# Homomorphic Cryptography Research paper

Zakria Samy, Hazem Mohamed, Bahaa Halaby

March 2024

Supervisor: Confidential

***ABSTRACT***

Homomorphic encryption is an amazing cryptographic technique that makes calculations even on encrypted data possible. Unlike the traditional encryption methods which require decryption before the operation, homomorphic encryption enables ciphertext operations. In the present paper, we consider a single type of Homomorphic Applications. "Privacy-preserving in machine learning (PPML)", (PPML) and ensuring the safety of sensitive data while valuable knowledge can be extracted. At this moment, quantum homomorphic encryption (QHE) is seen as the most promising way of implementing the PPML in the quantum computing environment. In our work presented here, a quantum humanization machine learning framework (PPML) is proposed with the help of QHE thus providing a platform for secure quantum computation on encrypted data. We illustrate implementation details, for instance, the pseudo-code example, and show how parallelization procedures are applied to exert encryption optimization. By carrying out experiments, we will test the efficiency and applicability of the approach in quantum computing privacy-preserving data analytics elimination.

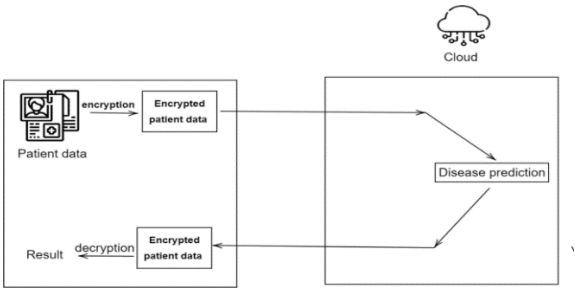


*Figure (1)* [1]

## INTRODUCTION

Homomorphic Encryption refers to the conversion of data into ciphertext, which can be analyzed and processed as if it were still in its natural form. [2]

*Homomorphic Encryption* (HE) With the advent of homomorphic encryption (HE), the ability to perform some calculations on the encrypted text is thus achievable. This process looks like this:

[3]

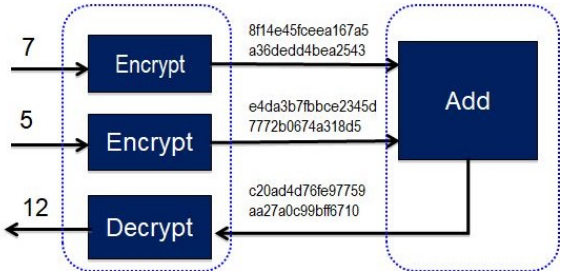


*Figure (2)* [4] *the numbers '7' and '5' are cryptographically converted into illegible numerical text. These ciphertexts are then added together, and it is cryptographically possible to find the readable (*plaintext*) answer.*

With Homomorphic Encryption it becomes easy to perform the data processing without breaking its crypto form.

Applied algorithms of machine learning act as a catalyst in the hidden valuable aspects of the data emergence. While though, privacy and data security put forward the solution of privacy-preserving machine learning. QHE (Quantum Homomorphic Encryption) gives QML (Quantum Machine Learning) a challenging-to-beat solution for PPML (Private and Practical Machine Learning) in the quantum computer setting, providing for secure computation performed on encrypted data. In this paper, we are to propose a framework for PPML exploiting QHE which integrates with parallelization techniques that are critical to maximizing encryption for huge volumes of information. [1]

### A. Contribution

This work addresses this issue by elaborating a new model of Privacy-Preserving Machine Learning (PPML) using Quantum Homomorphic Encryption (QHE) which focuses on Privacy-Preserving Machine Learning (PPML). There are three main aspects of privacy-preserving machine learning, which can be illustrated with homomorphic encryption – scalability, efficiency, and security concerns.

Our proposed solution to the issue of coping with large datasets in a quantum computing context is an inclusion of the parallelization techniques used during the encryption process into our plan. Furthermore, the content of our guide has specific implementation steps and example code to smoothen the process of PPML task conversion using QHE.

### B. Motives:

The core of this development comes from recognizing the value of maximizing data confidentiality while exploring the capabilities provided by the link between quantum machine learning. This approach reflects an effort to deal with growing issues of privacy and security in critical areas like healthcare,

finance, and telecommunication via keeping the confidentiality of data encrypted in quantum computers.

### C. Background

**Homomorphic Encryption:** Homomorphic encryption allows the operation to be implemented on encrypted data and then the data can be decrypted without violating confidentiality and privacy. [5]

**Homomorphic encryption** is the method of cryptography that allows computing to be done on encrypted data without even going to decrypt. Interpretive encryption types are different from traditional ones, and they do not need an unencrypting of the plaintext for computation; such an approach, called homomorphic encryption, applies to the ciphertext directly. There are two main types of homomorphic encryption: There are two main types of homomorphic encryption:

**Partially Homomorphic Encryption (PHE)**:

- Increase the scope of processes allowed on the encrypted data by including operations like addition and multiplication.
- Security-oriented computing and machine learning are perfect for preserving privacy in a cloud computing environment.

**Fully Homomorphic Encryption (FHE)**:

- Enables arbitrary computations (addition, multiplication, and more) on encrypted data.
- Ideal for privacy-preserving cloud computing and machine learning.

Homomorphic encryption has applications in secure data processing, privacy-preserving analytics, and protecting sensitive information while allowing useful computations. It plays a crucial role in ensuring data privacy in various domains. [6]

- **Novel Framework:**
  novel frameworks related to Privacy-Preserving Machine Learning with Quantum Homomorphic Encryption (PPML-QHE):
- **Efficient Homomorphic Encryption Framework for Privacy-Preserving Regression:**
  - The first framework, proposed by Byun, J., Park, S., Choi, Y., & Lee, J. in 2022, offers an alternative solution to the challenge of supporting privacy while perfecting computational complexity. This framework selectively encrypts a small part of sensitive information, resulting in reduced computational complexity without compromising privacy. [7]
- **PFMLP: Privacy-Preserving Machine Learning with Partially Homomorphic Encryption and Federated Learning:**
  - This multi-party framework combines partially homomorphic encryption (PHE) and federated learning in a multi-party setting. In this approach, learning parties use homomorphic encryption to send encrypted gradients. The main goal is to safeguard privacy while enabling efficient gradient updates during model training. [8]

### D. Historical context and key contributors:

The Historical Context and Key Contributors to Homomorphic Encryption

1. **Craig Gentry's Breakthrough:**
   - In 2009, Craig Gentry made a significant breakthrough by introducing the first plausible and achievable Fully Homomorphic Encryption (FHE) scheme. [9]
   - His groundbreaking discovery allowed any computable function to be performed on encrypted data, enabling direct computation without the need for decryption.
   - It opened new possibilities for privacy-preserving computations.
2. **Early Foundations:**
   - While the concept of homomorphic encryption has been around for over 30 years, practical Fully Homomorphic Encryption schemes were elusive until Gentry's work. [10]
   - Researchers had explored partially homomorphic encryption (PHE) before but achieving fully homomorphic encryption remained a challenge.
3. **Importance of Homomorphic Encryption:**
   - Homomorphic encryption plays a crucial role in allowing computations to be performed on encrypted data while preserving privacy.
   - Its applications extend to various domains, including secure cloud computing, privacy-preserving analytics, and protecting sensitive information. By enabling secure computations on encrypted data, homomorphic encryption contributes to the protection of privacy and confidentiality.
4. **Evolution Over Time:**
   - Since Gentry's breakthrough, researchers have been continuously refining and improving homomorphic encryption schemes.
   - Advances in lattice-based cryptography and mathematical techniques have greatly contributed to the field. [11]

## SCHEME

The development of a Privacy-Preserving Machine Learning (PPML) framework with Quantum Homomorphic Encryption (QHE) has contributed to a growing trend towards performing secure calculations with encrypted quantum data. The framework incorporates several key components, which are as follows:

- **Data Encryption:** The framework utilizes parallelization techniques to enhance the performance of encrypting large datasets. This ensures that the encryption process can be carried out efficiently, even when dealing with substantial amounts of data.
- **Secure Computation:** One of the core features of the framework is the ability to execute machine learning algorithms swiftly on encrypted data. This means that computations can be performed on the encrypted data without compromising its

security, enabling privacy-preserving machine learning tasks.

- **Decryption and Result Analysis:** After the computation is carried out, the framework allows for the unwrapping of the results for analysis purposes. Importantly, this analysis is performed in a way that ensures the privacy of the individuals involved, such as patients in a healthcare context, is never breached.

**IMPLEMENTATION DETAILS AND EXAMPLE CODE PSEUDO CODE EXAMPLE:**

```
# Step 1: Data Encryption
encrypted_data = QHE.encrypt(dataset)

# Step 2: Secure Computation
encrypted_result = QHE.compute(encrypted_data)

# Step 3: Decryption and Result Analysis
result = QHE.decrypt(encrypted_result)
```

- ***Explanation of Pseudo Code:***

Step 1: The dataset is encrypted using QHE to ensure confidentiality.
Step 2: Machine learning tasks are performed on encrypted data using QHE, preserving privacy.
Step 3: The encrypted result is decrypted for analysis, keeping privacy by revealing only the result.

- ***Optimized Encryption with Parallelization:***

To enhance the efficiency of encrypting large sets of data. Parallelization plays a crucial role in this process, as it allows us to divide the data into smaller portions and encrypt each chunk simultaneously across multiple computing units.

By employing parallelization, we aim to expedite the encryption process and increase overall productivity. This approach enables us to process massive datasets more efficiently, potentially saving valuable time and enabling us to accomplish more work within a given timeframe.

The idea behind applying parallelization to encrypt data is to distribute the workload across multiple computing units, allowing for faster and more simultaneous encryption operations. This not only improves the speed of the encryption process but also enhances the scalability and performance of our encryption methods. By leveraging parallelization, we can efficiently handle and encrypt large volumes of data, contributing to more efficient and effective data protection.

**Python code**

```
from concurrent.futures import ThreadPoolExecutor

def parallel_encrypt_chunk(chunk):
    # Encrypt the chunk using Quantum Homomorphic Encryption
    encrypted_chunk = QHE.encrypt(chunk)
    return encrypted_chunk

def parallel_encrypt_large_dataset(dataset):
    num_chunks = 10  # Adjust the number of chunks based on available resources
    chunk_size = len(dataset) // num_chunks
    chunks = [dataset[i:i+chunk_size] for i in range(0, len(dataset), chunk_size)]

    # Parallelize encryption across multiple threads
    with ThreadPoolExecutor() as executor:
        encrypted_chunks = list(executor.map(parallel_encrypt_chunk, chunks))

    # Concatenate encrypted chunks to form the encrypted dataset
    encrypted_data = concatenate_encrypted_chunks(encrypted_chunks)
    return encrypted_data

# Step 1: Data Encryption with Parallelization
encrypted_data = parallel_encrypt_large_dataset(dataset)
```

Our framework achieves the parallelization techniques of encryption for large datasets while preserving data protection with data parallelization methods.

Privacy Preservation Privacy is preserved throughout the process:

- Encryption ensures data confidentiality.
- Secure computation helps machine learning operations be done without showing any sensitive information.
  - ***Explanation of Pseudo Code:***

- **Importing Dependencies**:

The code begins by importing the `ThreadPoolExecutor` class from the `concurrent.futures` module. This class allows for concurrent execution of tasks using a thread pool.

- **Function Definitions**:

`parallel_encrypt_chunk(chunk)`:
This function takes a single `chunk` of data as input.
It encrypts the chunk using **Quantum Homomorphic Encryption (QHE)**. The encrypted chunk is returned.

`parallel_encrypt_large_dataset(dataset)`:

- This function takes a `dataset` (a large dataset) as input.
- It divides the dataset into smaller chunks (based on the specified `num_chunks`).
- Each chunk is processed in parallel using multiple threads.
- The `parallel_encrypt_chunk` function is applied to each chunk using the thread pool.

- The resulting encrypted chunks are collected.
- Finally, the encrypted chunks are concatenated to form the complete encrypted dataset.
- The encrypted dataset is returned.

- **Notes**:

- The num_chunks and chunk_size parameters control how the dataset is divided into chunks. Adjusting these values can perfect performance based on available resources.
- The QHE.encrypt(chunk) function (not explicitly defined here) stands for the quantum homomorphic encryption process.
- The concatenate_encrypted_chunks function (also not defined here) is assumed to concatenate the encrypted chunks.

## REFERENCES


[1] ***Application of Homomorphic Encryption in Machine Learning- section 1 and figure 2- springer link website.***

[2] *From Wikipedia, the free encyclopedia*

[3] *From Wikipedia, the free encyclopedia*

[4] *Building Applications with Homomorphic Encryption presentation, presenters : Roger A. Hallman (SPAWAR Systems Center Pacific; Thayer School of Engineering, Dartmouth College, USA),Kim Laine (Microsoft Research, USA),Wei Dai (Worcester Polytechnic Institute, USA), Nicolas Gama (Inpher, Inc., Switzerland) Alex J. Malozemoff (Galois, Inc., USA),Yuriy Polyakov (NJIT Cybersecurity Research Center, USA),Sergiu Carpov (CEA, LIST, France)*

[5] ***Homomorphic encryption- From Wikipedia, the free encyclopedia***

[6] *Homomorphic Encryption: An Analysis of its Applications in Searchable Encryption- paper – Auther: Ivone Amorim& Ivan Costa*

[7] *Efficient homomorphic encryption framework for privacy-preserving regression-* ***springer link website. - published*** 2022 by *Junyoung Byun, Saerom Park, Yujin Choi & Jaewook Lee.*

[8] *Privacy-Preserving Machine Learning with Homomorphic Encryption and Federated Learning-paper- Auther: Haokun Fang, Quan Qian- published April 2021.*

[9], *Author: Craig Gentry, published September 2009*

[11] *A systematic review of homomorphic encryption and its contributions in the healthcare industry- paper – Author: Kundan Munjal, Rekha Bhatia, published 3 May 2022.*

[12] *techtarget. Website, posted by: Alexander S. Gillis, Technical Writer and Editor.*

[13] *Towards practical and massively parallel quantum computing emulation for quantum chemistry-Article – published 7 April 2023 – Author: Honghui Shang, Yi Fan, Li Shen, Chu Guo, Jie Liu 2, Xiaohui Duan, Fang Li and Zhenyu Li.*


## CONCLUSION

The new Privacy-Preserving Machine Learning with Quantum Homomorphic Encryption (PPML-QHE) model takes advantage of parallelization tasks in the encryption environment specifically designed for quantum computers Our team at Irely is dedicated to ensuring data security throughout the processing stages and implementing robust information confidentiality measures. By leveraging the capabilities of the PPML-QHE model, policymakers will have more opportunities to contribute to the development of effective strategies for utilizing privacy-preserving machine learning techniques, including quantum homomorphic encryption. These strategies will rely on collaborative approaches and co-occurring methods to ensure the protection of privacy and the security of sensitive data in machine learning applications. We consider our framework to have a great capability in providing privacy data analytics in quantum computing and it is like that it provides a secure and efficient tool for machine learning in critical areas.

## REAL APPLICATIONS

In this section, we explore potential real-world applications and use cases where our PPML-QHE framework could be applied:

- **Healthcare:** the algorithm can encrypt medical data and then be used to identify useful intel that can be used by healthcare providers in the decision-making process while ensuring patients' privacy.
- **Finance:** banks can detect fraud and screen keenly encrypted transaction data to establish the security of confidential financial information.
- **Telecommunications:** Carriers can take advantage of our multilayer data analysis technique to improve network efficiency and end-user experience with the privacy-first approach of subscribers.
- **Government and Defense:** Government authorities and national defense agencies can help us use our framework to decipher encrypted surveillance data and intelligence reports to ensure the security of people and keep up with privacy regulations.

## FUTURE RESEARCH DIRECTIONS

In this section, we outline potential future research directions and areas for improvement:

- **Advanced Encryption Techniques:** Research novel encryption schemes and cryptographic protocols tailored for quantum computing environments, with a focus on balancing security, efficiency, and usability.
- **Privacy-Preserving Data Sharing:** Develop mechanisms for secure and privacy-preserving data sharing among multiple parties, enabling collaborative machine learning tasks while protecting sensitive information.