Research track

Case study

Alexandre Bois, Chao Wang, Marion Sauvage, Yassine Abbahaddou, Thibaut Hurson

Confidential AI

Supervisors :
Fabien Imbault
Bich-Liên Doan

27 novembre 2020

Report : State of the Art

CentraleSupélec

Table of contents

1 Introduction

1.1 Context

Artificial intelligence is a promising and already acclaimed field to solve with new techniques very challenging issues in various domains such as Health, Defense or Finance. The opportunities presented for face recognition applications are immense. Nevertheless it implies a lot of technical challenges in terms of computational efficiency or data collection and storage.

Indeed to process data for smartphones applications purposes or security procedures, secure bio-metric identification, safe storage, third-party servers and algorithms are required. The challenge is huge since it requires to secured at every step of a process. Technologies developed to ensure this privacy need to be not too computationally expensive for most application that require rapidity, and should also not deteriorate the data or lead to wrong results. In this paper, we will be making a state of the art of the available privacy techniques, focusing on among other things k-anonymization, differential privacy, and homomorphic encryption.

1.2 Legal and Ethical Aspect

At a time when 3.8 billion people are the owner of a smartphone and share their personal data, location or preferences with many application, the question of confidentiality is at the center of attention for individuals and institutions. Indeed the main actors of information technology industry in the US, that is to say Google, Apple, Facebook, Amazon and Microsoft, or the Big Six in Asia (Baidu, Huawei, Tencent, Xiaomi, Alibaba and ByteDance) have access to a huge amount of individuals, firms, national institutions and have a financial power that exceed many countries GDP.

The power given by the data those firms own and by the influence they can have on users with their products, many governments are taking measures to protect individuals and data in general. Privacy standards are rising accross the globe. One of the most ambitious protection measure is the General Data Protection Regulation (GDPR), established by the European Union in 2018. It addresses many privacy issues, namely the transfer of personal data outside the EU ans EEA areas, simplification of the regulatory environment for firms, and unification of privacy standards and data protection in the EU.

2 Traditional Methods

2.1 Satistical disclosure limitation (SDL)

Satistical disclosure limitation (SDL) techniques had been widely adopted to analyze and share privacy-sensitive data. This category of techniques includes basic methods such as the limitation of detail, the suppression of sensitive information, rounding, or the addition of noise. More advanced methods encompass sampling, matrix masking, data swapping or aggregation among others. [MH$^+$11]

Yet, the assumption that the problem of anonymization was solved was revoked in the 1990s when researchers succeeded in de-anonymizing anonymized datasets. One of the main re-identification technique at that time was to link a de-identified record with identified records available in the outside world. [Ohm09] Latanya Sweeney used this technique to re-identify de-identified hospital records. Dr Sweeney discovered that most people in the United State where uniquely identified by three attributes : their date of birth, sex, and ZIP code. By linking the hospital records with public voting records containing the name, address, date of birth, sex, and ZIP code of voters, Sweeney managed to re-identify the hospital record. The date of birth, sex, and ZIP code are attributes whose values when taken together can identify an individual and are called quasi-identifiers.

2.2 k-anonymity, l-diversity and t-closeness

The tremendous increase in the demand for person-specific data have made it impossible for data holders to use traditional statistical techniques to share sensitive information. Sweeney introduced the k-anonymity model to cope for this lack of privacy protection. In this model, a data set respect k-anonymity protection if the information of each individual is indistinguishable with at least k-1 other individuals. [Swe02] Although the k-anonymity model succeed in protecting against identity disclosure, Machanavajjhala et al. [MKGV07] pointed out that it does not protect against sensitive attributes disclosure. Secondly, they proved that k-anonymity does not protect again attackers that are using background knowledge. Machanavajjhala et al. introduced a new model called l-diversity in which each sensitive attributes should have at least l well-represented values.

Ninghui et al. proved that l-diversity cannot prevent attribute disclosure if the overall distribution is skewed. They also showed that semantic closeness of attribute values in equivalent class can lead to the disclosure of sensitive information. To twart those limitations, Ninghui et al. introduced a more advanced model called t-closeness. [LLV07] This model uses the notion of equivalence class which can be defined as a set of records that have the same values for the quasi-identifiers. In t-closeness, the distribution of a sensitive attribute in an equivalence class is close to the distribution of the attribute in the overall table. In other word, the distance between the distribution of the attribute in an equivalence class and the distribution in the overall table should be no more than the threshold t.

2.3 Federated Learning

The concept of federated learning was first proposed by Google[KMRR16] in order to build machine-learning models based on datasets that are distributed across multiple devices while preventing data leakage. Instead of putting all data owners' perspective data together to train a model, federated learning's data owners collaboratively train a model without expose each one's own data to others, while preserving the model's accuracy performance.

Federated Learning could be divided into two different categories[YLCT19]. We denote the data held by each data owner i as matrix $\mathcal{D}_i$, then the feature space as $\mathcal{X}$, the label space as $\mathcal{Y}$, and the sample ID space as $\mathcal{I}$, so they constitute the complete training dataset $(\mathcal{I}, \mathcal{X}, \mathcal{Y})$. Based on how data is distributed, we classify federated learning into horizontal and vertical federated learning. The horizontal one is introduced in the scenarios in which datasets share the same feature space but different space in samples $(X_i = \mathcal{X}_j, \quad \mathcal{Y}_i = \mathcal{Y}_j, \quad I_i \neq I_j, \quad \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j)$, for example two regional banks' business is very similar, but the interaction set of their users is very small . The vertical one is applicable to cases in which two datasets share the same sample ID space but differ in feature space $(\mathcal{X}_i \neq \mathcal{X}_j, \quad \mathcal{Y}_i \neq \mathcal{Y}_j, \quad I_i = I_j, \quad \forall \mathcal{D}_i, \mathcal{D}_j, i \neq j)$, for example a bank and a e-commerce company in the same city are likely to have similar user sets, while one records the user's revenue and expenditure behavior while the other retains the user's browsing and purchasing history.

Having known the two kinds of federated learning, now we illustrate their architecture separately. For horizontal federated learning, we suppose all participants are honest whereas the server is honest but curious, so no leakage of information from participant to server is allowed. with this principle, the training process contains four steps :

• Step 1 : Participants locally compute training gradients ; mask a selection of gradients with encryption/differential privacy/secret sharing techniques ; and send masked results to the server.

• Step 2 : The server performs secure aggregation without learning information about any participant.

• Step 3 : The server sends back the aggregated results to participants.

• Step 4 : Participants update their respective model with the decrypted gradients.

For vertical federated learning, we assume participants A and B are honest but curious to each other (suppose B also has label data that the model needs to predict), so a third-party collaborator C who is honest and does not collude with A or B is involved. Before training the model, the federated learning system first uses the encryption-based user ID alignment techniques to determine the common users of both A and B, without exposing their respective data. Then the training process can also be divided into four steps :

•Step 1 : Collaborator C creates encryption pairs and sends a public key to A and B.

•Step 2 : A and B encrypt and exchange the intermediate results for gradient and losscalculations.

•Step 3 : A and B compute encrypted gradients and add anadditional mask, respectively. Balso computes encrypted loss. A and B send encrypted values to C.

•Step 4 : C decrypts and send the decrypted gradients and loss back to A and B. A and Bunmask the gradients and update the model parameters accordingly.

3 Current Methods

3.1 Diffrential Privacy

Differential Privacy (**DP**) is a mathematical approach to quantify the privacy level of a randomized algorithm (M). It allows protecting privacy while preserving statistical usefulness : The DP algorithms are mainly used for publishing datasets without sharing identifiable information.

One of the primary goals of DP is to balance between minimizing privacy loss and maximizing accuracy. The parameter ϵ is used to control the privacy level of (M) : The higher the value of ϵ, the higher is the privacy loss.

In general terms, the requirement for differential privacy can be described as follows : The algorithm (M) should produce similar outputs on neighboring databases $D \sim D'$, which differ by a single entry. This condition is expressed by the probability inequality :

$$\mathbb{P}(M(D) \in S) \leqslant e^{\epsilon} \cdot \mathbb{P}(M(D') \in S)$$

where S is a subset of the output space.

A commonly used method to achieve ϵ-DP is the **Laplace mechanism** as shown below : Consider a function f, the sensitivity Δf is the maximum influence that a single individual can have on the result of a numeric query :

$$\Delta f = max_{D,D'} ||f(D) - f(D')||$$

When a user submits a query for data,the Laplace mechanism adds noise drawn from a Laplace distribution to the requested data :

$$M_{Lap}(D) = M(D) + Lap\left(\frac{\Delta M}{\epsilon}\right)$$

where $Lap(\frac{\Delta M}{\epsilon})$ is a probability distribution of density

$$x \mapsto \frac{\epsilon}{2\Delta M}\text{exp}\left(\frac{-|x|\epsilon}{\Delta M}\right)$$

The ϵ-DP inequality for Laplace method has been demonstrated in [DR+14]. This technique make our data private while preserving statistical usfulness The noisy dataset can then be used in predictive algorithms. A graphical illustration of incorporating differential privacy into machine learning for privacy preserving is shown in Figure 1

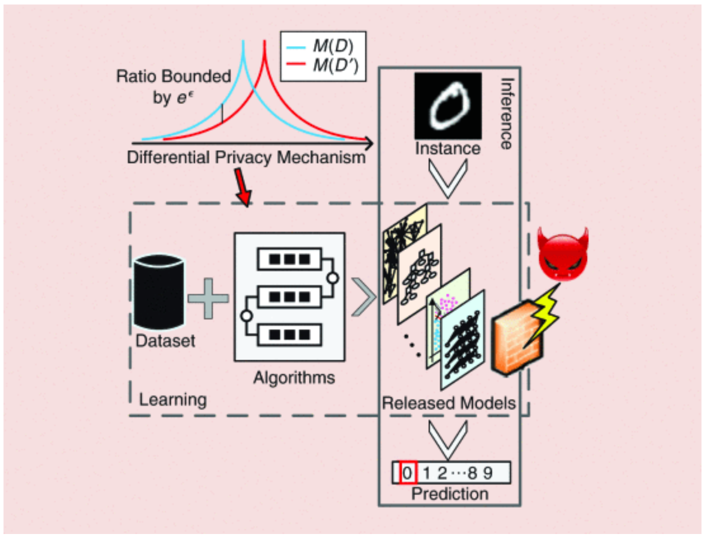

FIGURE 1 – Laplace mechanism combined to statistical algorithms

3.2 Homomorphic Encryption

Homomorphic encryption (HE) is a cryptographic primitive that makes it possible to compute additions and/or multiplications on encrypted data without decrypting them : the result of the computation over ciphertexts is an encryption of the result of the computation over plaintexts. We first give a formal definition of additively homomorphic encryption schemes (the definition can easily be adapted to multiplicatively homomorphic schemes), that allow additions (resp. multiplications) of ciphertexts. In what follows, λ denotes the security parameter : the other parameters must be chosen so that each desired property of the scheme is true with probability $1 - \mathsf{negl}(\lambda)$, where negl denotes any function that is negligible compared to the inverse of any polynomial (e.g. $\mathsf{negl}(\lambda) = 2^{-\lambda}$).

$$\begin{tikzcd}
\mathrm{enc}(m) \arrow[r, "\mathrm{eval}"] & \mathrm{enc}(f(m)) \\
m \arrow[u, "\mathrm{enc}"] \arrow[r, "f"] & f(m) \arrow[u, "\mathrm{enc}"']
\end{tikzcd}$$

Figure 2 – Homomorphic property

Definition 1 (Homomorphic Encryption) *A (public-key) additively homomorphic encryption scheme* HE *is a tuple of algorithms* $(\mathsf{HE.ParamGen}, \mathsf{HE.KeyGen}, \mathsf{HE.Enc}, \mathsf{HE.Dec}, \mathsf{HE.Add})$ *defined as follows :*

$\mathsf{HE.ParamGen}(\lambda)$: *generates the public parameters of the scheme, such as descriptions of plaintext space M, ciphertext space, keyspace, randomness distributions, etc. The output of* $\mathsf{ParamGen}$ *is assumed to be input to any subsequent algorithm.*

$\mathsf{HE.KeyGen}(\lambda) \to (pk, evk, dk)$: *outputs a public encryption key pk, a public evaluation key evk, and a secret decryption key dk.*

$\mathsf{HE.Enc}_{pk}(m) \to c$: *encrypts a message $m \in M$ under public key pk, and outputs ciphertext c.*

$\mathsf{HE.Dec}_{dk}(c) \to m^*$: *decrypts a ciphertext c using dk, and outputs a plaintext $m^* \in M$.*

$\mathsf{HE.Add}_{evk}(c_1, c_2) \to c^*$: *Given the evaluation key evk and two ciphertexts c_1, c_2, it computes and outputs a ciphertext c^*.*

A HE scheme should have the three following properties :

Correctness for all $m_1, m_2 \in M$, we have :

$$\Pr\left[\begin{array}{r|l} \mathsf{HE.Dec}_{dk}(\mathsf{HE.Add}_{evk}(c_1,c_2)) & (pk,evk,dk) \xleftarrow{\$} \mathsf{HE.KeyGen}(\lambda) \\ = m_1 + m_2 & c_i \leftarrow \mathsf{HE.Enc}_{pk}(m_i)\ (i \in \{1,2\}) \end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

Semantic security for any PPT adversary A and $m_0 \neq m_1 \in M$, it holds that :

$$\mathsf{Pr}\left[\begin{array}{c|c} & (pk,evk,dk) \xleftarrow{\$} \mathsf{HE.KeyGen}(\lambda) \\ A(pk,evk,c_b) = b & b \xleftarrow{\$} \{0,1\} \\ & c_b \leftarrow \mathsf{HE.Enc}_{pk}(m_b) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Compactness the ciphertext size is bounded by some fixed polynomial in the security parameter, and is independent of the size of the evaluated circuit or the number of inputs it takes.

The correctness property states that the scheme should work as expected regarding encryption, decryption and homomorphic operations. The semantic security property prevents an adversary to get any information on a message from a ciphertext and the public keys. Note that this definition implies that the encryption algorithm should be randomized, or else an adversary could simply compute the unique encryption of m_0 and m_1, compare it to c_b, and distinguish any pair of messages. The compactness property makes the scheme usable in practice, to compute many operations on many ciphertexts.

The idea of homomorphic encryption was formalized a long time ago ([RAD+78]) and there exists various homomorphic encryption schemes such as Paillier [Pai99] (additively) or RSA [RSA78] (multiplicatively). The problem of *fully homomorphic encryption* (FHE), that consists in finding out whether it is possible to create a scheme both additively and multiplicatively homomorphic, was only solved in 2009 by Gentry [Gen09] : such schemes exist, and it is thus possible to compute any arithmetic circuit over encrypted data. FHE can be defined as above, with two homomorphic operation algorithms (Add and Mult) instead of one, but the usual definition uses an algorithm called Eval to evaluate functions on ciphertexts, as follows :

Definition 2 (Fully Homomorphic Encryption) *A (public-key) additively homomorphic encryption scheme* HE *is a tuple of algorithms (*$\mathsf{HE.ParamGen}$, $\mathsf{HE.KeyGen}$, $\mathsf{HE.Enc}$, $\mathsf{HE.Dec}$, $\mathsf{HE.Eval}$*) defined as follows :*

$\mathsf{HE.ParamGen}(\lambda)$: *generates the public parameters of the scheme, such as descriptions of plaintext space M, ciphertext space, keyspace, randomness distributions, etc. The output of* $\mathsf{ParamGen}$ *is assumed to be input to any subsequent algorithm.*

$\mathsf{HE.KeyGen}(\lambda) \to (pk, evk, dk):$ *outputs a public encryption key pk, a public evaluation key evk, and a secret decryption key dk.*

$\mathsf{HE.Enc}_{pk}(m) \to c:$ *encrypts a message $m \in M$ under public key pk, and outputs ciphertext c.*

$\mathsf{HE.Dec}_{dk}(c) \to m^*:$ *decrypts a ciphertext c using dk, and outputs a plaintext $m^* \in M$.*

$\mathsf{HE.Eval}_{evk}(g, c_1, ..., c_n) \to c^*:$ *Given the evaluation key evk, an arithmetic circuit $g : M \to M^n$ and n ciphertexts $c_1, ..., c_n$, it computes and outputs a ciphertext c^*.*

The correctness property becomes : $\forall m_1, .., m_n \in M, \forall g : M^n \to M$,

$$\Pr\left[\begin{array}{r|c} \mathsf{FHE.Dec}_{dk}(\mathsf{FHE.Eval}_{evk}(g, c_1, .., c_n)) & (pk, evk, dk) \xleftarrow{\$} \mathsf{HE.KeyGen}(\lambda) \\ = g(m_1, .., m_n) & c_i \leftarrow \mathsf{HE.Enc}_{pk}(m_i)\ (i \in \{1, .., n\}) \end{array}\right] = 1 - \mathsf{negl}(\lambda)$$

In practice, true FHE is not easy to get and require inefficient operations because the known schemes rely on (generalizations of) the learning with errors problem ([LPR10, BV11]), and involve a noise that grows with each homomorphic operation, and its growth for multiplications quickly gets to high. Gentry scheme relies on a technique called *bootstrapping* [Gen09], that transforms a *leveled FHE scheme* (a FHE scheme that is only correct for circuits whose multiplicative depth does not exceed a certain bound) into a FHE scheme. It roughly works as follows : if the leveled scheme can homomorphically evaluate its Dec algorithm, applying it to a new encryption of a ciphertext would return the new encryption of the messsage, with a "reset" noise. This technique is very inefficient, so in practice only leveled schemes are used, and some have techniques to reduce the noise growth. Other recent schemes focused on fast bootstrapping to use schemes that bootstrap frequently but efficiently. Some of the most efficient schemes are described in [BV11, FV12, GSW13, BGV14, CGGI16].

Homomorphic encryption is an important research subject in cryptography, as it makes it possible to compute any function while protecting the users' data. This would allow anyone to delegate computations to powerful servers without disclosing private information. For example, a weak device such as a phone could delegate work to remote servers, or a researcher could use supercalculators on encrypted data. FHE could be used on sectors where confidentiality is important (security, banking, medical) to allow external parties to analyse the data.

3.3 Secure (multi-party) Computation

Secure multi-party computation (SMPC) is a subfield of cryptography which allows to perform analyses on encrypted data by splitting the latter among multiple parties. In this way, the different parties are applying a publicly known function on their share of the data whithout being at risk of revealing the entire data. Revealing data would require both parties to grant permission. SMPC allows secret sharing in semi-trusted and low-trusted environments as described in [GMDR20]. In short, it should preserve the correctness of the output, while guarantying privacy of the multiple inputs.

SMPC can also hide the function that has to be applied to data by using universal circuits. The function owner shares its function via the Cloud. Thus, if the service provider program universal circuits to emulate the so-called function, the latter is kept private. However universal circuits are limited to small functions and cannot be used for machine learning.

Another limitation of this method is that having multiple independent inputs requires big infrastructures. To put together each individual input, it necessitates this content to be available online. Consequently, it raises scalability issues when it comes to huge amount of data.

An interesting application of Secure Multi Party Computation is in the medical field where patients data should remain private but needs to be access and analyzed by doctors to get a global pictures of patients suffering from a given condition, as mentioned in[AD18]. In the study, a platform was created where results and insights about groups of patients where provided to doctors freely. They could make query on platform and obtain data about patients. The latter would not have their identity compromised thanks to Secure Multi Party Computation cluster which handles cryptographic shares of the original data. It is coordinated with data providers (hospitals) who transfer the data to computing nodes of the Secure Multi Party Computation cluster. A coordinator handles all the computation and storing requests, as described on the figure below.

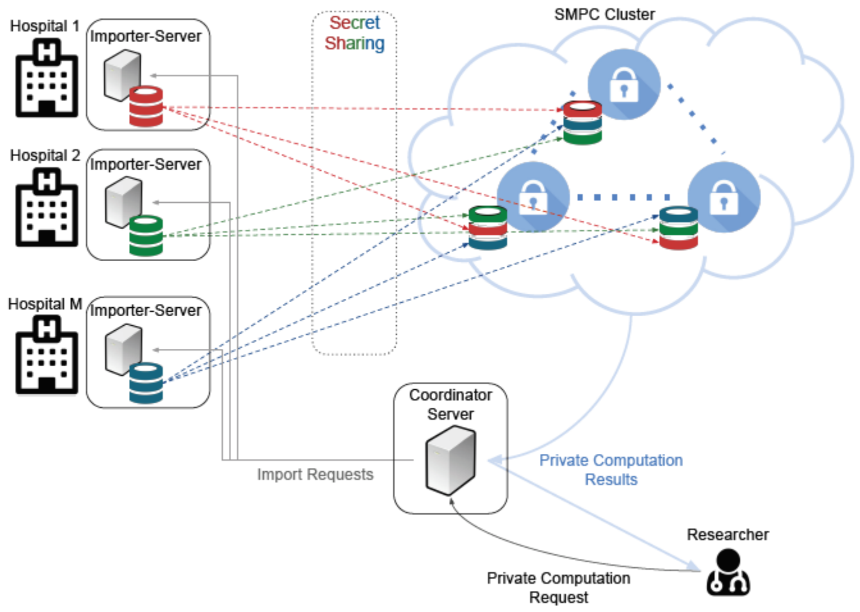

Figure 3 – Architecture of the medical SMPC system

3.4 Hardware Security Implementation

Although the cryptographic algorithms are mathmatically safe, their physical implementations on hardware can leak side-channel information (ex. the secret key) that can be attacked by third parties. So the main objective of cryptohardware is the design of secure cryptographic devices onto electronic platforms to assure data and algorithm privacy, resisting any kind of malicious attack.

One of the simplest to use is a crypto device with secure key storage. These devices, such as the Microchip ATECC508A and family, store crypto keys for a host processor as well as providing authentication services and key creation during public key exchanges, without exposing their private keys. Many also offer tamper detection, erasing stored keys when subjected to intrusive physical attacks.

Apple also proposed a patent called the secure enclave processor (SEP) and used it on its iPhone. The purpose of the Secure Enclave is to handle keys and other info such as biometrics that is sensitive enough to not be handled by the AP. It is isolated with a hardware filter so the AP cannot access it. It shares RAM with the AP, but its portion of the RAM (known as TZ0) is encrypted. The secure enclave itself is a flashable 4MB AKF processor core called the secure enclave processor.

3.5 Hybrid and Frameworks

Because of the diversity of the methods being developed and the diversity of the different types of application, researchers have explored how to make the best of the different privacy-preservation algorithms.

PC Mahawaga Arachchige et al. [CMABK^{+}20] introduced a framework named PPaaS (Privacy Preservation as a Service) which aims to simplify the utilization of privacy preserving methods. This framework includes a pool of privacy preserving techniques and select for each application the most suitable one. PPaas has been built to understand the level of privacy required by data owners while taking into account the utility of data requesters.

Hence, using such a framework could enable companies to dispose of the tedious work of selecting the best privacy-preservation algorithm while preserving utility.

4 Face Recognition

4.1 Context/ Challenges

Face recognition is one the major applications of artificial intelligence techniques today. Face recognition can be used to identify a person for example in a group of people, and to authenticate a person, that is to say check somebody identity. It is applied to many fields and issues such as city surveillance, phone unlocking, defense, or various security purposes as explained in [CNI19].It raises many ethical and personal privacy questions and it requires to have a secured process.Apart from the ethical and safety issues, problems which arises with that kind of usage are the one presented earlier with a major constraints due to image size. It implies storage and computational issues, which often involve using additional servers and cloud storage. Images must be sent to those instances in format that will not allow security breaches or any vulnerability.

4.2 Methods comparisons

Differential Privacy (**DP**) is one of the most efficient privacy-preserving techniques used for open face image dataset. There are two ways a differentially private system can work : *Global differential privacy* **GDP** and *Local differential privacy* **LDP**.

In the GDP setting, each user sends their data to this aggregator without noise. The aggregator takes this data, and transforms it with a differentially private mechanism. The aggregator can then publish the result or share it with third parties at the end of the process. The big gain using GDP is the accuracy . With the LDP, noise is directly added to each individual data point. Thus, users are more protected as they do not have to trust a third party.

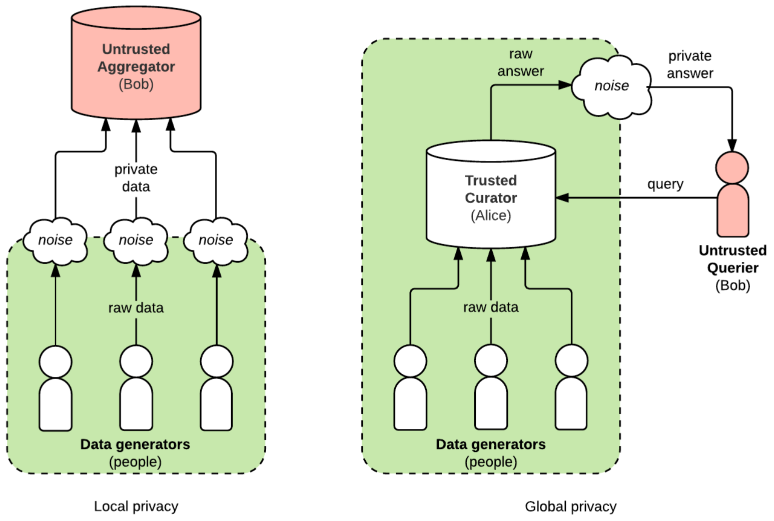

FIGURE 4 – Global vs Local DP

So, the main idea behind differential privacy is that the answer to a query sent to a noisy database would be the same than if no noise had been added. Of course, this is not exactly true, and there is a compromise to be found between accuracy and privacy : a small privacy budget ϵ will give good differential privacy but more errors. A low sensitivity Δ makes it possible to use larger values of ϵ and thus to get more accuracy for a given level of privacy.

With fully homomorphic encryption, on the other hand, exact computations can be performed over encrypted data, and the result is the encryption of the answer to the query. If the encryption scheme used has the *correctness* property (see 3.2), the only way to get a wrong result would be because of an error in encryption or decryption, which happens only with negligible probability (in cryptography, this typically means less than $\frac{1}{2^{128}}$ or even less for very secure applications). The same applies to privacy : if the encryption scheme has the *semantic security* property (3.2), one can only distinguish two encrypted elements with a negligible advantage. Thus, the above compromise between accuracy and privacy does not exist, but the drawback is efficiency : state of the art FHE (and even leveled FHE) have very large ciphertext spaces, on which operations can be complex (for example, polynomial multiplication instead of simple multiplication) and some require additional operations such as bootstrapping or relinearization, that are also inefficient. This time, the comprise is between correctness/privacy and efficiency, and it is controlled by the security parameter λ, but low values of λ are not considered acceptable in cryptography.

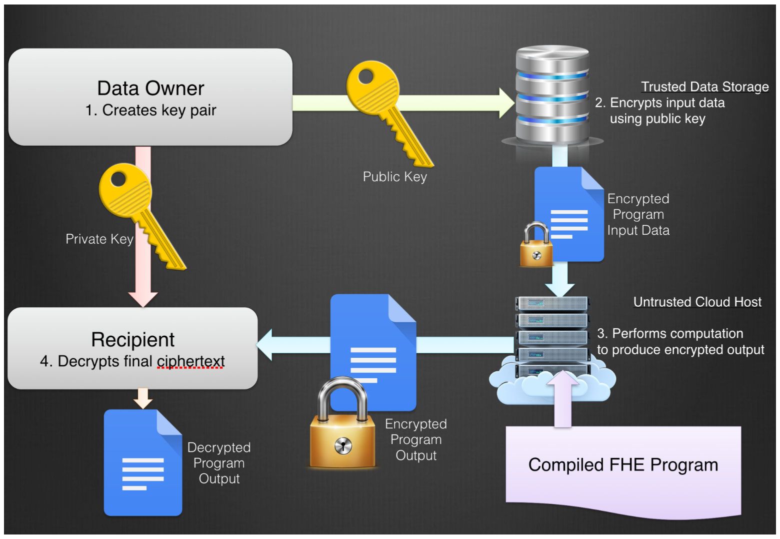

FIGURE 5 – Private computation using FHE

The above arguments suggest that computing with FHE is a very universal (any function can be applied) and secure technique but lacks efficiency, except for simple functions (e.g. with low multiplicative depth), at least for now, whereas differential privacy is more relevant for specific applications with low-sensitivity queries or that need less privacy protection than cryptographic indistinguishability. It is very efficient because nothing changes in the computation after having added the noise to the data. Moreover, the two techniques fundamentally differ in their settings : differential privacy generates noisy data, from which the original data cannot be fully recovered but that must still contain some information about the clean data in order for the computation to work, so if the scheme is not secure enough some information might be learnt by an attacker that uses a denoising algorithm. On the other hand, FHE requires to handle the public and private keys (correctly generate them, safely communicating and storing them), and although it is provably impossible to get information about a ciphertext without the secret key (assuming some NP problem is indeed hard), anyone who gets that key can fully recover the data.

Face recognition (or more generally image recognition) seems to be a suitable case to use differential privacy, because, first, FHE would be very inefficient to use on a database of images and even less with complex AI algorithms such as neural networks, and because the sensitivity should be quite low for similar images such as a database of faces.

5 Conclusion

The above bibliographic study shows that simple anonymization methods are not sufficient to efficiently protect sensible data. There exists several more advanced methods (hardware based security, differential privacy, homomorphic encryption, secure multiparty computation) that makes it possible to get the result of computations without disclosing the input data. Those methods each require very different settings (hardware, random noise, public-key architecture, MPC setting) and have different properties. As cryptographic methods are often inefficient on images databases (typically because they achieve privacy and exact computations) compared to differential privacy, and because we do not have access to secure hardware, we implemented an algorithm to use differential privacy to classify noisy data, first with handwritten digits and then with faces (see the code provided with this report). The results are very good for digits, but the accuracy of the face classifier goes to zero for a level of noise that is not sufficient to hide the identity of the person. This illustrates very well the trade off between accuracy and privacy implied by the differential privacy method. We would probably get better results with a better trained classifier, and with more training data.

All those technique can be used in many applications other than facial recognition. Data could be of different nature than image data which would make it easier to anonymize by adding noise or to encrypt using homomorphic encryption.

Références

[AD18] G. Giannopoulos Athanasios and I. Mouris Dimitris. Privacy preserving medical data analytics using secure multi party computation. an end-to-end use case. 2018.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3) :1–36, 2014.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In *Annual cryptology conference*, pages 505–524. Springer, 2011.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption : Bootstrapping in less than 0.1 seconds. In *international conference on the theory and application of cryptology and information security*, pages 3–33. Springer, 2016.

[CMABK$^+$20] Pathum Chamikara Mahawaga Arachchige, Peter Bertok, Ibrahim Khalil, Dongxi Liu, and Seyit Camtepe. Ppaas : Privacy preservation as a service. *arXiv e-prints*, pages arXiv–2007, 2020.

[CNI19] CNIL. Facial recognition for a debate living up to the challenges. 2019.

[DR$^+$14] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4) :211–407, 2014.

[EFG$^+$09] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In *International symposium on privacy enhancing technologies symposium*, pages 235–253. Springer, 2009.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012 :144, 2012.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[GMDR20] A.Kaissis Georgios, R. Makowski MArcus, Rückert Daniel, and F.Braren Rickmer. Secure, privacy-preserving and federated machine learning in medical imaging. pages 305–311, 2020.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors : Conceptually-simpler, asymptotically-faster, attribute-based. In *Annual Cryptology Conference*, pages 75–92. Springer, 2013.

[KMRR16] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization : Distributed machine learning for on-device intelligence, 2016.

[LLV07] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness : Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer, 2010.

[MH$^+$11] Gregory J Matthews, Ofer Harel, et al. Data confidentiality : A review of methods for statistical disclosure limitation and methods for assessing privacy. *Statistics Surveys*, 5 :1–29, 2011.

[MKGV07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity : Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1) :3–es, 2007.

[Ohm09] Paul Ohm. Broken promises of privacy : Responding to the surprising failure of anonymization. *UCLA l. Rev.*, 57 :1701, 2009.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[RAD$^+$78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11) :169–180, 1978.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978.

[SSW09] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Efficient privacy-preserving face recognition. In *International Conference on Information Security and Cryptology*, pages 229–244. Springer, 2009.

[Swe02] Latanya Sweeney. k-anonymity : A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05) :557–570, 2002.

[YLCT19] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning : Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), January 2019.