Journal paper review on

A Survey on Privacy-Preserving Machine Learning with Fully Homomorphic Encryption

Presented by
Manigandan Ramadasan

Contents

Introduction

WHAT

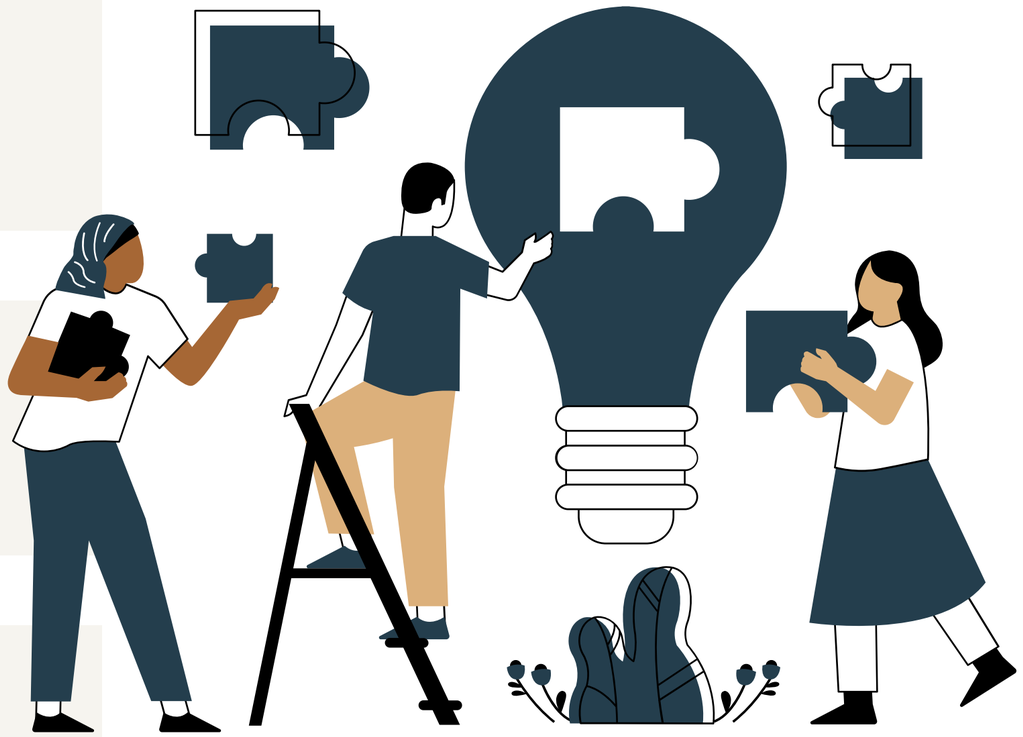
Homomorphic encryption is a form of encryption that allows **computations to be carried out on ciphertexts**

WHY

Privacy and **Security**

WHEN

HE is used when you want to **process sensitive data while preserving the privacy of the information.** Example: Healthcare, Cloud etc.



Homomorphic Encryption

- HE enables applying mathematical operation directly to the ciphertext in usch a way that decryption of ciphertext results in same answer as applying the operations to the original unencrypted data.
- Two Major Operations:
  - Additvely Homomorphic Encryption
  - Multiplicative Homomorphic Encryption

Illustration (+) :

C1 <--- Encryption(M1)

C2 <--- Encryption(M2)

C3 <--- C1 + C2

M3 <--- Decryption(C3)

M3 == M1 + M2

Illustration (*) :

C1 <--- Encryption(M1)

C2 <--- Encryption(M2)

C3 <--- C1 * C2

M3 <--- Decryption(C3)

M3 == M1 * M2

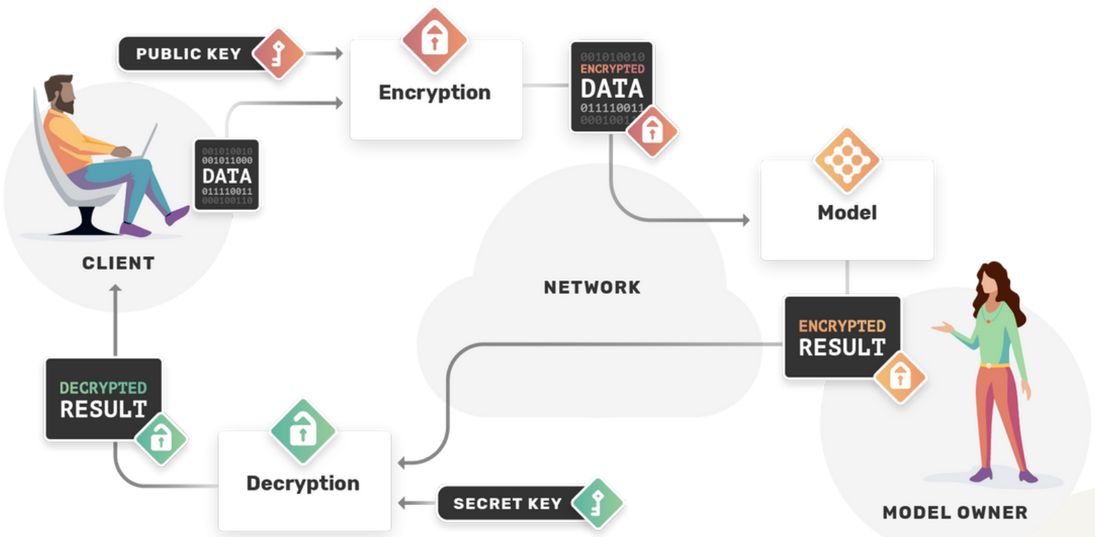


Image Credits: Working of HE. Lopardo, A. (OpenMinded)

Homomorphic Encryption (Contd.)

Partially Homomorphic Encryption (PHE)

This supports any number of a single type of operation either addition or multiplication, not both.
Example: Only Additively Homomorphic Encryption (Pallier encryption), Only Multiplicative Homomorphic Encryption (RSA).

Somewhat Homomorphic Encryption (SHE)

This supports a limited number of operations, both addition and multiplication, but not unlimited because of noise growth.

Leveled Homomorphic Encryption (LHE)

Subset of Somewhat Homomorphic Encryption (SHE). Number of operations performed is limited on the multiplicative depth.

Fully Homomorphic Encryption (FHE)

FHE supports any number of addition and multiplication operation on the ciphertext without leaking any information about the data.

Fully Homomorphic Encryption

FHE supports any number of addition and multiplication operation on the ciphertext without leaking any information about the data and defining the FHE will be as follows:

- Setup
  - Scheme (BFV, CKKS etc)
  - Security parameters (n)
  - Functionality parameters (q)
- Key generation
  - Secret key, public key
- Encryption
  - Vector of numbers ---> Ciphertext
- Evaluation (Function/Model)
- Decryption

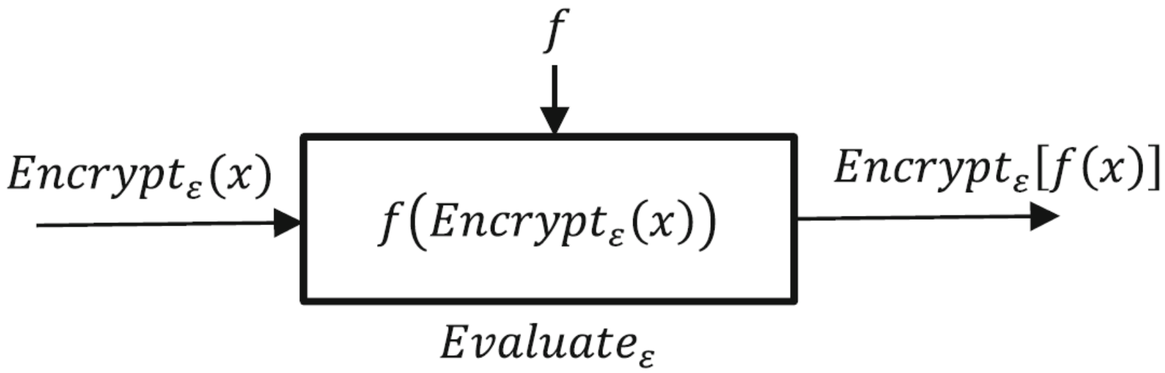


Image Credits: HE Scheme. Pulido-Gaytan, L. B.

Two properties of FHE: Correctness and Compactness

FHE – Bootstrapping (Contd.)

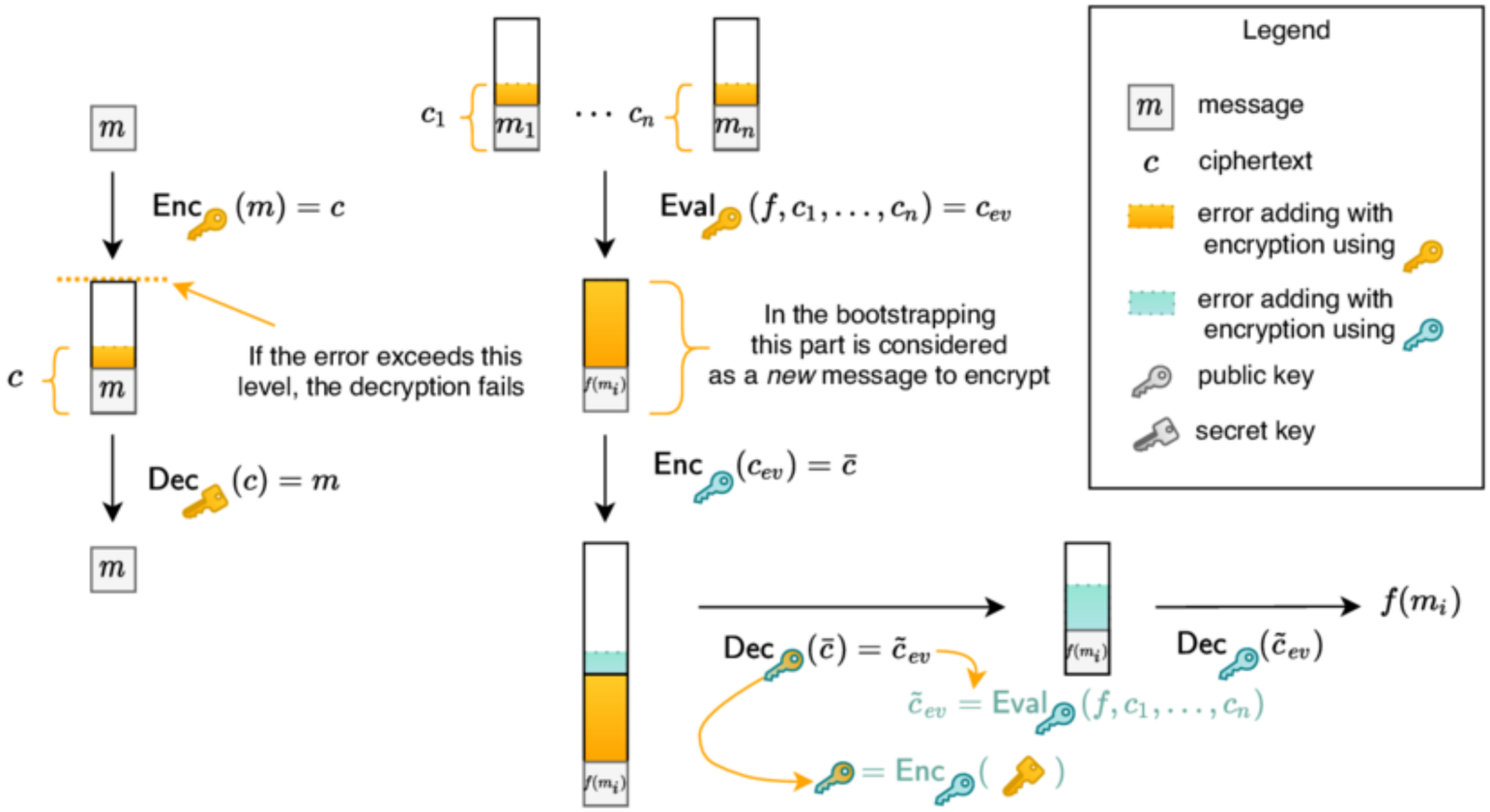


Image Credits: Bootstrapping technique. Marcolla, C.

It is a way to refresh a ciphertext, reducing its noise without knowing the decryption key and without altering the underlying plaintext.

Algorithm 1. Recryption function in bootstrapping

Input: $pk_2, D_\varepsilon, \langle \overline{sk_{1j}} \rangle, c_1$

Output: c_2

$Set\ \overline{c_{1j}} \leftarrow Encrypt_\varepsilon(pk_2, c_{1j})$

$c_{2j} \leftarrow Evaluate_\varepsilon(pk_2, D_\varepsilon, \langle \langle \overline{sk_{1j}} \rangle, \langle \overline{c_{1j}} \rangle \rangle)$

FHE – Key Switching (Contd.)

- In simple notations, bootstrapping can be defined as follows:

$$C'' = Encrypt_{\varepsilon}\left(pk_2, Decrypt_{\varepsilon}\left(sk_1, C'\right)\right)$$

- In the above equation, pk2 (which is otherwise bootstrapping key, bk) can be one of the alternative:
  - The secret key, **sk is encrypted under itself** which leads to bk = Encryption(sk, sk)
  - The secret key, **sk is encrypted under another key namely sk'** which leads to bk = Encryption(sk', sk).
- First alternative requires circular security assumption.
- Second alternative requires key management (depending upon the number of operations) and eventually it will also lead to circular security assumption.

sk1 --> sk2 --> ..... --> skn --> **sk1**

Advances in the field of FHE

First Generation FHE

- First generation FHE was based on ideal lattices and used a technique called bootstrapping to control the noise introduced by computations on ciphertexts.
- Heavy computational requirements.

Second Generation FHE

- The second generation of FHE schemes introduced the Learning With Errors (LWE) problem as the main security assumption. and the scheme is popularly known as BFV.

Random uniform integer matrix $A \in Z_q[n \times m]$

Secret key $s \in Z_q[m \times 1]$

Small randomly sampled error $e \in Z_q[n \times 1]$

The result is $b \in Z_q[n \times 1]$ which we can see as an encryption of 0 that can mask a message m of the same dimensions when added to it

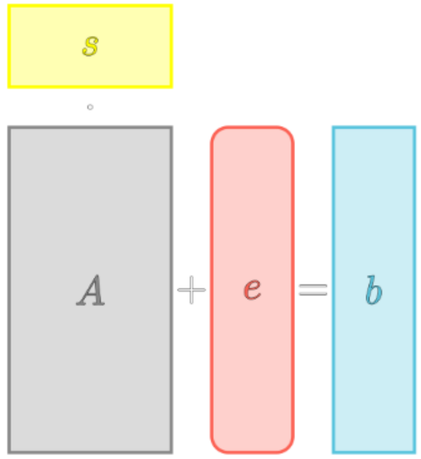


Image Credits: Working of HE. Lopardo, A. (OpenMinded)

Advances in the field of FHE (Contd.)

Second Generation FHE

- The encryption and decryption scheme in second generation FHE is as follows:

$$\mathsf{Enc}\,(\mathsf{pk}, \mu \in \mathcal{M}) \to c \textit{ : given a public key } \mathsf{pk} = (-\mathbf{A}, \mathbf{b}) \textit{ and a message } \mu, \textit{ sample } \mathbf{r} \leftarrow\!\!\$\, \{0,1\}^m$$
$$\textit{and return } c = (\mathbf{c}_1, c_2) = (-\mathbf{A}^\intercal \mathbf{r}, \mathbf{b}^\intercal \mathbf{r} + \mu).$$

$$\mathsf{Dec}\,(\mathsf{sk}, c) \to \mu' \textit{ : given a secret key } \mathsf{sk} \textit{ and a ciphertext } c = (\mathbf{c}_1, c_2) \in \mathbb{Z}_q^{n+1}, \textit{ compute}$$

$$c_2 + \langle \mathsf{sk}, \mathbf{c}_1 \rangle = \mathbf{s}^\intercal \mathbf{A}^\intercal \mathbf{r} + \mathbf{e}^\intercal \mathbf{r} + \mu - \mathbf{s}^\intercal \mathbf{A}^\intercal \mathbf{r} = \mathbf{e}^\intercal \mathbf{r} + \mu.$$

- Homomorphic addition:

$$c_1 = (\mathbf{c}_{11}, c_{12}) \leftarrow \mathsf{Enc}\,(\mathsf{pk}, \mu_1) \qquad c_2 = (\mathbf{c}_{21}, c_{22}) \leftarrow \mathsf{Enc}\,(\mathsf{pk}, \mu_2) \qquad c^+ = (\mathbf{c}_{11} + \mathbf{c}_{21}, c_{12} + c_{22})$$
$$c_{12} + c_{22} + \langle \mathsf{sk}, \mathbf{c}_{11} \rangle + \langle \mathsf{sk}, \mathbf{c}_{21} \rangle = \cdots = \mathbf{e}^\intercal (\mathbf{r}_1 + \mathbf{r}_2) + (\mu_1 + \mu_2)$$

- Homomorphic multiplication:

$$\langle \hat{\mathbf{s}} \otimes \hat{\mathbf{s}}, \hat{\mathbf{c}}_1 \otimes \hat{\mathbf{c}}_2 \rangle = \langle \hat{\mathbf{s}}, \hat{\mathbf{c}}_1 \rangle \cdot \langle \hat{\mathbf{s}}, \hat{\mathbf{c}}_2 \rangle = (\ (\bullet) + \mu_1) \cdot (\ (\bullet) + \mu_2) = \ (\bullet) + (\mu_1 \cdot \mu_2)$$

- If inital noise was B, after L multiplication it will grow into B^{2^L}

Advances in the field of FHE (Contd.)

Third Generation FHE

- Third generation FHE (GSW) doesn`t require the need of relinearization or modulus switching which makes it better than second generation FHE.
- The encryption and decryption is as follows:

$\mathsf{Enc}(\mathsf{pk}, \mu \in \mathcal{M}) \to \mathbf{C}$: given a public key pk and a message μ, sample a random matrix $\mathbf{R} \leftarrow_{\$} \{0,1\}^{m \times m}$, and output

$$\mathbf{C} = \begin{pmatrix} -\mathbf{A} \\ \mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal \end{pmatrix} \mathbf{R} + \mu \mathbf{G} \in \mathbb{Z}_q^{n \times m}.$$

$\mathsf{Dec}(\mathsf{sk}, \mathbf{C}) \to \mu'$: given a secret key sk and a ciphertext $\mathbf{C}$, compute

$$\mathsf{sk}^\intercal \mathbf{C} = \hat{\mathbf{s}}^\intercal \left(\begin{pmatrix} -\mathbf{A} \\ \mathbf{s}^\intercal \mathbf{A} + \mathbf{e}^\intercal \end{pmatrix} \mathbf{R} + \mu \mathbf{G} \right) = \mathbf{e}^\intercal \mathbf{R} + \mu \hat{\mathbf{s}}^\intercal \mathbf{G},$$

and return μ', the closest integer to $\hat{\mathbf{s}}^\intercal \mathbf{G}$.

$$\mathbf{G} = \mathbf{g}^\intercal \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times n\ell} \qquad \mathbf{g} = \left(1, 2, 4, \ldots, 2^{\ell-1}\right)$$

$$\ell = \lceil \log q \rceil$$

$$\mathbf{G} = \begin{bmatrix} 1 & 2 & \cdots & 2^{\ell-1} & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 2 & \cdots & 2^{\ell-1} & \cdots & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 & 2 & \cdots & 2^{\ell-1} \end{bmatrix}$$

FHE and MLaaS

- Machine Learning as a Service (MLaaS) is a set of services that provide machine learning tools as part of cloud computing services.
- With FHE, you can encrypt your data in such a way that it's still possible to perform meaningful computations (like machine learning algorithms) on the data while it remains encrypted.
- Working of MLaaS and FHE:
  - Data encryption
  - Model training and prediction
  - Return of results
  - Decryption
- Generally researchers focus mainly on Hyperplane Decision Making (SVM), Naive Bayes Classifier, Decision Tree and Neural Network Algorithms.
- Very few researches have been done on Logistic Regression as it mainly focuses on gradient descent. Due to this high computation, Logistic Regression won't be much efficient than other algorithms.

FHE and MLaaS (Contd.)

- Comparisions, argmax, dot product plays a significant role in Machine Learning algorithms and the paper titled "Machine Learning Classification over Encrypted Data" by Bost, A. et al. They focused on how to give prediction results using FHE rather than training the model using FHE.

Hyperplane Decision Makinig

Here, to give classification results, dot product on the feature vector and weights is performed and finally argmax operation is performed to know the final class.

Naive Bayes Classifier

Here, to give classification results, dot pproduct on the feature vector and weights is performed and finally argmax operation is performed to know the final class.

$$\hat{Y} = \underset{Y_i \in Y}{\operatorname{argmax}} \left[\Pr(Y = Y_i|X)\right] = \underset{Y_i \in Y}{\operatorname{argmax}} \left[\Pr(Y = Y_i) \prod_{j=1}^{d} \Pr(X = X_j|Y = Y_i)\right]$$

FHE and MLaaS (Contd.)

Naive Bayes Classifier

Client input: Data point $x \in \mathbb{Z}^d$, FHE public key PK.
Server input: Tables P and T, FHE secret key SK.
Client output: The index of the class with the highest classifier score.

1: The server encrypts the tables P and T.
2: The client homomorphically evaluates

$$[p_i] = [P_i] \prod_{j=1}^{d} [T_{i,j}(X_j)]$$

for $i = 1, \ldots, c$.
3: The client uses the server to privately compute $\hat{i} = \text{argmax}_i p_i$.
4: The client outputs $\hat{i}$.

FHE and MLaaS (Contd.)

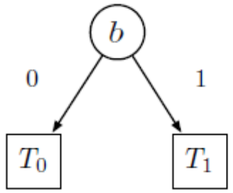
Decision Tree Algorithm

In DT, it has decision nodes and leaf nodes and it is important that the client doesn't learn about the tree structure.
So the tree is generally represented in as a polynomial :

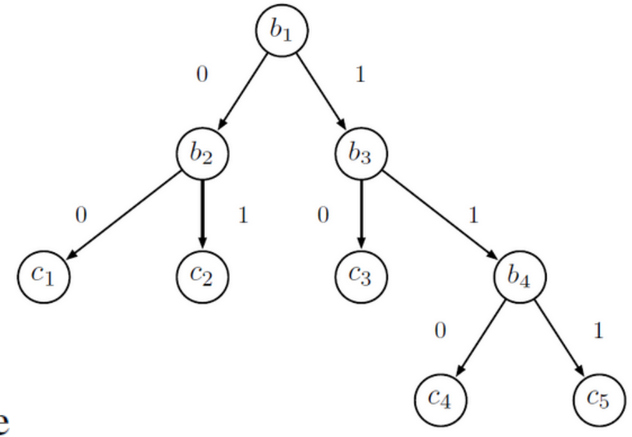
$\mathcal{F}$, a recursive procedure for constructing P given a binary decision tree T:

If T consists only of a leaf node with category index c_i, $\mathcal{F}(T) = c_i$.

If T is empty, return $\mathcal{F}(T) = 0$.



Otherwise, T has an internal node using boolean b and T_0 and T_1 are left and right subtrees. Then $\mathcal{F}(T) = b \cdot \mathcal{F}(T_1) + (1 - b) \cdot \mathcal{F}(T_0)$.



$$P(b_1, b_2, b_3, b_4, c_1, \ldots, c_5) = b_1(b_3 \cdot (b_4 \cdot c_5 + (1 - b_4) \cdot c_4) + (1 - b_3) \cdot c_3) + (1 - b_1)(b_2 \cdot c_2 + (1 - b_2) \cdot c_1)$$

FHE and MLaaS (Contd.)

Decision Tree Algorithm

Client input: Data point X, FHE secret key SK.
Server input: The polynomial form P of a decision tree, FHE public key PK.
Client output: The index of the leaf node containing the classification of X.

1: The client and model owner perform comparisons so that the server learns $[b_i]$, the (encrypted) binary output of each internal node.
2: The server uses $[b_i]$ to homomorphically evaluate the polynomial P.
3: The client receives and decrypts the output, $[P(X)]$, to receive their final classification.

Neural Networks

The first ever NN using FHE was CryptoNets. From that, a lot of developments has been made such as aplying normalization procedure, using different FHE schemes, using stochaistic gradient etc.

Application and Tools

Microsoft SEAL (Simple Encrypted Arithmetic Library): Developed by **Microsoft** Research, this library provides a set of encryption schemes including the Brakerski-Gentry-Vaikuntanathan (**BGV**) scheme and the Cheon-Kim-Kim-Song (**CKKS**) scheme.

HElib: Developed by **IBM**. It provides implementations of the **BGV** scheme, the somewhat homomorphic predecessor to the Gentry-Sahai-Waters (**GSW**) scheme and **CKKS** and it performs computations with limited bootstrapping.

TFHE (Fast Fully Homomorphic Encryption over the Torus): An open-source library implementing **LWE over Torus** scheme, a third-generation FHE scheme that aims to be faster and more efficient than earlier schemes.

Application and Tools

PALISADE (Practical Asymmetric Lattice-based Integrated Suite of Scalable Software And DEsign): This library offers a variety of lattice-based encryption techniques, including several homomorphic encryption schemes.

In python, there are many open source libraries like TenSEAL, PyFHEL, PyCrCNN, Python-FHEz etc.

Applications include medical field, cloud computing, financial service, MLaaS, signal processing, voting systems, spam filtering, image processing, smart cities etc.

Conclusion

The integration of FHE with MLaaS has the potential to revolutionize the field of machine learning, providing strong privacy guarantees without sacrificing the privacy of the data.

The authour concludes by pointing the areas that needs improvement such as:

- Bootstrapping procedure
- Improving the technical characteristics
- Expanding the scope of the homomorphic ciphers
- Designing and implementing machine learning models

Questions and Feedback

Thank You !

References

1. Pulido-Gaytan, L. B., Tchernykh, A., Cortés-Mendoza, J. M., Babenko, M., & Radchenko, G. (2021). A survey on privacy-preserving machine learning with fully homomorphic encryption. In High Performance Computing: 7th Latin American Conference, CARLA 2020, Cuenca, Ecuador, September 2–4, 2020, Revised Selected Papers 7 (pp. 115-129). Springer International Publishing.
2. Minelli, M. (2018). Fully homomorphic encryption for machine learning (Doctoral dissertation, Université Paris sciences et lettres).
3. Lopardo, A. (2020). What is Homomorphic Encryption? OpenMined Blog. https://blog.openmined.org/what-is-homomorphic-encryption/
4. Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F. H., & Aaraj, N. (2022). Survey on Fully Homomorphic Encryption, Theory, and Applications. Proceedings of the IEEE, 110(10), 1572-1609.
5. Bost, R., Popa, R. A., Tu, S., & Goldwasser, S. (2014). Machine learning classification over encrypted data. Cryptology ePrint Archive.
6. Wood, A., Najarian, K., & Kahrobaei, D. (2020). Homomorphic encryption for machine learning in medicine and bioinformatics. ACM Computing Surveys (CSUR), 53(4), 1-35.
7. Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., & Wernsing, J. (2016, June). Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In International conference on machine learning (pp. 201-210). PMLR.