A New Scalable Distributed Homomorphic Encryption Scheme for High Computational Complexity Models

Amar Almaini
Edinburgh Napier University
Deggendorf Institute of Technology
Deggendorf, Germany
amar.almaini@th-deg.de

Jakob Folz
Deggendorf Institute of Technology
Deggendorf, Germany
jakob.folz@th-deg.de

Dominik Woelfl
Deggendorf Institute of Technology
Deggendorf, Germany
dominik.woelfl@stud.th-deg.de

Ahmed Al-Dubai
Edinburgh Napier University
Edinburgh, UK
a.al-dubai@napier.ac.uk

Martin Schramm
Deggendorf Institute of Technology
Deggendorf, Germany
martin.schramm@th-deg.de

Michael Heigl
Deggendorf Institute of Technology
Deggendorf, Germany
michael.heigl@th-deg.de

***Abstract*—Due to the increasing privacy demand in data processing, Fully Homomorphic Encryption (FHE) has recently received growing attention for its ability to perform calculations over encrypted data. Since the data can be processed in encrypted form and the output remains encrypted, only an authorized user or a user who holds the key can decrypt the data and understand its meaning. Hence, it is possible to securely outsource data processing to untrustworthy but powerful public computing resources on the edge. However, due to the high computational complexity, FHE-based data processing experiences scalability related concerns. It is currently unclear whether FHE can be used to solve large-scale problems. In this paper, we propose a novel general distributed FHE-based data processing approach as a concrete step towards solving the scalability challenge. The main idea behind our approach is to use slightly more communication overhead for a shorter computing circuit in FHE, hence, reducing the overall complexity. We verify our new model's efficiency and effectiveness by comparing the distributed approach with the central approach over various FHE schemes (CKKS, BGV, and BFV). This is performed using one of the more popular libraries of FHE "Microsoft SEAL", by performing specific mathematical operations and observing the time consumed. The empirical results demonstrate that the proposed approach results in a significant reduction in time, up to 54% compared to the traditional central approach.**

***Index Terms*—Fully Homomorphic Encryption, Distributed Data Processing, Microsoft SEAL, Security, Edge Computing**

I. Introduction

Over the past decade, various high-tech industries have seen a proliferation of groundbreaking innovations, which has been accompanied by an avalanche of large amounts of data in fields such as healthcare, finance, and manufacturing. In industry, data is no longer hosted on a single server or cluster, and additional computational resources are required to process these data. Cloud and edge computing techniques have emerged as effective big data solutions. However, storing and processing sensitive user data on publicly-accessible cloud servers raises concerns about data security and privacy. Traditional encryption solutions are incapable of resolving the conflict between processing large amounts of data in computationally powerful but not completely trustworthy public cloud computing. Homomorphic encryption with its capability to do computations over the encrypted data can offer the answer for this conflict [1][2]. Before sending confidential data to the cloud for processing, this data can be encrypted using FHE on the client side. In this manner, confidential data can be processed using the cloud's vast computing capabilities without disclosing any information about the data because the remote server only sees ciphertexts and never has access to the secret key, users can be confident that it learns nothing about their data. However, due to the high computational complexity and long processing times, FHE-based data processing is still not widely adopted, especially in scenarios with limited computational resources and strict time constraints. Distributed Homomorphic Encryption, which focuses on distributing computations across numerous instances, could be a feasible option. In this paper, we examine the performance of the CKKS, BFV, and BGV schemes utilizing one of the best-known libraries, "Microsoft SEAL" [3] by applying the arithmetic operations addition and multiplication first in a central approach, in which all operations are performed on just one instance, and then in a distributed approach, in which the operations are performed on multiple computing instances. We observed the consumed time for all schemes in both approaches, and the results demonstrate that the distributed approach can yield extremely promising results in terms of significant improvement in the time required to obtain the same results. Finally, note that the distributed approach, in addition to reducing processing time, eliminates meta-level

This research work has been supported by the research project 16KISA128K of the German Federal Ministry of Education and Research.

inferences about the nature of the computations, which could be an additional privacy benefit. The major contributions of this paper are presented as:

- We propose a new distributed FHE-based data processing approach to address the scalability challenges encountered by FHE-based data processing due to its high computational complexity. Our approach utilizes distributed computing techniques to divide the computation workload among multiple computing instances, thus reducing the overall computational burden on a single instance.
- We evaluate the performance of our proposed approach by comparing three commonly used FHE schemes (CKKS, BFV, and BGV) using Microsoft SEAL library and arithmetic operations on vectors up to2^{16} in centralized and distributed approaches. Our results show that the distributed approach significantly reduces processing time due to increased computational power and parallel processing.
- Additionally, our distributed approach eliminates the potential for meta-level inferences regarding the computations, leading to increased privacy. This is achieved through the distribution of computations across multiple nodes, rendering it challenging for any single entity to comprehend the nature of the computation and access sensitive information.

Overall, our proposed distributed FHE-based data processing approach offers a promising solution to the scalability challenges of FHE-based data processing. We believe that this approach can pave the way for the wider adoption of FHE-based data processing in various applications, such as cloud computing, big data analytics, and machine learning. The remaining part of this paper is organized as follows: In Section 2, we introduce the related work; In Section 3, we present the flow and functionality of the centralized and distributed approach; Section 4 specifies the libraries and FHE schemes employed, as well as the composition of the evaluation times; In Section 5, the various schemes are contrasted initially, and then the distinctions between centralized and distributed approaches are highlighted; In Section 6, we conclude this paper and in Section 7 we discuss the future work.

II. Related Work

The last decade has witnessed a proliferation of research studies focused on various applications of FHE technology. This innovative encryption technique provides an attractive solution for processing sensitive user data in the public domain, as it enables computations on encrypted data without compromising its confidentiality or breaching the privacy of the users. However, FHE's high computational complexity and the significant time required for data encryption limit its use in time-sensitive applications and prompted researchers to explore alternative strategies to enhance its performance. One promising solution to improve the efficiency of FHE is to use hardware accelerators such as graphics processing units (GPUs) and field programmable gate arrays (FPGAs) in the implementation of homomorphic encryption schemes. These specialized hardware platforms have the ability to perform complex calculations in parallel, significantly reducing the time required to encrypt data and perform computations on encrypted data. Researchers have been exploring the use of these hardware accelerators in combination with FHE algorithms to achieve improved performance. The authors of [4] presented one of the earliest works on GPU implementation of an FHE scheme. The study utilized the Fast Fourier Transform (FFT) algorithm to develop a small parameter size version of the lattice-based FHE system by Gentry and Halevi [5] on an NVIDIA C2050 GPU. The authors reported substantial speedups, with encryption, decryption, and recryption operations obtaining factors of 7.68, 7.4, and 6.59, respectively. The GPU's parallelization capabilities were leveraged to enhance the FHE performance, with the FFT technique employed to address the system's bottleneck, which is the computation of large modular multiplications. Despite these performance improvements, the authors noted that the implemented FHE scheme is still impractical due to the high latency encountered during encryption and recryption operations. The authors of [6] introduced two techniques for performing statistical analysis with FHE on hardware with limited computational power, such as IoT devices. Given the high computational requirements of FHE, it can pose a significant challenge for IoT devices with limited processing power. The first technique, called SHE+FHE, involves encrypting plaintext using Somewhat Homomorphic Encryption (SHE) on the IoT device side, which is less computationally intensive than FHE. The SHE ciphertext is then transformed into FHE ciphertext on the cloud service side, where the computational requirements are less of a concern. The second technique, called SHE+TRIVIUM+FHE, uses the TRIVIUM secret key cryptosystem to encrypt the plaintext and the SHE public key cryptography to encrypt the TRIVIUM key. According to the results of their experiment, the authors confirmed that this technique can significantly reduce the load on IoT devices. The work presented in [7] introduces a new distributed homomorphic image encryption approach. In the encryption phase, the original RGB image is divided into its R, G, and B-channel components. For each channel image, the pixel intensity values are then decomposed into multiple sub-values, resulting in the creation of several component channel images. To secure these component channel images, each is encrypted using the same encryption key and, if needed, compressed before being transmitted or stored. On the decryption side, the encrypted component channel images are decompressed if necessary and decrypted using the same key used during the encryption phase. This process creates the R, G, and B-channel recovered images, which are then combined to form the recovered RGB image. According to the authors, the proposed homomorphic image encryption scheme is robust and can effectively withstand various security threats while also providing enhanced protection for encrypted images. In [8], a comprehensive comparison of the performance of two FHE schemes, the BFV and CKKS, was conducted using one of the widely used FHE libraries, Microsoft SEAL. This study aimed to evaluate the performance of these schemes in terms

of the time required to complete specific arithmetic operations. The authors performed a thorough analysis of the time consumed by each scheme while performing these operations and compared the results to determine which scheme was more efficient. This study provide valuable insights into the performance of FHE schemes and can inform the selection and implementation of FHE in real-world applications. The study's methodology and results can serve as a useful reference for future research in this field. The authors in [9] address the issue of the high computational overhead of FHE by presenting a novel solution, F1, an FHE programmable accelerator. This accelerator is designed to execute full FHE programs. The authors argue that the unique design methodology of F1 is what sets it apart from other FHE acceleration solutions. The accelerator utilizes high-throughput functional units that speed up basic computations, which are common to higher-level operations. In addition, the co-design of the compiler and hardware reduces data movement, which has been identified as a key bottleneck in traditional FHE acceleration solutions. This combination of efficient functional units and optimized data movement results in faster and more efficient execution of FHE programs. The authors in [10] presented two novel hardware architectures aimed at significantly accelerating the BFV homomorphic encryption scheme's encryption and decryption operations. By utilizing high-performance polynomial multipliers, the authors were able to achieve remarkable speed-ups in the BFV encryption and decryption operations. The authors adopted a hardware/software co-design approach, where the encryption and decryption operations were offloaded to an FPGA and the remaining operations were executed in software running on a desktop computer. As per the results reported by the authors, their hardware/software co-design approach led to substantial improvements in the encryption and decryption time compared to their pure software solutions. Specifically, the encryption time was reduced by a factor of approximately 12 and the decryption time was reduced by a factor of approximately 7. These results demonstrate the potential of hardware-accelerated homomorphic encryption schemes to greatly improve the performance of encryption and decryption operations. Extensive efforts have been made to enhance the performance of FHE schemes. Researchers in the field have proposed several hardware architectures, such as those presented in [11] and [12], to improve the efficiency and speed of FHE operations. In addition, the utilization of Graphics Processing Unit (GPU) accelerators, as shown in [13], [14], and [15], has proven to be an effective method for improving the performance of FHE. These works demonstrate the growing interest and investment in the development of more efficient and secure FHE solutions. The hardware architectures and GPU accelerators aim to address the major limitations of FHE schemes, such as high computational complexity and long processing times. These improvements not only lead to a faster and more efficient encryption and decryption process but also enable the widespread adoption of FHE in various fields, such as finance, healthcare, and manufacturing, where sensitive data needs to be processed and stored securely. As the amount of data being generated continues to grow, it is crucial to develop new and improved methods for secure and efficient processing, making these works crucial contributions to the field.

III. The proposed Approach

In this section, we present the two approaches employed in this study. In the centralized approach, all homomorphic encrypted data are processed by a single resource, such as a central server. This approach simplifies the computation process, as all data is in one location. However, it also creates a single point of failure, meaning that if the central resource becomes unavailable, the entire system fails. In addition, the central resource must have sufficient computing power to handle all of the computations, which can be a challenge for large-scale data processing. In the distributed approach, on the other hand, computations are performed by multiple resources in parallel, with each resource processing a portion of the data. The encrypted data is divided into smaller chunks, each of which is sent to a different resource for processing. After the computations are complete, the results are then merged back together to form the final output. This approach has several advantages over the centralized approach. Firstly, it allows for a more scalable solution, as the processing load can be distributed across multiple resources. This reduces the risk of a single resource becoming a bottleneck and improves overall processing times. Secondly, it provides greater fault tolerance, as the system can continue to function even if one or more resources fail. In conclusion, both the centralized and distributed approaches have their strengths and weaknesses, and the choice between the two depends on the specific requirements of each application.

A. Centralized

The centralized approach in Fig. 1 is designed to provide secure and efficient computation on sensitive data. In this approach, two messages, $M1$ and $M2$, are initially generated, where each message $M = \{i_1, i_2, ..., i_n\}$ is a list of integers with values ranging from 1 to 1000. The length of the message n is determined by the test session and can range from $2^7, ..., 2^{16}$. To ensure the confidentiality of the data, the client encrypts both messages with a homomorphic procedure, producing encrypted messages $E1$ and $E2$. The encrypted messages and the necessary relinearization key are then transmitted to the server for computation. On the server side, the addition of $E1 + E2$ is performed to obtain a new message, $E3$. The computation is then completed by multiplying each element in $E3$ by 0.5, resulting in the final message, $E4$. The client alone holds the key to decrypt the message, which is sent back to the client after the calculations are completed.

B. Distributed

The distributed approach, as depicted in Fig. 2, is the main focus of this paper. In this approach, two messages $M1$ and $M2$ are generated in the first step, which are similar in

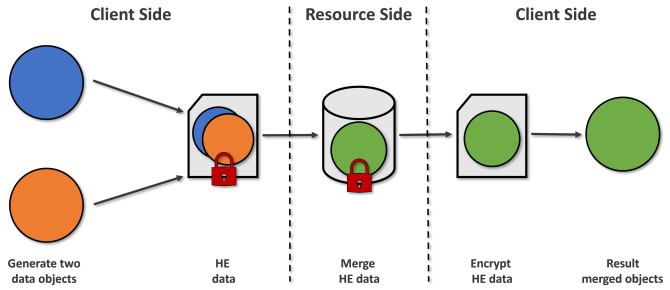


Fig. 1: Centralized model

content and structure to the centralized approach. However, the difference lies in the number of workers that are available for processing the messages. The number of workers can be specified as a set $\{Worker_1, Worker_2, ..., Worker_k\}$. Based on the selected number of workers, the messages $M1$ and $M2$ are divided into partial messages $M1_1, M1_2, ..., M1_k$ and $M2_1, M2_2, ..., M2_k$, respectively, to ensure that each worker receives a portion of both messages. These partial messages are then encrypted using a homomorphic procedure, resulting in $E1_1, E1_2, ..., E1_k$ and $E2_1, E2_2, ..., E2_k$, which are distributed to the corresponding workers. The workers perform the same operations as in the centralized approach, i.e., addition and multiplication, but on partial messages, and return partial results $E3_1, E3_2, ..., E3_k$. The partial results are sent back to the client, where they are decrypted and recomposed into an overall result $M3 = M3_1 \| M3_2 \| ... \| M3_k$. This distributed approach offers several advantages over the centralized approach. One advantage is that it allows for parallel processing of messages, which leads to a faster processing time. This is especially beneficial in large-scale computations, where the processing time can be significantly reduced. Another advantage is that it provides increased security, as the messages are encrypted and distributed among multiple workers, making it more difficult for unauthorized access. Additionally, this approach also provides increased scalability, as the number of workers can be easily increased or decreased depending on the processing needs.

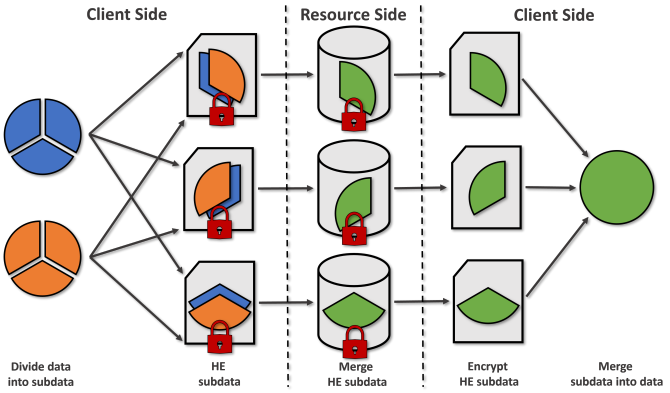


Fig. 2: Distributed model

IV. Evaluation

In this section, we present the results of our scientific study. A detailed description of the libraries utilized, along with the evaluated schemes, and the time stamps applied are provided to ensure a comprehensive understanding of the outcomes and to facilitate the replication of the experimental setup.

A. Utilized Libraries

This study utilized two libraries, openMPI and SEAL, for performing distributed homomorphic computations.

OpenMPI: OpenMPI is a communication framework for multiple processes, including those across different hosts or virtual machines [16]. Computations are divided into multiple processes, prioritized by rank, and processed by different machines. In this study, openMPI was used to implement the distributed approach for homomorphic encryption.

SEAL: SEAL [17] is an open-source library developed by the Cryptography Research Group at Microsoft Research, which supports homomorphic computations through various schemes. A wrapper was used to enable Python support in this study, as SEAL was originally designed for C++ platforms. To test a larger range of values, the predetermined safety constraints for SEAL, which enforce security standards proposed by [18], were deactivated in the test runs, as they only permit certain lengths.

B. Evaluated Schemes

Three schemes were implemented and evaluated in this study, all of which are based on the same operating principle. The process begins with encoding the message, followed by encryption. Then, arithmetic operations are performed on the encrypted data. The data is then decrypted, decoded, and the final result is obtained. A timeline of the homomorphic operations is depicted in Fig. 3.

BGV/BFV: The Brakerski-Gentry-Vaikuntanathan (BGV) schema from 2011 and the Brakerski/Fan-Vercauteren (BFV) schema from 2012 belong to the second generation of encryption schemes, capable of computing messages consisting of integers [19][20]. Despite their structural similarities, the two schemes differ in their parameters. The BFV scheme is unique in that its ciphertext modulus remains constant throughout the evaluation, making it scaling independent. Meanwhile, the BGV scheme uses multiple smaller modulo to better control the noise generated during computations through modulo switching. Despite being part of the older generation of schemes, both BGV and BFV offer higher computational speed compared to newer schemes. However, they are limited to processing integer values only.

CKKS: The Cheon, Kim, Kim, and Song (CKKS) scheme [21][22][23], is more versatile compared to the previously presented BGF and BFV schemes. Unlike the latter, CKKS supports multiplicative inverses and thereby enables the implementation of divisions. Furthermore, it enables the computation of complex numbers by encoding messages into real and imaginary values, which are then represented

as polynomials. The calculations are performed on these polynomials, resulting in an approximation of the outcome after decoding. However, this approach is more time-consuming than the BGF and BFV schemes.

C. Composition Timestamp

This section presents the time records for the two approaches to provide a deeper understanding of their behavior. For this, we grouped the times into different processing steps (see Fig. 3). The initial steps involve generating the keys, encoding the data, and encrypting it. Based on the scheme, the key generation process creates keys of equal length as the polynomial to be encrypted. Keys are also generated for other scheme-specific functions, such as relinearization, which is required after multiplication. Since the values to be evaluated are supplied as vectors, they must first be converted to polynomials; this process is known as encoding. The two mathematical operations of addition and multiplication are represented in the following steps. Finally, in the remaining steps, the data is decrypted and decoded using the same method used in encoding.

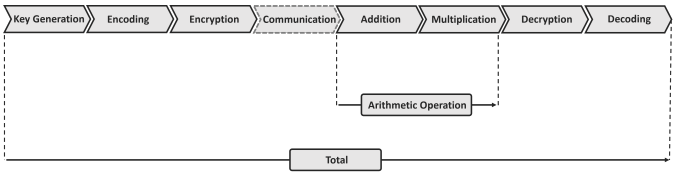


Fig. 3: Composition of different used times

V. Experimental Work and Results

In this study, the mathematical operations of adding and multiplying two vectors of varying sizes were performed homomorphically. A range of vector sizes 2^n were selected, along with appropriate encryption parameters for each scheme, for the experiment. The processing time for all schemes was compared in both centralized and distributed approaches, using the average of 50 test runs for each distinct message length (vector size). The results are presented in tables with a similar format and consistent acronyms. The time consumed for key generation (Key Gen.), encryption (Encr.), which includes encoding time, communication (Com.), arithmetic operations (Arith.), decryption (Decr.), and the total time (Total) are displayed. Additionally, for easy comparison, an improvement column (Impr.) was included to show the percentage improvement gained by the distributed approach compared to the centralized approach in terms of total time.

A. Scheme comparison

The objective of this study is not to compare the different schemes. However, for the purpose of demonstrating the variation in the time required for homomorphic computations across different schemes, a comparison was conducted. The results, presented in Table I, depict the comparison between BFV, BGV, CKKS in the centralized approach. The experiment was conducted using 50 test runs with a vector size of 2^{16} (selected to be in the higher range for a better comparison of the time values).

Scheme	Key Gen.	Encr.	Com.	Add.	Mult.	Decr.	Total
BFV	331.41	155.48	1251.63	1.67	186.78	17.23	1944.19
BGV	333.35	157.33	1218.63	1.62	84.95	16.47	1812.34
CKKS	630.86	437.47	2484.71	3.27	99.16	34.54	3690.00

TABLE I: Processing time in (ms) of messages with vector size of 2^{16} using the different schemes on a single resource

The results indicate that BGV was the fastest in terms of total time, attributed to its efficient multiplication time. The other times of BGV are comparable to those of BFV. On the other hand, CKKS required the longest time, with a higher overhead in key generation and encryption leading to the longest time loss. In conclusion, our experimental setup demonstrates that BFV is 6.78% slower and CKKS is 103.60% slower than BGV for equivalent arithmetic operations.

B. Centralized vs. Distributed Approach Comparison

The results of the computation times for the arithmetic operations, key generation, encoding, decoding, encryption, and total time for the BFV scheme are presented in Table II. The data is divided into three categories: centralized approach, distributed approach with four instances, and distributed approach with eight instances. This categorization is done to highlight the impact of increasing the number of computational resources on the scheme's performance. The results indicate that the computation times increase as the vector size grows in all approaches. However, the distributed approach demonstrates lower computation times compared to the centralized approach, leading to a reduction in the total time. The BFV scheme achieved a maximum improvement of 54% in terms of total time, based on the vector size, and the improvement increases to 68% when the communication time between the computational resources is not considered.

Table III presents the computation times for all operations in the BGV scheme, similar to the previous table. The performance of the BGV scheme is comparable to that of the BFV scheme, given the size of the vector and the number of computational resources utilized. With the BGV scheme, an improvement in total computation time was observed, up to 51% for larger vector sizes, and up to 65% without communication time.

Finally, Table IV displays the computation times for all operations in the CKKS scheme. The scheme exhibits similar performance to other schemes. For the CKKS scheme, an improvement in total time by up to 48% was observed for larger vector sizes, and up to 58% without communication time.

The figures below depict the variations among the tested approaches. The left figure displays the processing time in milliseconds (y-axis) of the three approaches, centralized (represented in blue), distributed with four workers (represented in orange), and distributed with eight workers (represented in

Vector Size	Key Gen.	Encr.	Com.	Arith.	Dec.	Total	Impr.(%)
Centralized							
2^7	3.13	0.57	7.73	0.35	0.07	11.86	-
2^8	3.78	0.85	10.23	0.63	0.09	15.58	-
2^9	4.97	1.35	15.01	1.22	0.15	22.69	-
2^{10}	7.68	2.38	24.63	2.46	0.26	37.41	-
2^{11}	13.24	4.57	43.24	5.05	0.50	66.60	-
2^{12}	23.80	8.90	90.01	10.13	0.95	133.79	-
2^{13}	44.23	17.36	194.02	21.03	1.91	278.56	-
2^{14}	87.28	35.42	309.65	44.2	3.85	480.41	-
2^{15}	169.05	73.57	584.52	90.82	8.15	926.11	-
2^{16}	331.41	155.48	1251.63	188.45	17.23	1944.19	-
Distributed with 4 Workers							
2^7	3.00	1.44	9.71	0.13	0.15	14.44	-22
2^8	2.78	1.59	11.54	0.19	0.18	16.28	-4
2^9	3.14	2.00	14.35	0.36	0.22	20.07	12
2^{10}	3.88	3.17	20.17	0.65	0.32	28.20	25
2^{11}	4.99	5.01	29.82	1.29	0.52	41.63	37
2^{12}	7.82	9.11	55.91	2.68	0.94	76.46	43
2^{13}	12.97	17.48	95.71	5.13	1.84	133.13	52
2^{14}	23.68	34.18	193.48	10.17	3.59	265.10	45
2^{15}	45.99	68.36	377.39	21.35	7.30	520.38	44
2^{16}	88.82	136.85	712.99	43.93	15.05	997.65	49
Distributed with 8 Workers							
2^7	2.93	2.59	12.2	0.09	0.26	18.07	-52
2^8	2.96	2.72	14.4	0.13	0.27	20.48	-31
2^9	2.91	3.22	18.58	0.20	0.33	25.25	-11
2^{10}	3.13	3.98	22.79	0.36	0.42	30.68	18
2^{11}	3.93	6.33	33.3	0.65	0.65	44.85	33
2^{12}	5.30	10.64	52.73	1.33	1.07	71.05	47
2^{13}	7.76	18.01	96.99	2.67	1.91	127.34	54
2^{14}	13.31	35.83	169.49	5.28	3.69	227.6	53
2^{15}	23.37	69.65	373.29	10.66	7.56	484.53	48
2^{16}	46.11	139.43	742.58	22.43	15.58	966.12	50

TABLE II: Comparison of processing times in (ms) between centralized and distributed approach with BFV scheme

Vector Size	Key Gen.	Encr.	Com.	Arith.	Dec.	Total	Impr.(%)
Centralized							
2^7	3.15	0.57	7.77	0.17	0.07	11.73	-
2^8	3.79	0.87	10.20	0.28	0.09	15.23	-
2^9	5.10	1.38	15.04	0.54	0.14	22.20	-
2^{10}	7.69	2.41	24.48	1.07	0.24	35.88	-
2^{11}	13.11	4.64	43.13	2.18	0.47	63.52	-
2^{12}	24.03	9.14	88.16	4.46	0.91	126.7	-
2^{13}	44.99	18.18	186.25	9.38	1.82	260.62	-
2^{14}	87.57	36.14	314.24	20.00	3.67	461.62	-
2^{15}	170.67	74.12	592.95	41.33	7.66	886.74	-
2^{16}	333.35	157.33	1218.63	86.56	16.47	1812.34	-
Distributed with 4 Workers							
2^7	2.98	1.45	9.18	0.07	0.15	13.83	-18
2^8	2.77	1.60	11.13	0.10	0.17	15.77	-4
2^9	3.10	1.99	13.73	0.17	0.21	19.20	14
2^{10}	3.81	3.11	18.89	0.32	0.30	26.43	26
2^{11}	5.03	5.08	29.34	0.57	0.50	40.52	36
2^{12}	7.78	9.21	52.73	1.07	0.90	71.70	43
2^{13}	12.82	17.37	92.72	2.22	1.73	126.86	51
2^{14}	23.77	34.50	192.10	4.48	3.42	258.27	44
2^{15}	45.24	68.69	380.40	9.48	6.93	510.73	42
2^{16}	89.34	139.72	742.75	19.95	14.34	1006.11	44
Distributed with 8 Workers							
2^7	2.91	2.62	12.32	0.05	0.25	18.14	-55
2^8	2.96	2.75	14.24	0.07	0.27	20.29	-33
2^9	2.81	3.14	17.79	0.10	0.31	24.15	-9
2^{10}	3.26	4.21	23.49	0.17	0.42	31.55	12
2^{11}	3.90	6.33	33.90	0.32	0.62	45.07	29
2^{12}	5.08	10.14	51.66	0.60	0.98	68.46	46
2^{13}	7.75	18.25	98.22	1.19	1.77	127.19	51
2^{14}	13.23	35.68	170.74	2.37	3.50	225.53	51
2^{15}	23.30	69.48	392.43	4.72	6.83	496.76	44
2^{16}	45.54	136.52	729.15	9.95	13.75	934.92	48

TABLE III: Comparison of processing times in (ms) between centralized and distributed approach with BGV scheme

gray), as a function of the vector size of the messages (x-axis). The right figure illustrates the improvement of the distributed approaches (y-axis) compared to the centralized approach in terms of percentage.

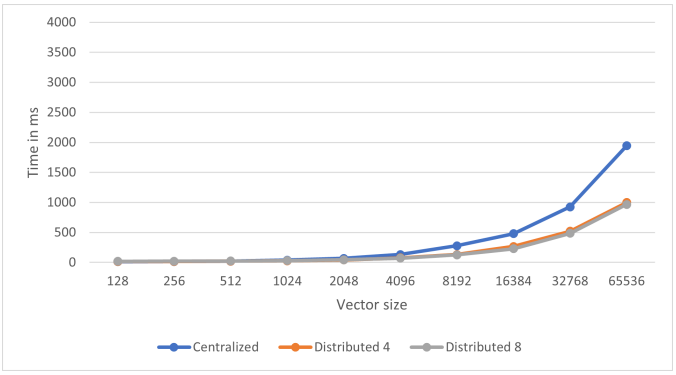
The results of the BVF scheme are presented in Fig. 4. The findings indicate that switching to a distributed approach with four workers is beneficial in terms of processing time when the message length reaches 512, with a reduction in processing time from 22.69ms to 20.07ms (12% improvement). Further, a distributed approach with eight workers is faster than the centralized approach for messages of length 1024 and above, with a processing time of 37.41ms compared to 30.68ms (18% improvement). For messages of length 4096 and above, it is also recommended to use eight workers instead of four, as the increased overhead due to the eight workers is outweighed by the larger message size. In general, the results suggest that up to a message size of 2048, the different approaches are relatively comparable, but as the message size increases, the disparities between them become more pronounced.

The results obtained from the BGV scheme, depicted in Fig. 5, are comparable to those obtained from the BFV scheme. In terms of processing times, the distributed approach with four workers performs better at a message size of 512, with a processing time of 19.20ms compared to 22.20ms (representing a 14% improvement). Conversely, the distributed approach using eight workers in BGV generally yields similar results compared to the distributed approach with eight workers in BFV. As a result, the curve of the results achieved with BGV is very similar to those of BFV.

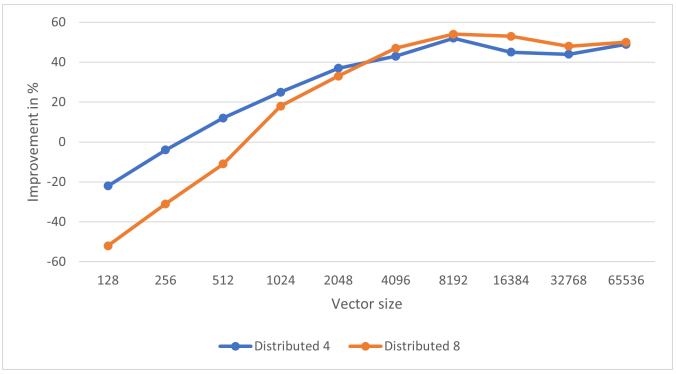
The results in Fig.6 demonstrate that the CKKS scheme leads to longer message processing times compared to BGV and BFV. However, it demonstrates the advantage of a distributed approach at a smaller message size. The distributed approach with four workers improves processing time by 9% (19.27ms vs. 21.19ms) at a message size of 256. The distributed approach with eight workers outperforms the centralized approach starting at 512 message size, yielding a 23% improvement (31.30ms vs. 34.77ms). Eight workers can already attain better results than four workers at a message size of 2048 (3% improvement, 70.24ms vs. 72.23ms). The

Vector Size	Key Gen.	Encr.	Com.	Arith.	Dec.	Total	Impr.(%)
Centralized							
2^7	3.61	0.97	10.10	0.18	0.09	14.95	-
2^8	4.66	1.63	14.43	0.33	0.15	21.19	-
2^9	7.14	2.91	23.82	0.63	0.27	34.77	-
2^{10}	12.04	5.73	41.78	1.27	0.53	61.36	-
2^{11}	22.01	11.53	82.31	2.56	0.98	119.39	-
2^{12}	42.51	23.37	167.44	5.28	1.96	240.56	-
2^{13}	80.37	46.85	306.22	11.26	3.96	448.65	-
2^{14}	155.99	98.53	569.52	23.21	8.08	855.32	-
2^{15}	305.64	207.46	1211.48	48.34	16.76	1789.67	-
2^{16}	630.86	437.47	2484.71	102.43	34.54	3690.00	-
Distributed with 4 Workers							
2^7	2.71	1.75	10.29	0.07	0.17	14.99	0
2^8	3.00	2.22	13.70	0.11	0.22	19.27	9
2^9	3.62	3.58	19.02	0.20	0.33	26.74	23
2^{10}	4.69	6.14	28.40	0.34	0.54	40.12	35
2^{11}	7.31	11.42	51.83	0.66	1.00	72.23	40
2^{12}	12.32	22.34	91.22	1.27	1.87	129.02	46
2^{13}	21.64	43.29	196.51	2.55	3.69	267.68	40
2^{14}	41.79	88.30	370.48	5.35	7.43	513.34	40
2^{15}	81.23	181.18	750.58	11.17	15.18	1039.34	42
2^{16}	157.05	376.06	1419.04	23.26	31.37	2006.78	46
Distributed with 8 Workers							
2^7	2.94	3.00	14.35	0.05	0.29	20.63	-38
2^8	2.72	3.43	18.44	0.07	0.33	24.99	-18
2^9	3.09	4.65	23.03	0.11	0.43	31.30	10
2^{10}	3.76	7.44	33.21	0.22	0.66	45.30	26
2^{11}	4.84	12.51	51.40	0.39	1.10	70.24	41
2^{12}	7.30	22.58	98.15	0.70	1.99	130.71	46
2^{13}	12.43	45.66	170.76	1.38	3.95	234.19	48
2^{14}	21.92	88.40	392.42	2.80	7.70	513.25	40
2^{15}	42.11	184.91	737.99	5.58	15.65	986.23	45
2^{16}	83.99	373.96	1423.49	11.37	33.27	1926.07	48

TABLE IV: Comparison of processing times in (ms) between centralized and distributed approach with CKKS scheme



(a) Comparison of the processing times



(b) Improvement of the distributed approach over the centralized approach

Fig. 4: Comparison of centralized and distributed approaches at different vector sizes with BVF scheme.

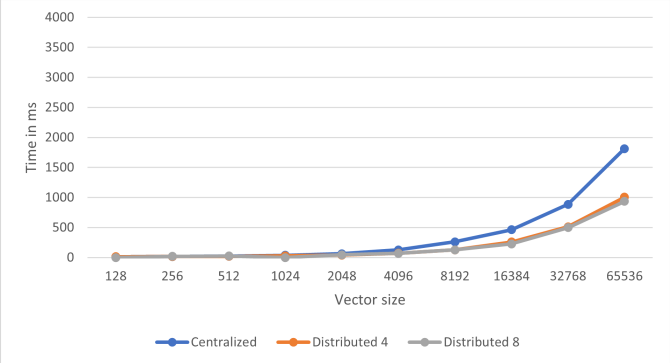
improvement pattern is consistent with previous schemes, but occurs at an earlier stage.

VI. Conclusion

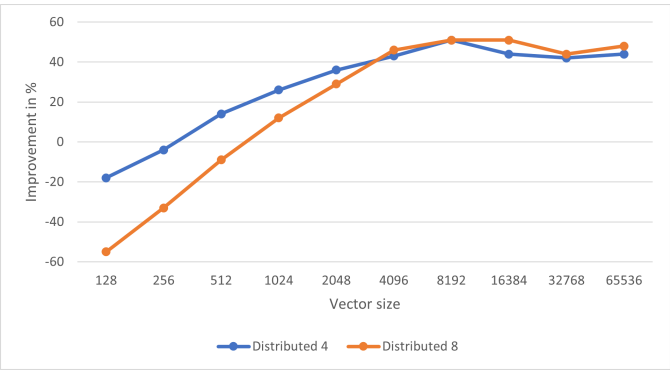
In this study, we propose a novel distributed FHE approach to process sensitive and confidential data while maintaining the security and privacy of the information. The use of FHE allows for computations to be performed on encrypted data without the need for decryption, thus preserving the privacy and security of the data. To evaluate the performance of our proposed approach, we set up a test environment using multiple virtual instances hosted on an external server. We utilized one of the most widely adopted FHE frameworks, which includes the CKKS, BGV, and BFV schemes, to perform various arithmetic operations on datasets of varying sizes up to 2^{16}. We conducted 50 test runs for each combination of the three FHE schemes and various data sizes, comparing the results of the centralized and distributed approaches. The results of our experiments were extremely encouraging, with up to 54% time savings in the distributed approach compared to the centralized approach. This suggests that our proposed distributed FHE approach is a promising solution for handling large volumes of confidential data in untrustworthy public environments without compromising security and privacy. Furthermore, we also analyzed the scalability of our proposed approach by increasing the number of virtual instances and the size of the datasets. The results showed that the distributed FHE approach is highly scalable and can handle large amounts of data without compromising security and privacy. In conclusion, our proposed distributed FHE approach is a promising solution for handling high volumes of confidential data in untrustworthy public environments without violating security and privacy. The approach is highly scalable and can be implemented in various scenarios where data privacy and security are of utmost importance. We believe that this approach can have significant implications for various industries, including healthcare, finance, and government, where sensitive and confidential data need to be processed and analyzed without compromising security and privacy.

VII. Future Work

The present study aims to evaluate the performance of three different FHE schemes, namely the Braverman-Fan-Vercauteren (BFV), the Brakerski-Gentry-Vaikuntanathan

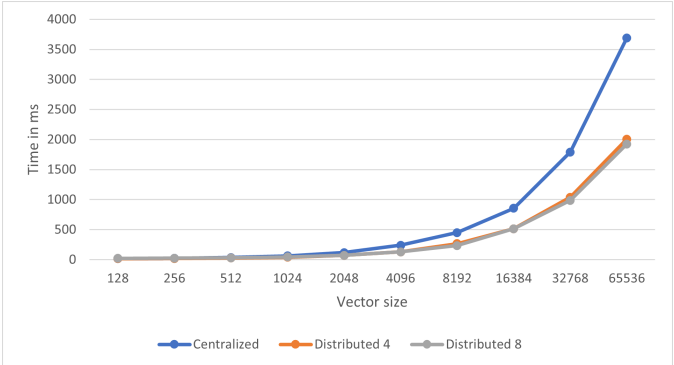


(a) Comparison of the processing times

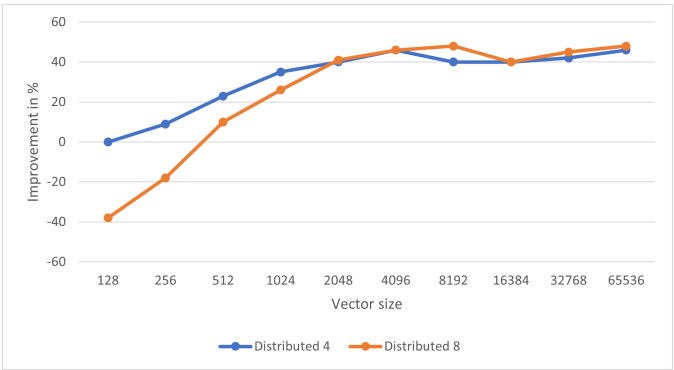


(b) Improvement of the distributed approach over the centralized approach

Fig. 5: Comparison of centralized and distributed approaches at different vector sizes with BGV scheme.



(a) Comparison of the processing times



(b) Improvement of the distributed approach over the centralized approach

Fig. 6: Comparison of centralized and distributed approaches at different vector sizes with CKKS scheme.

(BGV), and the Cheon-Kim-Kim-Song (CKKS) schemes, using the Microsoft SEAL library. The primary objective of this study is to compare the performance of these FHE schemes when performing arithmetic operations on data vectors of varying sizes, up to 2^{16}, using both centralized and distributed approaches. In the centralized approach, the arithmetic operations were performed on a single computational resource. In contrast, the distributed approach involved the use of multiple computational resources, and we also examined the impact of varying the number of resources on the performance of the FHE schemes. To measure the performance of the FHE schemes, we documented and compared the consumed time in both approaches. To further validate our results, we are currently developing a hardware test environment consisting of several Raspberry Pi and NVIDIA Jetson Nano devices [24]. This testbed will represent the various computational resources and will enable us to adapt and rerun our tests. The results of these tests will be made available in the near future. Furthermore, we are focusing on integrating the distributed approach into a framework we developed for machine learning-based resource allocation. In this framework, partial data is distributed to the edge's best resources through a neural network, ensuring that they are used to their fullest potential and minimizing communication times. This approach aims to optimize the performance of the FHE schemes by ensuring that the computational resources are used efficiently and effectively. Overall, the goal of this study is to provide a comprehensive evaluation of the performance of different FHE schemes in distributed environments, with a focus on edge computing. We hope that the results of this study will provide valuable insights for researchers and practitioners working in the field of FHE, and will help to guide the development of more efficient and secure FHE systems for edge computing applications.

References

[1] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. P. Fitzek, and N. Aaraj, "Survey on fully homomorphic encryption, theory, and applications," *Proceedings of the IEEE*, vol. 110, no. 10, pp. 1572–1609, 2022. DOI: 10.1109/JPROC.2022.3205665.

[2] A. B. Levina, V. Y. Kadykov, and D. I. Kaplun, "New direction in cryptography: Homomorphic encryption," in *2021 International Conference Automatics and Informatics (ICAI)*, 2021, pp. 234–237. DOI: 10.1109/ICAI52893.2021.9639809.

[3] C. Aguilar Melchor, M.-O. Kilijian, C. Lefebvre, and T. Ricosset, "A comparison of the homomorphic encryption libraries helib, seal and fv-nfllib," in *International Conference on Security for Information Technology and Communications*, Springer, 2018, pp. 425–442.

[4] W. Wang, Y. Hu, L. Chen, X. Huang, and B. Sunar, "Accelerating fully homomorphic encryption using gpu," Sep. 2012, pp. 1–5, ISBN: 978-1-4673-1577-7. DOI: 10.1109/HPEC.2012.6408660.

[5] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," in *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, ser. EUROCRYPT'11, Tallinn, Estonia: Springer-Verlag, 2011, pp. 129–148, ISBN: 9783642204647.

[6] M. Matsumoto and M. Oguchi, "Speeding up encryption on iot devices using homomorphic encryption," in *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, IEEE, 2021, pp. 270–275.

[7] M. I. Wade, M. Chouikha, T. Gill, W. Patterson, T. M. Washington, and J. Zeng, "Distributed image encryption based on a homomorphic cryptographic approach," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, IEEE, 2019, pp. 0686–0696.

[8] S. M. Fawaz, N. Belal, A. ElRefaey, and M. W. Fakhr, "A comparative study of homomorphic encryption schemes using microsoft seal," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 2128, 2021, p. 012 021.

[9] N. Samardzic, A. Feldmann, A. Krastev, *et al.*, "F1: A fast and programmable accelerator for fully homomorphic encryption," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21, Virtual Event, Greece: Association for Computing Machinery, 2021, pp. 238–252, ISBN: 9781450385572. DOI: 10.1145/3466752.3480070. [Online]. Available: https://doi.org/10.1145/3466752.3480070.

[10] A. C. Mert, E. Öztürk, and E. Savaş, "Design and implementation of encryption/decryption architectures for bfv homomorphic encryption scheme," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 353–362, 2019.

[11] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact ring-lwe cryptoprocessor," in *International workshop on cryptographic hardware and embedded systems*, Springer, 2014, pp. 371–391.

[12] R. De Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede, "Efficient software implementation of ring-lwe encryption," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2015, pp. 339–344.

[13] W. Dai and B. Sunar, "Cuhe: A homomorphic encryption accelerator library," in *International Conference on Cryptography and Information Security in the Balkans*, Springer, 2015, pp. 169–186.

[14] I. Syafalni, G. Jonatan, N. Sutisna, R. Mulyawan, and T. Adiono, "Efficient homomorphic encryption accelerator with integrated prng using low-cost fpga," *IEEE Access*, vol. 10, pp. 7753–7771, 2022.

[15] W. Jung, S. Kim, J. H. Ahn, J. H. Cheon, and Y. Lee, "Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 114–148, 2021.

[16] MPICH. "High-performance portable mpi." (2008), [Online]. Available: https://www.mpich.org/ (visited on 12/08/2022).

[17] *Microsoft SEAL (release 4.0)*, https://github.com/Microsoft/SEAL, Microsoft Research, Redmond, WA., Mar. 2022.

[18] M. Albrecht, M. Chase, H. Chen, *et al.*, "Homomorphic encryption security standard," HomomorphicEncryption.org, Toronto, Canada, Tech. Rep., Nov. 2018.

[19] A. Kim, Y. Polyakov, and V. Zucca, "Revisiting homomorphic encryption schemes for finite fields," in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2021, pp. 608–639.

[20] M. Chase, H. Chen, J. Ding, *et al.*, "Security of homomorphic encryption," *HomomorphicEncryption. org, Redmond WA, Tech. Rep*, 2017.

[21] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, 2021, pp. 648–677.

[22] J. Cho, J. Ha, S. Kim, *et al.*, "Transciphering framework for approximate homomorphic encryption," in *International Conference on the Theory and Application of Cryptology and Information Security*, Springer, 2021, pp. 640–669.

[23] B. Li, D. Micciancio, M. Schultz, and J. Sorrell, "Securing approximate homomorphic encryption using differential privacy," in *Annual International Cryptology Conference*, Springer, 2022, pp. 560–589.

[24] A. Kurniawan, "Introduction to nvidia jetson nano," in *IoT Projects with NVIDIA Jetson Nano: AI-Enabled Internet of Things Projects for Beginners*. Berkeley, CA: Apress, 2021, pp. 1–6, ISBN: 978-1-4842-6452-2. DOI: 10.1007/978-1-4842-6452-2_1. [Online]. Available: https://doi.org/10.1007/978-1-4842-6452-2_1.