Big Data Privacy by Design Computation Platform

Rui Claro[1,2][0000−0003−0176−2720], José Portêlo[2][0000−0002−9816−8957], Miguel L. Pardal[1,3][0000−0003−2872−7300], and Raquel Pinho[2][0000−0002−8862−0356]

[1] Instituto Superior Técnico, Universidade de Lisboa, Portugal
[2] Altran Portugal
[3] INESC-ID Lisboa, Portugal

Abstract. We live in the age of Big Data, and personal user data, in particular, is necessary for the operation and improvement of healthcare services. Many times, the capture and use of personal data are not made explicit to the users, but they are central to the business model of companies. However, each person's right to privacy needs to be respected.
With the goal of reconciling these two conflicting needs, we designed and implemented a proof-of-concept platform for performing privacy-preserving computations. In particular, we implemented privacy-preserving versions of Machine Learning algorithms, namely Decision Trees, k-Means, Logistic Regression, and Support Vector Machines, using Secure Multi-party Computations with Homomorphic Encryption and Garbled Circuits. For each combination of Machine Learning algorithms with Secure Multi-party Computation techniques, we present the reasoning behind our choices and their potential consequences in terms of performance.
The ultimate goal is to provide *Privacy-Preserving Computation as a Service*. With this platform, we wish to contribute to the faster integration of solutions developed by the scientific community in enterprise systems, thus reducing the time required for innovation to reach products used by many people where privacy improvements are urgently needed.

Keywords: Privacy-preserving Computations · Machine Learning · Big Data · Secure-Multi-Party Computations · Privacy-preserving Platform.

1 Introduction

The term *Big Data* means that there are vast amounts of data being analysed and processed by companies every day [7]. Through this data processing, meaningful information can be obtained to improve existing systems or to discover new approaches in business models. Machine Learning (ML) algorithms in the context of Big Data processing can produce significant results, so that it is possible to do knowledge learning from datasets in order to predict future labels (i.e. classes of data) or clusters for new data. An example of this can be seen in the field of Healthcare, where it can be beneficial to analyse patient records from different hospitals in order to identify inefficiencies and develop best practices [8]. For

example, Google Deepmind is developing ML algorithms for faster patient triage and admission processes in hospitals[4], and IBM Watson is supporting medical personnel consider treatment options for their patients[5].

There are restrictions to the processing of personal data, such as patient data. *Privacy* can be defined as the ability or right of an individual to protect his/her personal information, and extends the ability or right to prevent invasions on the personal space of said individual [2]. If patient data can be processed with privacy then they can enable novel applications and scientific breakthroughs in Healthcare. By combining ML algorithms and privacy-preserving techniques, it is possible to create Data Mining processes that, not only allow for knowledge learning on large datasets, but also help maintain a level of privacy desired by individuals and compliant with existing legislation [5].

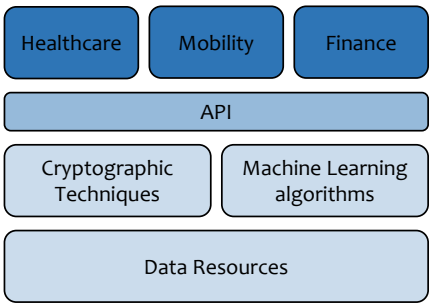


Fig. 1. Conceptual view of the platform.

In Figure 1 we present the conceptual view of our platform. The *data resources* represent the datasets that are used in the classification process. The data processing itself is done using the combination of ML algorithms and cryptographic techniques for performing privacy-preserving computations. The Application Programming Interface (API) layer abstracts details and provides the operations of the platform itself, which allow a simplified building of applications and data visualizations. The use-cases describe the various subjects that can be addressed using this platform, and allow us to place it in real-world scenarios that have high impact and demand in Big Data operations. More use-cases are possible beyond Healthcare, Mobility and Finance, as the platform is designed for general use.

In this work, we present a proof-of-concept platform for privacy-preserving distributed ML computations without resorting to trusted third parties. With it, we aim to give users a platform that provides simplified access to privacy-preserving techniques that can be used to meet privacy requirements in data processing. We provide a detailed comparison of four ML algorithms, namely: Decision Trees (DT), k-Means, Logistic Regression (LR) and Support Vector Ma-

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

chines (SVM); combined with two Secure Multi-party Computations (SMPC): Garbled Circuits (GC) and Homomorphic Encryption (HE); presenting details on how this can be performed. We performed an experimental evaluation of the adapted ML algorithms using publicly available datasets, and compare the results with a baseline.

The paper is further organized as follows. Section 2 describes the ML algorithms and the SMPC techniques used in the platform. In Section 3, we present the related work. Section 4 describes the design of the platform, detailing the adjustments done to ML algorithms. In Section 5, we present the experimental results. Finally, in Section 6 we present the conclusions and propose future work.

2 Background

In this section we present the ML algorithms for which we wish to develop privacy-preserving implementations and the used SMPC techniques.

2.1 Machine Learning Algorithms

Decision Trees (DT): A decision support tool composed of nodes and leaves, with each node representing the decisions to take, and each leaf representing class labels. Classification of a sample is accomplished by traversing the tree from the top, comparing the features selected on each node with its respective threshold, and choosing one branch or the other accordingly, repeating the process until a leaf is reached. At each tree node, a decision is computed using:

$$f_{\mathrm{DT}}(x_i) = x_i \overset{?}{\geq} \theta_j \tag{1}$$

where x_i is the feature value of interest of the testing sample and θ_j is the decision threshold of node j. If the output is 0, the left hand child is selected; if it is 1, the right hand child is selected.

Support Vector Machines (SVM): An SVM model represents the samples as points in space, mapped so that the margin between the two classes is as wide as possible. The vectors that define this margin are called Support Vectors (SVs). The classification of new samples in SVM is done using the scoring function in Equation 2, where each testing sample x is attributed to a prediction label.

$$f_{\mathrm{SVM}}(x) = \sum_{i=1}^{m} \alpha_i K(x_{SV}^{(i)}, x) + b \tag{2}$$

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

***k*-Means:** An iterative algorithm, with two distinct steps. 1) Each instance is assigned to a cluster, by calculating the Euclidean distance, d_E, between that instance and each centroid. Then, the lowest distance indicates which cluster the instance is assigned to. 2) Each centroid is updated to be the mean of all the instances assigned to it. The algorithm stops when the centroids no longer change position. The classification of a new sample is done by computing the d_E of the new sample with each centroid, discovering which is closer. The predicted label of the sample is computed as described in Equation 3:

$$f_{\text{k-M}}(x) = \underset{C}{\operatorname{argmin}}\, d_E(x, C_j) \tag{3}$$

where C are the centroids of each cluster and x is the testing sample.

Logistic Regression (LR): A statistical model that analyses a dataset in order to determine an outcome. This binary LR model is used to estimate the probability of a binary response based on one or more variables. The classification of samples is done using the following equation:

$$f_{\text{LR}}(x) = \beta_0 + \sum_{i=1}^{m} \beta_i x_i \tag{4}$$

where β_0 is the intercept from the linear regression, β_i are each regression coefficient that is multiplied by each feature of the sample, and x is the testing sample.

2.2 Privacy-preserving techniques

Garbled Circuits (GC) [14] allow two mutually mistrusting parties to evaluate a function over their private inputs without resorting to a trusted third party. GC allows two parties holding inputs x and y to evaluate an arbitrary function $f(x, y)$ without leaking any information about their inputs beyond what is inferred from the function output. The idea behind GC is that one party prepares an encrypted version of a circuit that computes $f(x, y)$ and the second party then computes the output of the circuit without learning any intermediate values.

Homomorphic Encryption (HE) [11] is a cryptographic technique that allows computations to be carried with the ciphertext, so that, when decrypted, the resulting plaintext reflects the computation made. In other words, HE allows making some computation over the ciphertext, for example, addition, without decrypting it, and the result is the same as making that computation on the plaintext. This is of great importance because it allows chaining multiple services that make computations on a ciphertext, without the need to expose the data to those services. Homomorphic cryptosystems can be classified into two distinct groups: Partially Homomorphic Cryptosystems (PHE), where there is

only one operation that is allowed by the homomorphic property (ex: addition, multiplication, XOR); and Fully Homomorphic Cryptosystems (FHE), where it is possible to perform both addition and multiplication.

3 Related Work

Although well known Big Data platforms such as Apache Hadoop[6] and MongoDB[7] have been around for a while, most (if not all) of them were designed without Data Privacy concerns in mind. We envision a Big Data platform following a Privacy by Design approach, where data privacy is taken into consideration on every development step.

In many cases there are privacy-preserving versions of ML algorithms, for example, k-Means [12], LR [4] or SVM [13], but they are not made available in a platform.

There are also works on designing platform architectures focused on the protection of privacy in location-based services, and describing privacy-preserving algorithms for them [1], but these works do not actually perform any implementation or testing of such solutions.

4 Platform Design

We structured the platform design in two major parts: a non-privacy-preserving *baseline* and the *privacy-preserving implementation*. The former allows comparing the effects of the privacy-preserving techniques in terms of performance.

4.1 Non-privacy-preserving components

While designing the first part of our platform, the focus was to build a baseline so that meaningful observations could be achieved, while also paving the way to build the privacy-preserving approach. The models that we implemented allowed us to later adapt the prediction step of the ML algorithms for GC and HE, while also giving insight on which technique to use for each algorithm.

4.2 Privacy-preserving components

The privacy-preserving part consisted of adjustments to the evaluation processes of the ML algorithms in order to be compatible with two privacy-preserving techniques: GC and HE. These two techniques offer different means to obtain privacy-preserving computations. GC builds ciphered boolean circuits where most operations are possible to implement. However, arithmetic operations require a large number of logic gates, creating an overhead that makes GC very slow for those operations. So, for some of the ML algorithms, we used an HE system, since it offers arithmetic operations as core operations. The following sections describe the chosen combinations.

[6] http://hadoop.apache.org/
[7] https://www.mongodb.com/

Garbled Circuits and Decision Trees The process of evaluating a DT in a privacy-preserving context is similar to evaluating it in the usual manner, as described in Equation 1. The main differences are: basic operations such as comparisons are replaced with logic gates; and the evaluation of the DT involves evaluating every single node in it, to disclose the least possible information caused by observation of the computations. Figure 2 shows the computations done inside each node of the DT.

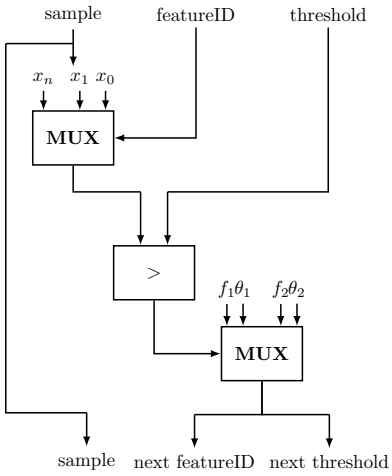
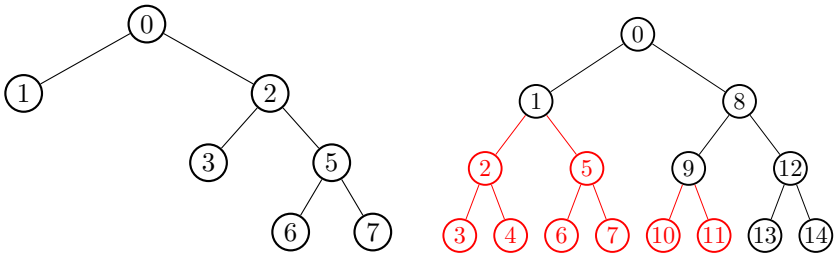


Fig. 2. Boolean circuit of each node in a DT.

Another aspect to mention is that the trees are always complete, i.e., the number of nodes n is always the maximum possible, and can be defined as $n = 2^{h+1} - 1$, where h is the *height* of the tree. Even though in most cases this will lead to an exponential increase of the number of nodes with increasing tree depth, we feel this is necessary to prevent information leaks due to an attacker being able to know the different path depths. Figure 3 shows the implications of this expansion.

Garbled Circuits and *k*-Means The process of evaluating the k-Means algorithm in a privacy-preserving manner is similar to evaluating in the usual manner. The operations in the prediction step of the algorithm were transformed into boolean circuits, with logic gates representing operations. In Figure 4 we show the circuit we have designed to represent the k-Means prediction, where d_E represents the Euclidean distance between testing sample x and each centroid C_i.

Homomorphic Encryption and Logistic Regression In order to use a FHE system, the prediction function for LR described in Equation 4 must be



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

Fig. 3. Expansion of binary trees.

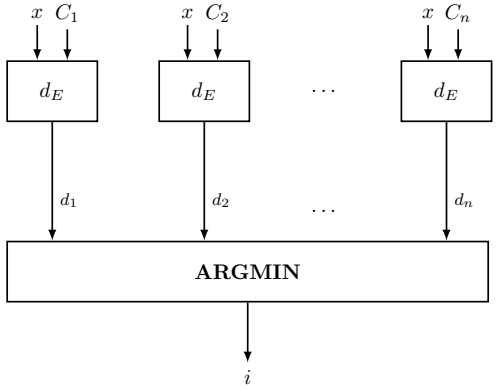


Fig. 4. Boolean circuit of the k-Means prediction.

converted to:

$$f_{\text{LR,FHE}}(x) = D_k\left(E_k(\beta_0) + \sum_{i=0}^{m} E_k(\beta_i) \cdot E_k(x_i)\right) \tag{5}$$

where E_k represents the encryption operation and D_k represents the decryption operation using the key k.

Converting Equation 5 to be computed using a PHE system is straightforward, but this can only be done under two assumptions: 1) the data to be evaluated (x) and the model parameters ($\beta_0, \beta_1, \ldots, \beta_m$) must come from two different parties, and 2) the owner of the model parameters must be the one processing the data. Under these assumptions, the linear prediction function for a additive PHE system becomes:

$$f_{\text{LR,PHE}}(x) = D_k\left(E_k(\beta_0) \cdot \prod_{i=1}^{m} E_k(x_i)^{\beta_i}\right) \tag{6}$$

Homomorphic Encryption and Support Vector Machines For the SVM algorithm, we only considered the linear kernel, as it simplifies the scoring function. The Equation 2 is then simplified to the following:

$$f_{\text{SVM}}(x) = \sum_{i=1}^{m} \alpha_i x_{SV}^{(i)} x + b = \sum_{i=1}^{m} \alpha_i \sum_{j=1}^{n} x_j x_{SV}^{(i,j)} + b \tag{7}$$

To compute this function using a FHE system, we must convert it to:

$$f_{\text{SVM,FHE}}(x) = D_k \left(\sum_{i=1}^{m} E_k(\alpha_i) \cdot \sum_{j=i}^{n} E_k(x_j) \cdot E_k(x_{SV}^{(i,j)}) + E_k(b) \right) \tag{8}$$

where E_k represents the encryption operation and D_k represents the decryption operation using the key k.

Like before, converting it to be computed using a PHE system is equally straightforward, and under the same two assumptions, the scoring function for a additive PHE system becomes:

$$f_{\text{SVM,PHE}}(x) = D_k \left(\prod_{i=1}^{m} \left(\prod_{j=1}^{n} E_k(x_i)^{x_{SV}^{(i,j)}} \right)^{\alpha_i} \cdot E_k(b) \right) \tag{9}$$

4.3 Architecture

The combination of the components above helped us create a data processing architecture for a privacy-preserving ML platform, presented in Figure 5.

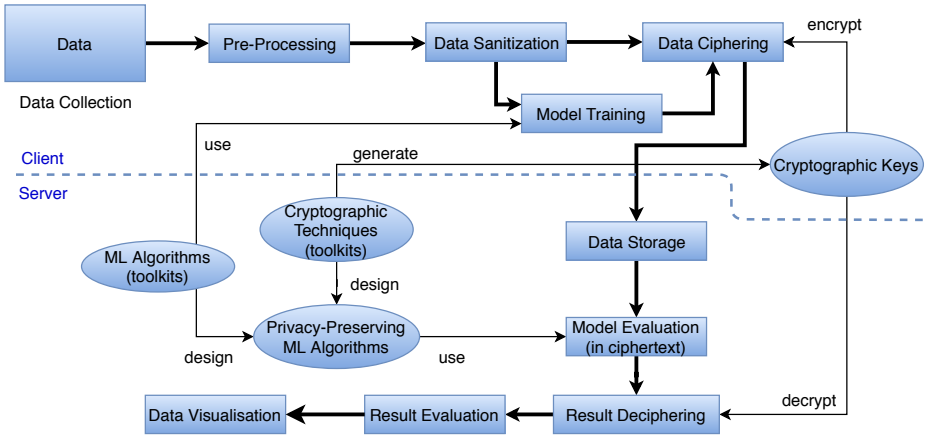


Fig. 5. Data processing architecture for the platform.

We assume that only two parties exist: the client and the server. The client represents a user or an individual who owns data and wishes store them and

to perform some processing over them, but does not have the capabilities to do so (e.g.: real-time processing, fault tolerance systems, scalable environment). Despite this, the client wishes to keep these data private. The server represents a cloud or service provider who has the capabilities to perform such processing. To achieve the privacy goals of both parties, the client pre-processes, sanitizes and encrypts the data and models before sending them to the server. The model training is performed in the usual manner. The model evaluation process is the main focus of our work, and it is where the privacy-preserving techniques are deployed. At the end of the flow, the platform produces the prediction results.

With this architecture, we aim at providing companies a way to integrate their Big Data systems processes with privacy-preserving ML algorithms, allowing them to provide additional data privacy guarantees to their clients.

5 Experimental Results

This section presents the evaluation results. The objective of the experimental evaluation is to answer two important questions: 1) How accurate is the prediction *versus* the baseline system? 2) How easily can the platform be adapted to different size and context of the datasets?

The datasets were split into three sets: training (70%), validation (15%) and testing (15%) sets. The training step of the baseline ML algorithms was performed using the scikit-learn toolkit for Python[8]. The GC results were obtained using the VIPP toolkit [10]. The results using FHE were obtained using the HElib toolkit [6]. The results using PHE were obtained using our own implementation of the Paillier cryptosystem [9].

For running the experiments, we used the datasets listed in Table 1. They are widely used in the literature.

Table 1. The datasets used in the evaluation.

Dataset	Subject	Instances	Features
Pima Indians Diabetes[9]	Healthcare	768	8
Breast Cancer Wisconsin[10]	Healthcare	569	30
Credit Approval[11]	Finance	690	15
Adult Income[12]	Governance	48842	14

[8] http://scikit-learn.org/
[9] https://www.kaggle.com/uciml/pima-indians-diabetes-database
[10] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
[11] http://archive.ics.uci.edu/ml/datasets/credit+approval
[12] https://archive.ics.uci.edu/ml/datasets/adult

5.1 Accuracy

After analysing the results obtained with the privacy-preserving ML algorithm implementations using GC, we verified that changing the number of bits for the actual numeric precision of the data and model parameters affects the accuracy of the results. The absolute error percentage values for the experiments on DT and k-Means are presented in Tables 2 and 3, respectively. It is to be noted that this error is computed versus the baseline prediction results, not the prediction labels from the dataset.

Table 2. GC+DT. Average absolute label prediction error vs. the baseline.

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

Table 3. GC+k-Means. Average absolute label prediction error vs. the baseline.

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

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

After analysing the results obtained using the PHE and FHE systems, we verified that all predicted labels and almost all function evaluation outputs match the baseline. The few examples when an exact match does not happen come mostly from the SVM scoring evaluation function implemented in HElib, and are most likely caused by the accumulation of the intrinsic noise generated every time an operation is performed between two ciphertexts. Therefore, we can conclude that our privacy-preserving versions of the ML algorithms using PHE and FHE have no relevant loss of prediction performance.

5.2 Discussion

Although we did not compare the performance of GC and HE directly, for instance by choosing a ML algorithm and implementing it using both privacy-preserving techniques, it is clear that the HE approach is adequate for ML algorithms that rely on arithmetic operations, and the GC approach is adequate for ML algorithms that rely on non-arithmetic operations.

We did not perform a detailed computational times analysis because we feel that an evaluation of execution times is less relevant, as it is extremely dependent on the hardware and toolkits used (which evolve through time), while the evaluations we performed in terms of accuracy are not. We are, however, aware of very efficient GC implementations for evaluating DT, but many of them do not scale adequately with the DT size [3]. We did not consider such implementation because we experimented with fully expanded DT of considerable depth.

An important remark on our experiments with GC is related to our choice to only analyse fully expanded DT instead of the original ones, in order to prevent any information leakage regarding the shape of the original tree. However, in most cases this causes an exponential growth of the number of nodes with increasing tree depths, leading to proportional increases in both the execution times and the communication costs.

Another important conclusion made possible by our experiments with HE is *when* each of the techniques should be used. We verified that PHE is, in fact, usable in practice but under some restrictions (e.g.: if there is no need for complex composition of operations and if data is separated between client and server), while FHE is more flexible but still too computationally expensive.

With our implementation, we were able to understand that, despite the fact that GC and HE are very different techniques, they can be used in almost the same manner. The main difference is that the ML algorithms must be adapted differently for each one. The tweaks done to the algorithms presented in Section 4.2 allowed us to implement privacy-preserving versions of them and running them in the same manner as the non-privacy-preserving approach.

We were also able to produce results with datasets from varied contexts, such as Healthcare or Finance, and of very different sizes, without the need to specifically adapt the algorithms for them. With this, we have shown that the platform can be used for different application domains.

6 Conclusions and Future Work

This paper presented a platform to perform privacy-preserving ML computations to be applied in Big Data applications. We discussed the existing techniques that provide the level of privacy compliance with the laws in force and matched those techniques with the most commonly used ML algorithms. We evaluated the solution by comparing two SMPC techniques: GC and HE. Overall, we produced a proof-of-concept platform that provides a unified and simplified API for privacy-compliant ML. This shortens the distance between the scientific community that develops the techniques and the companies that employ them in products that

impact many people. With our approach, the most recent scientific advances in privacy-preserving technologies can be applied faster in enterprise applications.

For future work, we propose the following points to enhance the functionalities of the platform and its performance: extend the platform to work with more ML algorithms (ex: Neural Networks or Naive Bayes), so that the platform can be used for more purposes (ex: Deep Learning); optimize the SMPC techniques used, to improve the performance of the platform; implement and test the SMPC techniques using other toolkits, also to improve the performance of the platform.

7 Acknowledgements

Work supported by Portuguese national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 (INESC-ID).

References

1. Abbas, F., Hussain, R., Son, J., Oh, H.: Privacy preserving cloud-based computing platform (ppccp) for using location based services. In: Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing. pp. 60–66. IEEE Computer Society (2013)
2. Anderson, R.: Security engineering. John Wiley & Sons (2008)
3. Bost, R., Popa, R.A., Tu, S., Goldwasser, S.: Machine learning classification over encrypted data. In: NDSS. vol. 4324, p. 4325 (2015)
4. Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. In: Advances in Neural Information Processing Systems. pp. 289–296 (2009)
5. D'Acquisto, G., Domingo-Ferrer, J., Kikiras, P., Torra, V., de Montjoye, Y.A., Bourka, A.: Privacy by design in big data: An overview of privacy enhancing technologies in the era of big data analytics. European Union Agency for Network and Information Security (2015)
6. Halevi, S., Shoup, V.: HElib-an implementation of homomorphic encryption. Cryptology ePrint Archive, Report 2014/039 (2014)
7. Lee, I.: Big data: Dimensions, evolution, impacts, and challenges. Business Horizons **60**(3), 293–303 (2017)
8. Lu, R., Zhu, H., Liu, X., Liu, J., Shao, J.: Toward efficient and privacy-preserving computing in big data era. IEEE Network **28**(4), 46–50 (2014)
9. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 223–238. Springer (1999)
10. Pignata, T.: Garbled circuit designer and executer from the visual information processing and protection (vipp) research group (2012), `http://clem.dii.unisi.it/~vipp/index.php/software/135-garbledcircuit`
11. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. Foundations of secure computation **4**(11), 169–180 (1978)
12. Upmanyu, M., Namboodiri, A.M., Srinathan, K., Jawahar, C.: Efficient privacy preserving k-means clustering. In: Pacific-Asia Workshop on Intelligence and Security Informatics. pp. 154–166. Springer (2010)
13. Vaidya, J., Yu, H., Jiang, X.: Privacy-preserving svm classification. Knowledge and Information Systems **14**(2), 161–178 (2008)
14. Yao, A.C.C.: How to generate and exchange secrets. In: Foundations of Computer Science, 1986., 27th Annual Symposium on. pp. 162–167. IEEE (1986)