A Study on Privacy Preserving Machine Learning with Homomorphic Encryption

Kazi Amit Hasan
Queen's University
Kingston, Canada
kaziamit.hasan@queensu.ca

ABSTRACT

Machine learning is being used in sectors from different domains, it often needs to deal with data that are extremely confidential. When sensitive data is used to train a machine learning model, the model's reliance on sensitive user data renders it unsuitable for creating Machine Learning workflows without compromising user privacy and confidentiality. These considerations apply to each and every machine learning method or model that deal with sensitive data. According to the findings of this project, we recommend the adoption of a customized homomorphic encryption scheme as a means of mitigating this danger. This scheme will enable proper encryption of user data by making use of a combination of public and private keys. In this project, I have explored different types of homomorphic encryption schemes. Also, I experimented the scheme in a sensitive dataset using linear regression. It has been noted that the encryption does a good job of preserving the confidentiality of the input test data as well as the data that corresponds to the results of the regression model. It protects the machine learning model and any sensitive user data that is linked with it from attacks involving model inversion as well as membership inference. All the codes associated with this project are publicly available at https://github.com/AmitHasanShuvo/CISC-870-Project

KEYWORDS

homomorphic encryption, machine learning, privacy

ACM Reference Format:
Kazi Amit Hasan. 2022. A Study on Privacy Preserving Machine Learning with Homomorphic Encryption. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Final Project Report for CISC 870).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Machine learning (ML) is a subset of artificial intelligence (AI) techniques that, instead of solving a specific problem, learn how to solve a set of problems that are similar. The purpose of machine learning is to automate some or all of the solutions to complex problems in a variety of fields. In recent times, machine learning (ML) has been implemented in various sectors including industry and academia. The expanding usage of machine learning in numerous fields has also increased the importance of enhancing its privacy [10].

A typical machine learning system workflow consists of several essential steps including data collection, performing exploratory data analysis (EDA), implementation of a machine learning model, evaluating the model, and making decisions based on model. Figure 1 represents a high level overview of a typical machine learning workflow.

Figure 1: A high level workflow of a machine learning system.

As machine learning is being used in sectors from different domains, it often needs to deal with data that are extremely confidential. For example, in order to build an early disease prediction model, a model must require healthcare related data, which needs to be treated very carefully to avoid any unwanted security issues. In addition to it, machine learning applications related to the financial sector deal with financial data, which is extremely sensitive as it contains information regarding individuals. Any security oriented attacks on financial data can create big problems if they aren't handled carefully. This represents the importance of security and privacy related issues in machine learning domain. Therefore, when sensitive data is used to train a machine learning model, the model's reliance on sensitive user data renders it unsuitable for creating Machine Learning workflows without compromising user privacy and confidentiality. These considerations apply to each and every machine learning method or model that deal with sensitive data. Moreover, as the usage of machine learning models is becoming more popular, they become increasingly susceptible to privacy breaches.

Model inversion and membership inference attacks are examples of privacy-focused machine learning model attacks. A malevolent person attempts to recover the confidential dataset used to train a supervised neural network in model inversion attacks. A successful model inversion attack should generate realistic and diverse samples that accurately represent each class in the private dataset [14]. A membership inference attack enables an attacker to query a trained machine learning model to determine whether or not a certain sample was used for the model's training dataset [3]. These are legitimate dangers to machine learning frameworks that attempt to secure all aspects of user privacy. The reliance of machine learning models on datasets is substantial. I am considering linear

Final Project Report for CISC 870, December 09, 2022, Kingston, Ontario, Canada

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

regression for this project, as it is a data-intensive machine learning approach. Linear regression trains the model based on the data and then drives model-based inference. If the data being supplied to the machine learning model is sensitive, then it will be impractical to employ machine learning models efficiently, since privacy will be compromised. If an adversary attacks a trained model with techniques such as "model inversion" and "membership inference," the adversary could get access to confidential data, which is a major concern. While we are concentrating on privacy concerns in machine learning models exclusively, there is another important issue that must be addressed, and that issue is the performance of the model with keeping privacy in mind. In addition, if we try to protect the privacy of training data by restricting access to sensitive data attributes in the machine learning model, we may end up putting the effectiveness of the machine learning model in jeopardy. As a result, we are faced with a choice between protecting the users' data privacy and maximizing the effectiveness of the data that we can provide to machine learning models.

Now, at this point, we have discussed about privacy-oriented attacks on machine learning models. One possible approach can be considered in order to eliminate attacks on the data in accordance with machine learning models, such as model inversion, we must encrypt not just the input training data, but also the output. Consequently, the adoption of homomorphic encryption algorithms is a potential method for protecting user privacy when gaining knowledge from user data utilizing machine learning technologies. Homomorphic encryption (HE) is a type of public encryption that is regarded as one of the most effective methods available. It is used to encrypt sensitive data by carrying out operations such as incremental addition and multiplication on the data. This encrypts the data in such a way that it enables us to apply any machine learning technique to data sets in an effective manner while still preserving user privacy. Homomorphic encryption enables us to encrypt the user data before sending it to the model, perform operations and evaluations on the encrypted data using the machine learning model, and finally decrypt the data to evaluate the results of the computation on the original user data. All of this can be accomplished without compromising the security of the data in any way [12]. Figure 2 demonstrates the overall high level workflow of homomorphic encryption. The steps are:

(1) Firstly, the client's machine encrypts the sensitive data using a public key and transmits it.
(2) Then, the machine-learning algorithms process the encrypted data and extract the necessary insights.
(3) After that, the outputs of the machine learning model are sent in an encrypted form.
(4) Lastly, the client can decrypt the data using a private key to view the results of the analysis, in a format that is compatible with the original format of the sensitive user data.

In this way, the confidentiality of data can be preserved in a machine learning model.

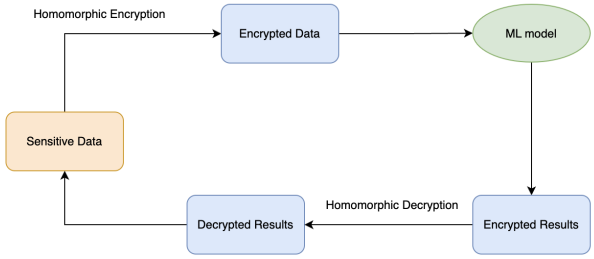

Figure 2: A high level workflow of a Homomorphic Encryption

In this project, our overall goal to learn about homomorphic encryption and it's use cases while preserving privacy of sensitive data. In particular, I want to explore different types of homomorphic encryption schemes and their use cases in ML. Also, I want to experiment and evaluate a selected approach that can ensure user privacy and model performance using HE.

In summary, I intend to answer the following research questions:

- **RQ1: What types of HE schemes are available in literature?**
- **RQ2: Are HE schemes efficient for sensitive data while preserving privacy?**

The **RQ1** would help me to understand different types of homomorphic encryption schemes along with their advantages, disadvantages and uses cases available in literature.

The **RQ2** would help me to evaluate the selected homomorphic encryption scheme while dealing with sensitive data in any machine learning workflow.

All the codes regarding this project can be found at: https://github.com/AmitHasanShuvo/CISC-870-Project

The project report is structured as follows: Section 2 presents the background and related works. Section 3 represents the tools and libraries that I used to do this project. Section 4 discusses about the detailed overview of my project. This section includes my implementation of HE algorithm from scratch and an application of implementing HE. Section 5 discusses about the limitations of FHE and our project. Section 6 contains the conclusion of my work.

2 BACKGROUND AND RELATED WORK

As the title of this project indicates, it is focused on homomorphic encryption and machine learning, with a focus on preserving privacy. So, I designed this section in such a way that it gives enough background and information to understand the overall objectives/goals of this project.

This section consists of two subsections. The first subsection deals with homomorphic encryption and its different schemes. This also provides answers to the research question that I mentioned before. The second subsection deals with homomorphic encryption's machine-learning domain.

As per the project guidelines, I have to demonstrate full understanding of at least four papers in this section. So, in order to fulfill this requirement, I am categorizing the papers based on my project

Table 1: A quick overview of reviewed academic papers.

Papers	HE Scheme	ML approches	Key Takeways
[9], [10], [13], [16]	HE, FHE	Yes	HE schemes, FHE in ML, open source implementation listing
[14]	HE, PHE	-	History of HE, PHE
[17]	HE, SHE	-	History of HE, SHE
[13]	HE	-	HE, FHE
[6]	HE	-	Performance of BFV & CKKS scheme
[1], [2], [4], [6]	HE, FHE	-	BGV, BFV, CKKS, performance comparison

structure. In order to understand the homomorphic encryption and it's variants, I had to go through several research papers. Table 1 represents a quick overview of the academic papers that I read extensively to do this project.

2.1 Homomorphic Encryption

This section discusses the fundamental principles of homomorphic encryption (HE) schemes and offers significant researches in a variety of research areas that have evolved from this methodology.

Homomorphic encryption (HE) is a type of public encryption that is considered as one of the most effective methods available. It is used to encrypt data by performing operations such as addition and multiplication on the data. He performs actions on cipher texts using only publicly available information, specifically without access to any secret key. Homomorphic refers to the relationship between the space of messages and the space of cipher texts. Thus, actions performed on cipher texts correspond to operations performed on the messages they encrypt. [12].

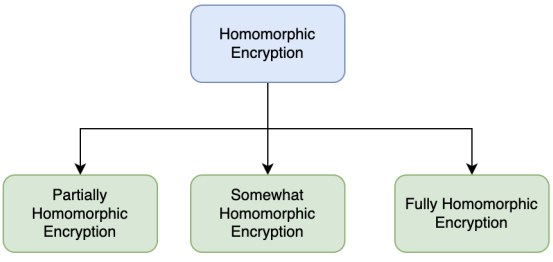

Figure 3: Different schemes of homomorphic encryption

In this context, Rivest et al. introduced the concept of homomorphic encryption [13]. This is refereed as **partially homomorphic encryption (PHE)**. Only a single form of mathematical operation is permitted on the encrypted message when the PHE scheme is being used. If the operation performed on cipher texts results in the encrypted form of the sum of the plain texts corresponding to those cipher texts, then the homomorphic scheme is referred to as additive. On the other hand, if the operation results in the encrypted form of the product of the plain texts, then the homomorphic scheme is referred to as multiplicative, but with an limited number of times [6]. In order to ensure a particular level of confidentiality, the HE encryption process requires adding an error term. Later **somewhat homomorphic encryption (SHE)** was introduced [16]. Although the SHE algorithm supports both addition and multiplication operation in ciphertexts, in practice, it only allows for a limited number of such operations to be performed.

The first **fully homomorphic encryption (FHE)** scheme was introduced, which marked a significant milestone in the history of cryptography's long and illustrious history [7]. This was accomplished by applying bootstrapping to somewhat homomorphic encryption. Performing an excessive number of multiplications requires a costly bootstrap procedure to decrypt the data, allowing for additional multiplications. This technique did not, however, lend itself well to practical applications; hence, this has been the focus of substantial research in order to get an encryption scheme that can be applied to practical applications in industries where privacy is a major concern [6]. When compared to a SHE scheme, an FHE scheme is more flexible because it does not place a limit on the number of homomorphic operations that can be performed. [12]

Now let's discuss about the encryption techniques available in FHE:

2.1.1 BGV Scheme: Brakerski et al. proposed that the key to their work is the construction of leveled Fully Homomorphic Encryption Schemes without the use of Bootstrapping techniques. This is the central principle that underpins their work. Lattice-based cipher texts can be properly managed in terms of their noise level using this technique [2].

2.1.2 BFV Scheme: Brakerski et al. developed a novel tensoring method for LWE-based completely homomorphic encryption. As a consequence of using this method, noise increases in a linear fashion, as opposed to the quadratic development that was observed in earlier methods [1].

In addition, Fan-Vercauteren et al. presented two improved variants of the re-linearization process. These variants not only provide more compact re-linearization keys, but they also make calculations more quickly. In the process of re-linearization, quadratic equations are converted into linear ones [5].

2.1.3 CKKS Scheme: The authors explain that the approach is mostly utilized for performing operations that are approximative to arithmetic. According to the authors, their decryption structure generates a result that is approximately equivalent to plain text and has a predetermined level of precision [4].

When compared to BFV and CKKS, BGV significantly better levels of efficiency However, its application is more challenging. They are identical both in terms of addition and multiplication. In both

the BGV and BFV models, it is possible to execute computations on integers. CKKS is also capable of carrying out computations on floating point numbers, although with a reduced level of precision.

At this point, this section gives us enough knowledge about homomorphic encryptions and its different schemes. **This is serves as the answer of our first research question.**

2.2 Homomorphic Encryption in Machine Learning Domain

This section discusses about the literature review of homomorphic encryption with a focus on machine learning.

Figure 2 presents a high-level overview of HE with the primary focus being placed on machine learning. It encrypts sensitive data by carrying out operations such as incremental addition and multiplication on the data. This encrypts the data in such a way that it enables us to apply any machine learning technique to data sets in an effective manner while still preserving user privacy. Homomorphic encryption enables us to encrypt the user data before sending it to the model, perform operations and evaluations on the encrypted data using the machine learning model, and finally decrypt the data to evaluate the results of the computation on the original user data. All of this can be accomplished without compromising the security of the data in any way [12].

Within the field of fully homomorphic encryption, Pulido et al. offered a detailed overview of overall concepts, state-of-the-art methods, limits, and potential applications. The authors made an effort to provide a comprehensive summary of FHE and machine learning, both from a theoretical and a practical standpoint. Because of this research, I now have a better understanding of the foundations of FHE as well as its application cases in the workflows of machine learning. The survey article provided some helpful information when it pointed out the three primary routes that could be taken by FHE in the future [12].

Shereen Mohamed Fawaz et al. demonstrated the performance of the BFV and CKKS schemes by using Microsof SEAL. They did this by applying the arithmetic operations of addition, multiplication, and squaring, and they monitored the amount of time that was used for each function that was applied in each scheme. In addition to that, several levels of homomorphic encryption techniques were described in this study. It was a big help in understanding the entire algorithm of the homomorphic encryption, which was something I had to build from the scratch [6].

Kim et al. demonstrated a secure technique for developing machine learning models using encrypted data that was based on homomorphic encryption. They proposed a workable technique for homomorphic encryption that was able to demonstrate both high performance and low storage costs. This paper served as the foundation for our machine learning application that was built utilizing HE [10].

An investigation into fully homomorphic encryption and its uses in the medical and bioinformatics fields was carried out by Wood et al. Because of this research, we now have an overview of homomorphic encryptions and the many algorithms they use. In addition to that, they provided a listing of a significant number of open source implementations of homomorphic encryption algorithms, which was of great assistance to us when we were working on this project [15].

Now, at this point, we have enough information about homomorphic encryptions and it's use cases in machine learning. We intend to investigate the viability of employing homomorphic encryption approaches to execute mathematical operations on sensitive data sets within a given machine learning workflow using homomorphic encryption. We will pay a great deal of attention to ensuring that the fundamental algebraic operations for machine learning models may be executed efficiently without allowing access to the raw sensitive user data.

3 TOOLS AND LIBRARIES

3.1 Pyfhel for Homomorphic Encryption

I have used Pyfhel library in order to implement my approach [9]. While reviewing literature, I came across a few open source implementation of homomorphic encryption libraries. They are listed in the Table 2 along with their supported schemes.

Table 2: Libraries which support different schemes of homomorphic encryptions. [6]

Libraries	Supported Schemes
Microsoft SEAL	supports the BFV and CKKS schemes
HElib	supports the CKKS and BGV schemes as well as bootstrapping
PALISADE	supports numerous homomorphic encryption algorithms with multiparty support, including BGV, BFV, CKKS, TFHE, and FHEW

Pyfhel was chosen for this project because it serves as a wrapper for the many capabilities that are included in well-known open-source homomorphic encryption libraries such as Microsoft SEAL, PALISADE, and HELib, among others.

3.2 Other Libraries

Apart from Pyfhel, I have also used scikit learn [11], numpy [8] library in order to achieve the goals that I discussed in previous sections.

3.3 Working Environment

For this project, I have used google colaboratory to run the codes and analysis that I learned from the literature. I tried to Pyfhel in my personal machine, but it was not supported as per the official documentation.

4 DETAILED OVERVIEW OF PROJECT

I have divided my entire approach into several sections. Section 4.1 discusses about the reason behind selecting the fully homomorphic encryption technique instead of others. Section 4.2 discusses

about the scratch implementation of homomorphic encryption using Python. This work was done in order to learn the HE in an effective way. Section 4.3 discusses about the implementation of FHE in a sensitive dataset with the help of ML. In this section, we will see how the prediction performs if we use HE. **This section answers the second research question that I established in previous section.** The codes related to this section can be found at : https://github.com/AmitHasanShuvo/CISC-870-Project

4.1 Fully Homomorphic Encryption

In order to get a head start on the project, I started by researching the various iterations of the homomorphic encryption system. I became aware of the potential of using homomorphic encryption schemes as a feasible technique while I was conducting research into the relevant academic literature for the purpose of finding a solution to the problem of preserving privacy while applying machine learning techniques to any sensitive data. The potential of homomorphic encryption to permit calculations on encrypted data via utilization of customised algebraic expressions appeared to be a game-changer for our problem and instantly made it more favorable for us in comparison to conventional encryption and decryption approaches. After I had come to the conclusion that using homomorphic encryption as my chosen method to protect sensitive data was the best course of action, the next obvious progression for me was to choose a specific variant of homomorphic encryption that I would be implementing. Once I had made this decision, I would be able to move on to the next step. I discovered three viable homomorphic encryption techniques when conducting our literature review. These strategies are fully homomorphic encryption, partially homomorphic encryption, and somewhat homomorphic encryption. I could utilize any one of these schemes to solve our problem. During my investigation, I observed that a partially homomorphic encryption method permits only one operation to be performed on encrypted data, either addition or multiplication. This technique does not restrict the number of times we can conduct mathematical operations on ciphertext. In comparison, the Somewhat Homomorphic Encryption method permits a subset of operations (addition or multiplication) up to a given level of complexity. Furthermore, the technique restricts the number of times these computations can be performed.

Consequently, I discovered the optimal solution to our problem within the Fully Homomorphic Encryption (FHE) scheme, an improved version of the Somewhat Homomorphic Encryption (SHE) scheme. FHE is capable of utilizing addition and multiplication an infinite number of times and is therefore a potential method for conducting secure multi-party computing efficiently. In addition, it is capable of performing arbitrary computations on ciphertexts, a capacity that is exclusive to this algorithm.

4.2 Implementing Homomorphic Encryption from Scratch

One of the main objectives of this project was to learn the cryptography related algorithms that were not covered in the class. This is why I tried to learn the homomorphic encryption algorithm by implementing it from scratch. My implementation is capable of performing fundamental operations such as key creation, encryption, decryption, and computations in the form of addition and multiplication.

(1) **Key Generation:** This process includes the generation of public and secret keys. Encryption is performed with the help of the public key, while decryption requires the private key.
(2) **Encryption:** This process includes the encryption process using public key.
(3) **Decryption:** This process includes the decryption process using private key.
(4) **Computation:** For this stage, I have experimented with addition and multiplication.

To test my implemented scheme, I considered two numbers as plain text and encrypted them using algorithm. Then I performed computations like addition and multiplication on it. The output of this method is shown in Figure 4.

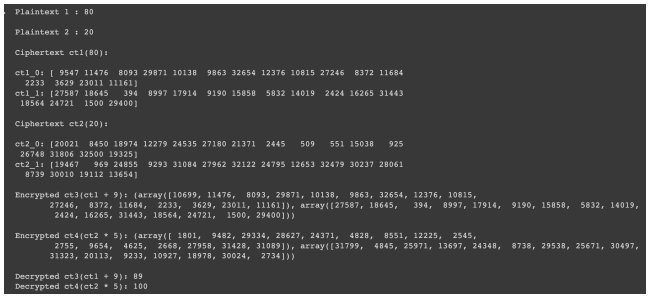

Figure 4: Output of the implementation of HE from scratch.

4.3 Applying Fully Homomorphic Encryption within a Machine Learning Model

As I discussed in my previous sections that homomorphic encryptions are used while handling sensitive data. For this project, I am considering a medical dataset as medical datasets are extremely sensitive. The dataset that I am using for this project is diabetes dataset for using HE on linear regression model [11]. This dataset is publicly available in scikit learn library [11]. It has ten different features of 442 diabetes patients. I am considering linear regression as a machine learning model for his project as it's heavily dependent on data.

As the primary objective of this project is to properly understand HE, I am concentrating more on HE principles in ML process. The problem statement can be formulated as follows: we are trying to predict the future progression of the disease. This makes it a linear regression problem.

After performing exploratory data analysis, I splited the entire dataset into training and testing sets. Due to the short size of the dataset, I regard 9.5% samples as test samples and the remaining samples as training samples. After that I performed the basic model fit operations on the data. In this project, I am considering body mass index (BMI) in order to predict the disease. After fitting the machine learning model, I got the he coefficients and intercept which helped me to create the mathematical equation of our model.

```
def predict(bmi):
  coeff = 941.43097333
  intercept = 153.3971362333169
  diabetes_progression = (coeff * bmi) + intercept
  return diabetes_progression
```

Figure 5: Formulating the mathematical equation.

```
# predict body mass index using encryption
bmi1 = -0.0730303
ctxt1 = HE.encryptFrac(bmi1)
dp_encrypted = predict_encrypted(ctxt1)
dp = HE.decryptFrac(dp_encrypted)
print(dp)

84.64414984895848

[18] # predict body mass index without encryption
     predict(-0.0730303)

     84.644149821735
```

Figure 6: Comparing the BMI using HE and regular approach.

Python for Homomorphic Encryption Libraries, abbreviated as Pyfhel, is selected for this project (reason described in Tools and libraries section). The library permits addition, subtraction, multiplication, and scalar product on encrypted vectors or scalars of integers and binary data. The body mass index (BMI) for which I would like to estimate diabetes progression is regarded confidential information. The encrypted data is provided for prediction to the model. The model replaces encrypted data in the linear equation for the evolution of diabetes with encrypted data and sends encrypted data back. The user can then decrypt the obtained disease progression. Because of this, the privacy of sensitive medical data pertaining to the user, such as their body mass index (BMI) and the obtained diabetes disease progression, is preserved through the use of homomorphic encryption in this scenario.

I have tested the model with respect to HE and regular approach. The body mass index (BMI) calculated with and without the use of HE is shown in Figure 6. It is shown in figure 6 that the values are identical. In addition, I was able to acquire a identical R2 score while assessing the effectiveness of the machine learning model.

5 LIMITATIONS

There's limitations of this project. As I had to update my project proposal after first submission, this caused a bit of unwanted delays. However, I tried my best to make this project as organized as I can within this short time. I intended to apply more machine learning algorithms and wanted a do a comparative analysis on it. But due to time limit constraints, I limited my experiment to linear regression only.

Apart from it, there are several limitations of FHE. It is not possible to have two encryption and decryption schemes run at the same time with separate keys on same dataset. FHE becomes infeasible while handling complex mathematical equations. These issues needs to be addressed to build a secure machine learning model.

6 CONCLUSION

In this project, I have investigated the use of homomorphic encryption in the domain of machine learning to preserve privacy. Also, I studied various types of homomorphic encryption and fully homomorphic encryption systems. Additionally, I developed one of the FHE schemes in Python and tested it. Then, I used homomorphic encryption to the linear regression model for the diabetic data set. Homomorphic encryption is a potential method for constructing machine-learning applications for a number of sectors in a secure manner. This project helped me a lot to learn about different homomorphic encryption methods. It was fun to explore the machine learning side of it.

REFERENCES

[1] Zvika Brakerski. 2012. Fully homomorphic encryption without modulus switching from classical GapSVP. In *Annual Cryptology Conference*. Springer, 868–886.

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

[3] Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramer. 2022. Membership inference attacks from first principles. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1897–1914.

[4] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *International conference on the theory and application of cryptology and information security*. Springer, 409–437.

[5] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *Cryptology ePrint Archive* (2012).

[6] Shereen Mohamed Fawaz, Nahla Belal, Adel ElRefaey, and Mohamed Waleed Fakhr. 2021. A Comparative Study of Homomorphic Encryption Schemes Using Microsoft SEAL. In *Journal of Physics: Conference Series*, Vol. 2128. IOP Publishing, 012021.

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

[8] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with NumPy. *Nature* 585, 7825 (2020), 357–362.

[9] Alberto Ibarrondo and Alexander Viand. 2021. Pyfhel: Python for homomorphic encryption libraries. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. 11–16.

[10] Andrey Kim, Yongsoo Song, Miran Kim, Keewoo Lee, and Jung Hee Cheon. 2018. Logistic regression model training based on the approximate homomorphic encryption. *BMC medical genomics* 11, 4 (2018), 23–31.

[11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[12] Luis Bernardo Pulido-Gaytan, Andrei Tchernykh, Jorge M Cortés-Mendoza, Mikhail Babenko, and Gleb Radchenko. 2021. A survey on privacy-preserving machine learning with fully homomorphic encryption. In *Latin American High Performance Computing Conference*. Springer, 115–129.

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

[14] Kuan-Chieh Wang, Yan Fu, Ke Li, Ashish Khisti, Richard Zemel, and Alireza Makhzani. 2021. Variational Model Inversion Attacks. *Advances in Neural Information Processing Systems* 34 (2021), 9706–9719.

[15] Alexander Wood, Kayvan Najarian, and Delaram Kahrobaei. 2020. Homomorphic encryption for machine learning in medicine and bioinformatics. *ACM Computing Surveys (CSUR)* 53, 4 (2020), 1–35.

[16] Andrew C Yao. 1982. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 160–164.