definitions for NN and then we have summarized the important applications of PP-NN. For PP-NN classification, we have introduced three solutions. One of these solutions employs PP-NN together with two-party computation (2PC) and provide a complete design in particular for UC1 in the healthcare umbrella defined in deliverable D2.1 [29]. In addition, we have also presented the initial design of the two different and generic PP-NN schemes that utilize partially homomorphic encryption and fully homomorphic encryption as cryptographic techniques and these solutions, however, are more generic and could be applied on other NN architectures as well. In addition, we have provided the preliminary design of PP collaborative training algorithm together with the robustness against existing attacks on collaborative training that is suitable to UC2.
- For PP clustering, we have investigated the problem that consists of regrouping data items in k clusters without disclosing the content of the data. First, we have presented an overview for the well-known clustering algorithms and PP version of these algorithms. Moreover, we have provided a preliminary design of a PP trajectory clustering solution based on the use of partially homomorphic encryption that can be considered as a potential solution for UC3.
- For PP counting, first, we have presented the background and definitions for cryptographic techniques. Furthermore, we have also introduced the preliminary design for basic PP statistics such as PP Counting based on functional encryption and PP set intersection and set union with encrypted Bloom filters. These solutions can be considered suitable to UC4.

The recently defined Use Case 5 (threat detection, UC5 [29]) will be studied and appropriate privacy preserving solutiosn wilThe complete design and specification of presented preliminary solutions will be provided in deliverable D3.3.

References

[1] AT&T Database of Faces.

[2] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751. Springer, 2015.

[3] Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In *Advances in Cryptology - EUROCRYPT 2017*, volume 10210 of *Lecture Notes in Computer Science*, pages 601–626. Springer, 2017.

[4] Shashank Agrawal and David J. Wu. Functional encryption: Deterministic to randomized functions from simple assumptions. In *Advances in Cryptology - EUROCRYPT 2017*, volume 10211 of *Lecture Notes in Computer Science*, pages 30–61. Springer, 2017.

[5] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Advances in Cryptology - CRYPTO 2016*, volume 9816 of *Lecture Notes in Computer Science*, pages 333–362. Springer, 2016.

[6] Nawal Almutairi, Frans Coenen, and Keith Dures. Secure third party data clustering using ϕ data: Multi-user order preserving encryption and super secure chain distance matrices. Artificial Intelligence XXXV, 2018. `https://www.springerprofessional.de/secure-third-party-data-clustering-using-data-multi-user-order-p/16295950`.

[7] Afonso Arriaga, Manuel Barbosa, and Pooya Farshim. Private functional encryption: Indistinguishability-based definitions and constructions from obfuscation. In *Progress in Cryptology - INDOCRYPT 2016 - 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings*, volume 10095 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2016.

[8] Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. *SIAM J. Comput.*, 45(6):2117–2176, 2016.

[9] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. How to backdoor federated learning. *CoRR*, abs/1807.00459, 2018.

[10] Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 67–98. Springer, 2017.

[11] Donald Beaver. Efficient multiparty protocols using circuit randomization. In *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, pages 420–432, n.a., 1991. Springer.

[12] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. Cryptology ePrint Archive, Report 2013/426, 2013. `https://eprint.iacr.org/2013/426`.

[13] Steven M. Bellovin and William R. Cheswick. Privacy-enhanced searches using encrypted bloom filters. Cryptology ePrint Archive, Report 2004/022, 2004. `https://eprint.iacr.org/2004/022`.

[14] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: A system for secure multi-party computation. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 257–266, New York, NY, USA, 2008. ACM.

[15] Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. Cca-secure inner-product functional encryption from projective hash functions. In *Public-Key Cryptography - PKC 2017*, volume 10175 of *Lecture Notes in Computer Science*, pages 36–66. Springer, 2017.

[16] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 325–341, 2005.

[17] P. S. Bradley and Usama M. Fayyad. Refining initial points for k-means clustering. pages 91–99. Morgan kaufmann, 1998.

[18] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.

[19] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. *J. Cryptology*, 31(1):202–225, 2018.

[20] Paul Bunn and Rafail Ostrovsky. Secure two-party k-means clustering. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pages 486–497, New York, NY, USA, 2007. ACM.

[21] Francisco Castells, Pablo Laguna, Leif Sörnmo, Andreas Bollmann, and José Millet Roig. Principal component analysis in ecg signal processing. *EURASIP Journal on Advances in Signal Processing*, 2007(1):074580, Dec 2007.

[22] Hervé Chabanne, Amaury de Wargny, Jonathan Milgram, Constance Morel, and Emmanuel Prouff. Privacy-preserving classification on deep neural network. Cryptology ePrint Archive, Report 2017/035, 2017. `https://eprint.iacr.org/2017/035`.

[23] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, pages 409–437, 2017.

[24] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachéne. Tfhe: Fast fully homomorphic encryption over the torus. Cryptology ePrint Archive, Report 2018/421, 2018. `https://eprint.iacr.org/2018/421`.

[25] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. Cryptology ePrint Archive, Report 2016/870, 2016. `https://eprint.iacr.org/2016/870`.

[26] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Improving tfhe: faster packed homomorphic operations and efficient circuit bootstrapping. Cryptology ePrint Archive, Report 2017/430, 2017. `https://eprint.iacr.org/2017/430`.

[27] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, volume 11273 of *Lecture Notes in Computer Science*, pages 703–732. Springer, 2018.

[28] Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with repetition for inner product. *IACR Cryptology ePrint Archive*, 2018:1021, 2018.

[29] Eleonora Ciceri, Stefano Galliani, Marco Mosconi, Monir Azraoui, and Sébastien Canard. D2.1: Use Cases and Requirements. PAPAYA Deliverable D2.1, 2019.

[30] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explor. Newsl.*, 4(2):28–34, December 2002.

[31] The European Commission. Regulation of the european parliament and of the council concerning the respect for private life and the protection of personal data in electronic communications and repealing directive 2002/58/ec (regulation on privacy and electronic communications), 2016.

[32] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. *European Transactions on Telecommunications*, 8(5):481–490, 1997.

[33] Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *Public Key Cryptography, 4th International*

Workshop on Practice and Theory in Public Key Cryptography, PKC 2001, Cheju Island, Korea, February 13-15, 2001, Proceedings, pages 119–136, n.a., 2001. Springer.

[34] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, pages 1–15, n.a., 2015. The Internet Society.

[35] 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*, PAIS '08, pages 3–11, New York, NY, USA, 2008. ACM.

[36] Nathan Dowlin, Ran Gilad-Bachrach, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 201–210. JMLR.org, 2016.

[37] W. Du and M. Atallah. Privacy-preserving cooperative statistical analysis. In *Proceedings of the 17th Annual Computer Security Applications Conference*, ACSAC '01, pages 102–, Washington, DC, USA, 2001. IEEE Computer Society.

[38] Edouard Dufour Sans, Romain Gay, and David Pointcheval. Reading in the dark: Classifying encrypted digits with functional encryption. *IACR Cryptology ePrint Archive*, 2018:206, 2018.

[39] Pierre-Alain Dupont and David Pointcheval. Functional encryption with oblivious helper. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2017, Abu Dhabi, United Arab Emirates, April 2-6, 2017*, pages 205–214. ACM, 2017.

[40] Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag.

[41] Yael Ejgenberg, Moriya Farbstein, Meital Levy, and Yehuda Lindell. Scapi: The secure computation application programming interface. Cryptology ePrint Archive, Report 2012/629, 2012. `https://eprint.iacr.org/2012/629`.

[42] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[43] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.

[44] Ben Fisch, Dhinakaran Vinayagamurthy, Dan Boneh, and Sergey Gorbunov. IRON: functional encryption using intel SGX. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 765–782. ACM, 2017.

[45] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

[46] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016.

[47] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `crypto.stanford.edu/craig`.

[48] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 169–178, New York, NY, USA, 2009. ACM.

[49] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. In *Proceedings of the 32Nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, pages 850–867, Berlin, Heidelberg, 2012. Springer-Verlag.

[50] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75–92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[51] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *Proceedings of the 7th International Conference on Information Security and Cryptology*, ICISC'04, pages 104–120, Berlin, Heidelberg, 2005. Springer-Verlag.

[52] Eu-Jin Goh. Secure indexes. *IACR Cryptology ePrint Archive*, 2003:216, 2003.

[53] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229, New York, NY, USA, 1987. ACM.

[54] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[55] Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in helib. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 93–120, 2018.

[56] Shai Halevi and Victor Shoup. Helib: An implementation of homomorphic encryption. https://github.com/shaih/HElib, 2019.

[57] Carmit Hazay and Yehuda Lindell. A note on the relation between the definitions of security for semi-honest and malicious adversaries. *IACR Cryptology ePrint Archive*, 2010:551, 10 2010.

[58] Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. TASTY: tool for automating secure two-party computations. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 451–462, New York, NY, USA, 2010. ACM.

[59] Briland Hitaj, Giuseppe Ateniese, and Fernando Pérez-Cruz. Deep models under the GAN: information leakage from collaborative deep learning. *CoRR*, abs/1702.07464, 2017.

[60] http://deeplearning.net. Convolutional neural networks (lenet). `http://deeplearning.net/tutorial/lenet.html`. [Online; accessed 14-April-2019].

[61] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[62] Geetha Jagannathan and Rebecca N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 593–599, New York, NY, USA, 2005. ACM.

[63] Somesh Jha, Luis Kruger, and Patrick McDaniel. Privacy preserving clustering. In Sabrina de Capitani di Vimercati, Paul Syverson, and Dieter Gollmann, editors, *Computer Security – ESORICS 2005*, pages 397–417, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[64] I.T. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[65] Angela Jäschke and Frederik Armknecht. Unsupervised machine learning on encrypted data. Cryptology ePrint Archive, Report 2018/411, 2018. `https://eprint.iacr.org/2018/411`.

[66] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A low latency framework for secure neural network inference. In *USENIX Security Symposium*, pages 1651–1669. USENIX Association, 2018.

[67] Florian Kerschbaum. Public-key encrypted bloom filters with applications to supply chain integrity. In *Data and Applications Security and Privacy XXV - 25th Annual IFIP WG 11.3 Conference, DBSec 2011, Richmond, VA, USA, July 11-13, 2011. Proceedings*, volume 6818 of *Lecture Notes in Computer Science*, pages 60–75. Springer, 2011.

[68] Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, volume 11035 of *Lecture Notes in Computer Science*, pages 544–562. Springer, 2018.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, n.a., 2012. n.a.

[70] K. Anil Kumar and C. Pandu Rangan. Privacy preserving dbscan algorithm for clustering. In Reda Alhajj, Hong Gao, Jianzhong Li, Xue Li, and Osmar R. Zaïane, editors, *Advanced Data Mining and Applications*, pages 57–68, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[71] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[72] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 593–604, New York, NY, USA, 2007. ACM.

[73] Dongxi Liu. Homomorphic encryption for database querying. Australian Provisional Patent, 2012. `https://patents.google.com/patent/EP2865127A4/en#patentCitations`.

[74] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious neural network predictions via minionn transformations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 619–631, New York, NY, USA, 2017. ACM.

[75] Jinfei Liu, Joshua Zhexue Huang, Jun Luo, and Li Xiong. Privacy preserving distributed dbscan clustering. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT '12, pages 177–185, New York, NY, USA, 2012. ACM.

[76] Adriana Lopez-Alt, Eran Tromer, and Vinod Vaikuntanathan. Cloud-assisted multiparty computation from fully homomorphic encryption. Cryptology ePrint Archive, Report 2011/663, 2011. `https://eprint.iacr.org/2011/663`.

[77] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the*

49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[78] J. Macqueen. Some methods for classification and analysis of multivariate observations. In *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[79] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay—a secure two-party computation system. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 20–20, Berkeley, CA, USA, 2004. USENIX Association.

[80] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. Inference attacks against collaborative learning. *CoRR*, abs/1805.04049, 2018.

[81] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *IEEE Symposium on Security and Privacy*, pages 19–38. IEEE Computer Society, 2017.

[82] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 245–254, New York, NY, USA, 1999. ACM.

[83] Ryo Nojima and Youki Kadobayashi. Cryptographically secure bloom-filters. *Trans. Data Privacy*, 2(2):131–139, 2009.

[84] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 223–238, n.a., 1999. Springer.

[85] The European Parliment and the Council of the European Union. European general data protection regulation. https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679&from=EN.

[86] Michael O. Rabin. How to exchange secrets with oblivious transfer. Cryptology ePrint Archive, Report 2005/187, 2005. `https://eprint.iacr.org/2005/187`.

[87] M. S. Rahman, A. Basu, and S. Kiyomoto. Towards outsourced privacy-preserving multiparty dbscan. In *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 225–226, Jan 2017.

[88] F. Rao, B. K. Samanthula, E. Bertino, X. Yi, and D. Liu. Privacy-preserving and outsourced multi-user k-means clustering. In *2015 IEEE Conference on Collaboration and Internet Computing (CIC)*, pages 80–89, Oct 2015.

[89] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[90] Kazue Sako, editor. *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, volume 9610 of *Lecture Notes in Computer Science*. Springer, 2016.

[91] Saeed Samet and Ali Miri. Privacy preserving id3 using gini index over horizontally partitioned data. In *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*, AICCSA '08, pages 645–651, Washington, DC, USA, 2008. IEEE Computer Society.

[92] Saeed Samet, Ali Miri, and Luis Orozco-Barbosa. Privacy preserving k-means clustering in multi-party environment. In Javier Hernando, Eduardo Fernández-Medina, and Manu Malek, editors, *SECRYPT*, pages 381–385. INSTICC Press, 2007.

[93] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1310–1321, New York, NY, USA, 2015. ACM.

[94] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Des. Codes Cryptography*, 71(1):57–81, 2014.

[95] S. Joshua Swamidass and Pierre Baldi. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *J. Chem. Inf. Model*, 47(3):952–964, 2007.

[96] Martin Thoma. The reuters dataset, July 2017.

[97] Maneesh Upmanyu, Anoop M. Namboodiri, Kannan Srinathan, and C. V. Jawahar. Efficient privacy preserving k-means clustering. In *Proceedings of the 2010 Pacific Asia Conference on Intelligence and Security Informatics*, PAISI'10, pages 154–166, Berlin, Heidelberg, 2010. Springer-Verlag.

[98] Jaideep Vaidya and Chris Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 206–215, New York, NY, USA, 2003. ACM.

[99] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. Faster secure two-party computation in the single-execution setting. Cryptology ePrint Archive, Report 2016/762, 2016. `https://eprint.iacr.org/2016/762`.

[100] XU Wei-jiang, HUANG Liu-sheng, LUO Yong-long, YAO Yi-fei, and JING Wei-wei. Privacy-preserving DBSCAN clustering Over vertically partitioned data. In *International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 850–856. IEEE, 2007.

[101] Wikipedia. Activation function — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Activation%20function&oldid=890334491`, 2019. [Online; accessed 01-April-2019].

[102] Wikipedia. Artificial neural network — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Artificial%20neural%20network&oldid=891279554, 2019. [Online; accessed 14-April-2019].

[103] Wikipedia. Backpropagation — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Backpropagation&oldid=891771701, 2019. [Online; accessed 14-April-2019].

[104] Wikipedia. Convolutional neural network — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Convolutional%20neural%20network&oldid=891636189, 2019. [Online; accessed 14-April-2019].

[105] Ming-Jun Xiao, Liu-Sheng Huang, Yong-Long Luo, and Hong Shen. Privacy preserving id3 algorithm over horizontally partitioned data. pages 239– 243, 01 2006.

[106] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164, n.a., 1982. IEEE Computer Society.

[107] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA, 1986. IEEE Computer Society.