Evaluating the use of the Homomorphic Algorithm on Computational Offloading

Francisco Gomes, Filipe Matos, Paulo Rego, Fernando Trinta, José de Souza

Evaluating the use of the Homomorphic Algorithm on Computational Offloading

Francisco A. A. Gomes[1], Filipe Fernandes S. B. de Matos[1], Paulo A. L. Rego[2], Fernando A. M. Trinta[2], and José N. de Souza[2]

[1] Federal University of Ceará (UFC), Crateús, CE, Brazil
almada@crateus.ufc.br, and filipe.fernandes@crateus.ufc.br
[2] Group of Computer Networks, Software Engineering and Systems (GREat)
Federal University of Ceará (UFC), Fortaleza, CE, Brazil
pauloalr@ufc.br, fernando.trinta@dc.ufc.br, and neuman@ufc.br

Abstract

Mobile accounts for almost half of the web traffic worldwide. However, these devices still have computational and energy limitations. Mobile Cloud Computing (MCC) tackles problems like this by migrating tasks and data to remote cloud environments. This technique is known as offloading. However, during this procedure, data is transmitted on the network without protection, which is unfeasible for applications with confidential data that cannot be exposed without security. This work propose a module to ensure the security of data transmitted during computational offloading and that it is easily expandable to any cryptographic algorithms. This work evaluates the performance of computational offloading when adopting Homomorphic cryptography during data migration. The results showed that depending on the operation performed, the time in the offloading process increases up to 900 times.

1 Introduction

The last few years have shown great popularization and growth in the use of mobile devices in society (e.g., smartphones, tablets, and smartwatches). These devices have more and more processing capacity and an increasing number of embedded sensors (e.g., temperature, luminosity, accelerometer, gyroscope), which enable the sensing of environmental data so that they can be interpreted and processed by various applications. However, the mobility provided by mobile devices brings limitations to their use. Furthermore, while using applications like these, users often enter sensitive information that could be exposed to various types of cyber attacks.

To get around the issue of limited resources on mobile devices, research has promoted the integration of mobile devices and cloud resources, creating the research area known today as Mobile Cloud Computing (MCC) [8]. Among the topics addressed in MCC, the most prominent is the use of the *offloading* technique to mitigate processing and storage problems on mobile devices by migrating tasks and data to remote cloud environments [3, 18]. As it is a shared medium, it is essential that any data transmitted during offloading be protected, especially those containing sensitive user information, such as bank details and medical exam results.

One of the methods adopted for data protection in these environments is encryption. Encryption makes the data unreadable at the sender, sends it across the network, and from that nonsense data retrieves the original data at the receiver. Usually, this is performed by encryption algorithms, which are basically programs that implement the idea of a chosen encryption method [4]. In addition to the classic techniques, such as symmetric (AES) and asymmetric (RSA) encryption, homomorphic encryption also guarantees data privacy[1][10]. Furthermore,

this kind of encryption allows operating directly on the encrypted data, while other techniques require decrypting the data before operating on it.

Using homomorphic encryption during the offloading can benefit the client, as it allows tasks or submitted data to be processed by the server without access to the original content. On the other hand, adopting homomorphic encryption requires a computational effort from the transmitter, which can be a problem for mobile devices and their limited resources.

Thus, this paper evaluates the impact of using homomorphic cryptography during the computational offloading performed by mobile applications. For this, we developed an extensible security module to add encryption algorithms to assist the developer in building secure applications. The results showed that depending on the operation performed, the time in the offloading process increases up to 900 times when using homomorphic encryption. The most significant contribution of this work is to present to developers the cost-benefit of using this type of encryption and, in this way, help them decide when to adopt it.

The rest of this article is organized as follows: Section 2 deals with the theoretical foundation and presents the fundamental concepts of the research. Section 3 presents related works. Section 4 presents an overview of the proposal, architecture, established definitions, and an application developed from the proposed solution. Section 5 presents the results of experiments carried out to show the solution's performance in terms of total time, communication, encryption, and execution. Finally, Section 6 concludes the article and exposes possible future works.

2 Background

This Section presents the main concepts, definitions, and characteristics related to the areas on which base this research. Thus, we discuss Mobile Cloud Computing, Offloading, and Homomorphic Encryption concepts.

2.1 Mobile Cloud Computing

MCC's main objective is to mitigate the problems of mobile computing (*e.g.*, energy, processing, and storage limitations). There are many definitions for this paradigm. According to [21], MCC is a computational paradigm that exploits the advantages of Cloud Computing to mitigate the problems of mobile computing and integrates these two areas. According to [15], MCC was created based on the concepts of Cloud and Mobile Computing and aimed to allow applications that require more sophisticated computing resources to run on different mobile devices, providing a good experience for users. Among the various techniques associated with MCC, the most cited in the literature is offloading [17]

2.2 Offloading

According to a research conducted by [8], mobile devices use the offloading technique to reduce energy consumption and improve computational performance by migrating processing and data to equipment with greater computational power and storage. The offloading operation differs from traditional Client-Server architecture's Request/Response mechanism. In the Request/Response mechanism, the servers are always responsible for processing tasks transmitted by the client. In the offloading operation, the client can process tasks if there is no Internet connection or if the mobile device does not benefit from delegating the task computation to the server [22].

Lin et al.[18] indicate two types of offloading operations: computational and data. Computational offloading is an operation that delegates the processing performed on the mobile device to another execution environment (Cloud), aiming to prolong battery life and increase computational capacity. Data offloading aims to extend the mobile device's storage capacity, sending the data to a machine with greater storage capacity.

In addition to the Cloud, mobile devices use other remote devices as offloading targets, such as cloudlets and other mobile devices. According to [24], cloudlets are server instances allocated on the same network as the clients and can handle offloading requests. Thus, when cloudlets are used, offloading remains close to client devices and generates advantages such as higher speed rates and lower latency rates [5].

2.3 Security Techniques

Due to the data migration required when offloading a mobile device to a remote environment, this data can travel on the network without any protection. Thus, there must be security in this migration. Traditional symmetric or asymmetric encryption methods are adopted to provide security in cloud environments. However, processing the data related to offloading must be decrypted on the server that has access to the original data, leaving it vulnerable [11].

2.3.1 Homomorphic Encryption

The homomorphic encryption method allows data to be encrypted and sent to the server. Even so, operations can be performed on that data without needing to decrypt it until it returns to the sender. It is not necessary to have access to the original text to manipulate it [2]. In addition to the operations of encrypting and decrypting data, this technique uses addition and multiplication operations.

Homomorphic encryption systems can be divided into fully homomorphic and partially homomorphic. Drozdowski et al. [7] state that a completely homomorphic system is a system that supports any number of addition and multiplication operations on the data. This first type of system was proposed in [12], but it proved inefficient in terms of processing time. Therefore, most systems with homomorphic properties are partially homomorphic systems [25]. Unlike the one mentioned above, this type of system is computationally practical but comes with the cost of supporting only limited mathematical operations on encrypted data. A partial solution mainly contains two operations: additive or multiplicative homomorphic encryption schemes. In partial, Paillier cryptosystems support addition, and ElGamal cryptosystems support multiplication.

3 Related Works

This Section presents the works related to the present proposal. These works consist of security solutions in the computer offloading the mobile device to a remote environment.

Gomes et al.[14] presented an analytical study on the impact of encryption algorithms on the performance of computational offloading performed by mobile applications. The work developed a security module that guarantees the confidentiality and integrity of data trafficked in offloading and added a framework that supports offloading [13]. Such a module has two components: one on the mobile side and one on the cloud. Such components adopt a hybrid encryption approach, using symmetric and asymmetric encryption algorithms to transmit information over the network. Therefore, the data object being transferred during offloading is encrypted; thus,

the privacy of the mobile device's data and, consequently, of its user is preserved. A notable weakness of this work is security in the offloading process because when the data arrives at the server, it needs to decrypt to operate on it. At this point, the server has access to the plain data, which can often be personal information that should not be visible.

Liu et al.[19] presented an implementation that adopts Steganography techniques in the computational offloading of images from mobile devices to remote servers. The main objective of this work is to provide data security and examine the energy consumption spent in this process. Through these techniques, offloading hides the data to be sent in an image and, on the server side, processes it in this hidden format. In addition, the authors propose an image recovery method based on the block data hiding method. Unfortunately, the authors did not evaluate the performance of the Steganography technique in offloading in terms of the total time of the operation.

Ren et al.[23] claim that data privacy concerns are increasingly affecting the Internet of things (IoT) and artificial intelligence (AI) applications, in which it is very challenging to protect the privacy of the underlying data. In recent, the advancements in the performances of homomorphic encryption have made it possible to help protect sensitive and personal data in IoT applications using homomorphic encryption-based schemes. This paper proposed a practical homomorphic encryption scheme that can enable data users in IoT systems to securely operate data over encrypted data, which can effectively protect the privacy of key data in the system. Furthermore, experiments were carried out to verify the encryption and decryption time and homomorphic operations, presenting a result that has little impact on offloading but does not show the amount of data in which these experiments were carried out. Thus, it is noted that a more detailed analysis was not carried out on the impact of this type of encryption at the processing level for the various scenarios with a large amount of data, which is what happens in a mobile cloud computing scenario.

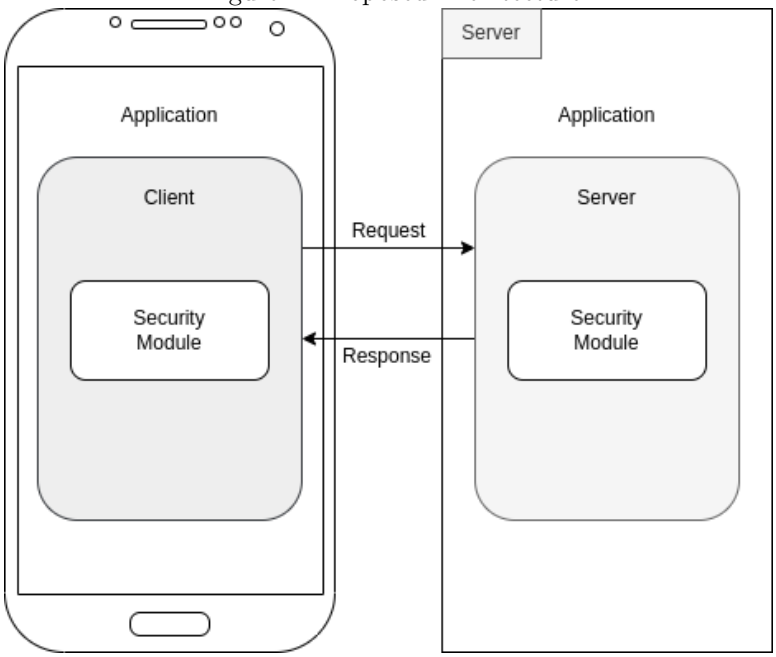
4 Proposal

Previous sections have pointed out relevant issues in the areas of mobile computing (*e.g.*, limited power, storage, and processing) and Mobile Cloud Computing (*e.g.*, lack of security during offloading). Thus, a solution was created that helps develop applications that use computational offloading to mitigate the problem of scarce resources on mobile devices. In addition, such a solution implements security techniques for the privacy of the data transferred during the procedure. Paillier's algorithm was adopted for the homomorphic cryptography process, the main public-key algorithm in which homomorphism is applied to its encryption and addition operations are performed.

4.1 Architecture

Figure 1 presents the proposed architecture. A security module was implemented to have encryption algorithms available to the application developer. The idea is that this module is used to ensure the security of data transmitted during computational offloading and that it is easily expandable to other cryptographic algorithms. The module is composed of two parts: a client and a server. At the beginning of computational offloading, the application invokes the client module to encrypt the data. Next, the encrypted data is sent to the server, which performs the desired computation directly on them in the case of homomorphic encryption. The result is returned to the client once processing is complete. Finally, upon receiving the encrypted result, the client module decrypts it and delivers the raw result to the application.

Figure 1: Proposed Architecture



The modules were developed using the Java programming language. To use homomorphic cryptography, we adopted the external library Javallier[1], which provides a set of methods that encrypt data and compute operations of addition and multiplication of numbers. When the server receives the encrypted data from the client, it can perform different procedures depending on the technique used for data security (in principle, it has the implementation of homomorphic encryption), and the server performs the necessary processing on the encrypted data itself. The result of the operation, of course, will also be encrypted data that will be promptly sent to the mobile application.

4.2 LoadBench

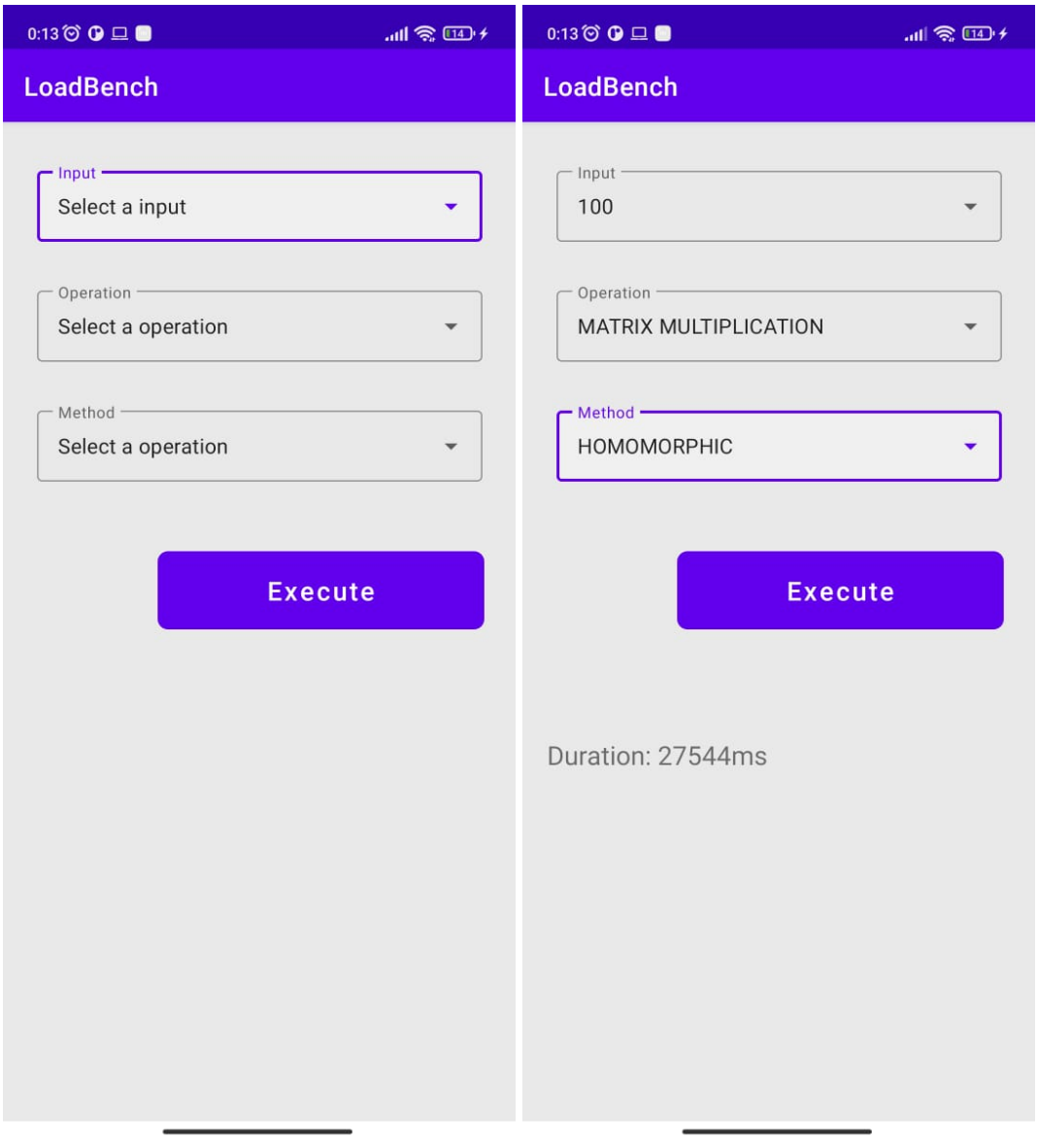
LoadBench[2] is a mobile application developed from the proposed solution that adopts the concept of benchmarking, where it is possible to measure the processing time of the requested/performed operation. The application is divided into a client and a server (allocated, respectively, on a mobile device and a Cloudlet). *LoadBench* supports the offloading of tasks using the homomorphic encryption technique. The tasks computed by *LoadBench* are three mathematical operations: 1) factorial of an integer (from 50 to 200) chosen by the user; 2) sum and 3) multiplication of square matrices, with dimensions of 50x50, 100x100, ..., 1000x1000 (defined by the user) and composed of random values of type *Integer* generated by the application.

The operation of the application and the communication between its components occurs as follows: once the mobile application is started (Figure 2.a), the user chooses the application's input data (matrix dimensions or operand factorial), the desired operation (factorial of a number, addition or multiplication of matrices) and whether to use the homomorphic data

[1] https://github.com/n1analytics/javallier
[2] https://github.com/henrique010/tcc-implementation

protection method or not (unsafe mode). After, the user requests the execution of the operation by clicking the *Execute* button. If the mobile device establishes a connection with the remote server, offloading is performed according to the previously chosen mode. Otherwise, the application processes the task on the device. At the end of the computation, the application displays the time required to compute, locally or remotely, the task (Figure 2.b).

Figure 2: Application Screens of LoadBench



5 Experiments and Results

To evaluate the impacts caused by the adoption of homomorphic encryption techniques during the computational *offloading* of mobile devices, the application *LoadBench* and a server were used to perform the processing requested by the mobile application. This application was used to perform tests, evaluating the total time of *offloading*, communication, and execution of the operation by the server.

5.1 Execution Environment Description

For the experiment, a *smartphone* and a *laptop* were used. The *smartphone* has the following features: Xiaomi Redmi Note 10; Qualcomm Snapdragon 678 2.2Ghz Octa-Core 64-bit processor; 4GB of main memory; 64GB of internal memory; and Android 11.0.0 (Google API 30). The *laptop*, which acts as a *cloudlet*, has the following settings: Acer Aspire A31542G; AMD Ryzen 5 3500u processor; 8GB DDR4 RAM; 256 SSD; Video Card Radeon 540x 2GB; and Ubuntu 20.04 LTS.

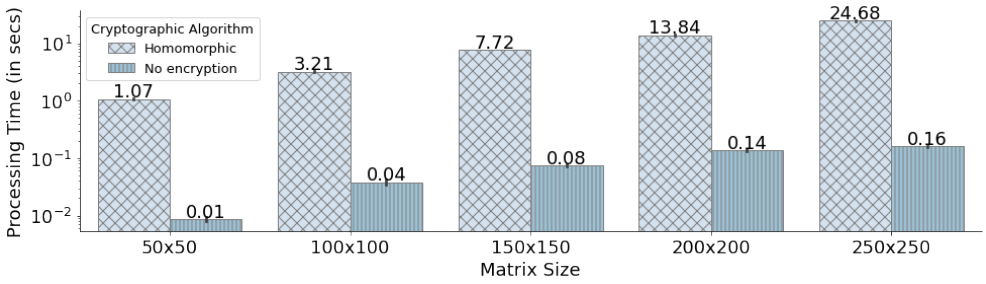
5.2 Description of the experiment

For the present research, the application *LoadBench* was adopted, and the following metrics were evaluated: 1) the processing time of offloading (total); 2) communication time (consists of upload and download time of data), and 3) execution time (time taken by the server to compute the task). For the experiment, 5 ranges of dimensions of rows and columns were empirically chosen for the matrices. Initially, starting with dimensions that always vary by fifty in the difference of rows and columns, such as 50x50, 100x100, 150x150, 200x200, and 250x250. Due to homomorphic encryption presenting RAM memory limitations when encrypting matrices with dimensions greater than 500x500, dimensions of this type were disregarded in the test. Regarding the mathematical operation, we have chosen matrix sum and multiplication operations because of their high computational complexity ($\theta(n^2)$ and $\theta(n^3)$, respectively). We run each scenario thirty times because, according to the Central Limit Theorem (TCL), when you have a sufficiently large sample, the probability distribution of the sample mean can be approximated by a normal distribution [9].

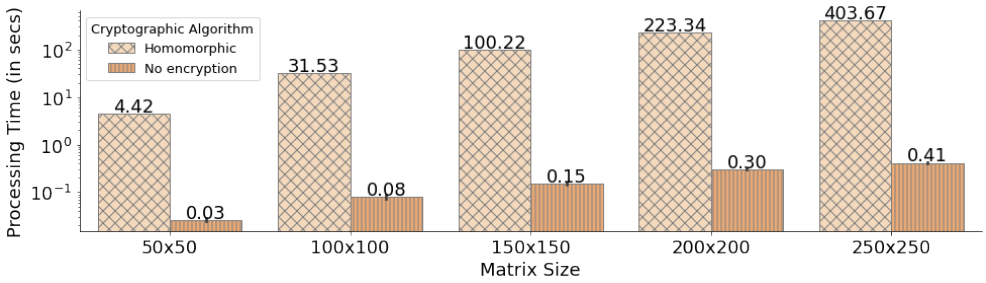
5.3 Results Obtained

Figure 3 presents the average processing time spent with offloading for the Homomorphic encryption algorithm used in the solution. Thus, in the following paragraphs, the results of using this encryption will be compared with the non-use. When comparing the results, we noted that the processing time is higher when adopting homomorphic encryption for all scenarios evaluated. We already expected this behavior, as the application's security layer was added, and it consumed more time to encrypt data before transmission. Notably, when performing the matrix multiplication operation, we noted that the increase in offloading time was 147.3 times for 50x50 matrices and 984.56 times for 250x250 matrices. We observed a similar result to the matrix addition operation, where the offloading using homomorphic encryption was 107 times (50x50) and 154.25 (250x250) slower than without encryption. These results indicate that, even for small arrays, this type of encryption is not suitable for using offloading for mobile applications.

Figure 4 presents the results related to the communication time, execution and encryption in the *offloading* processing. In this case, the No Encryption case (Figures 4(a) and (c)) has the



(c) Sum of Matrices



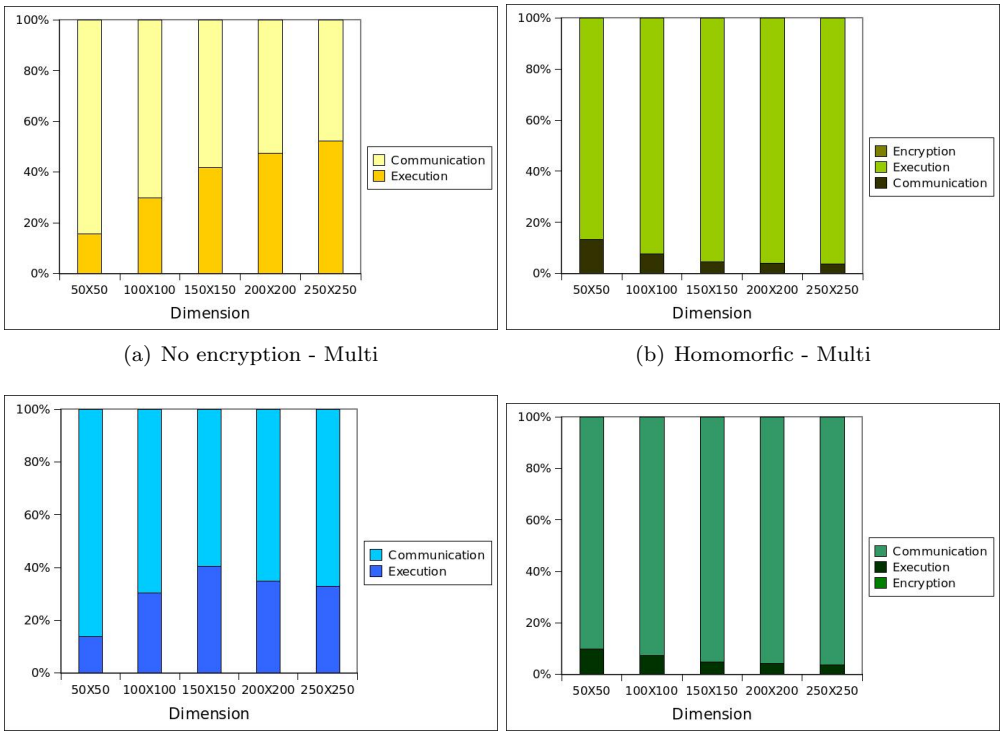
(d) Multiplication of Matrices

Figure 3: Results related to processing time metric (on logarithmic scale)

processing time divided only between communication and execution since no type of encryption is performed. As we observed in previous results when the Homomorphic encryption technique was adopted, most of the total offloading was dedicated to computing the operation on the server (Figure 4(b) and (d)). In addition, regardless of the technique used, as the matrix entries increase, the time spent to compute the operation significantly influences the total time spent in the process. It is also possible to notice that the inverse happens for the encryption time because, as the entries increase, the time spent with encryption proves less influential. From all that has been exposed, it is possible to conclude that using the homomorphic encryption technique will bring benefits related to data security. However, if the problem to be solved has a high complexity (multiplication), it will be a very inefficient process as that the data object to be trafficked has an increasing size. Suppose the problem to be solved is less complex (addition of matrices). In that case, the process takes 43 times more time, which confirms the inefficiency of this type of encryption for mobile devices, but that depending on the method to be performed, this impact it becomes smaller.

5.4 Statistical Data Analysis

Until now, we based our processing time analysis on the average values of the thirty repetitions performed in each scenario. However, there is no guarantee that these average values represent the results related to them well. Thus, we decided to apply a statistical test to assess whether the mean values are good representations of the results and, consequently, to reinforce the observations presented so far. Initially, we divided the results obtained into five groups. Each group contained the results of the two types of offloading performed (with homomorphic encryption



(a) No encryption - Multi

(b) Homomorfic - Multi

(c) No encryption - Add

(d) Homomorfic - Add

Figure 4: Percentage of processing time dedicated to Communication, Execution and Encryption operations

and without encryption) for each array size (50x50 to 250x250). As each group consisted of two subgroups of unpaired samples, we conducted the necessary tests to assess whether or not it would be feasible to apply the T-Student test. By applying the Bartlett test, we verified that the sample variances were not equal for all matrix sizes, which makes using the T-Student test unfeasible. Because of this, we chose the Mann-Whitney test [20]. In the Mann-Whitney test, the null hypothesis (H0) indicates no difference between the evaluated means, i.e., rejecting H0 means that the means differ. Table 1 shows the results of the Mann-Whitney test. We noticed that the null hypothesis was rejected for all matrix sizes. Thus, we conclude that the means shown in Figure 3 are statistically relevant and summarize the behavior of each method of offloading in each scenario.

Matrix Size	**50x50**	**100x100**	**150x150**	**200x200**	**250x250**
Sum of Matrices					
Mann-Whitney	$2.80 \cdot 10^{-11}$	$2.98 \cdot 10^{-11}$	$2.98 \cdot 10^{-11}$	$3.00 \cdot 10^{-11}$	$2.97 \cdot 10^{-11}$
Reject H0?	✓	✓	✓	✓	✓
Multiplication of Matrices					
Mann-Whitney	$2.95 \cdot 10^{-11}$	$3.00 \cdot 10^{-11}$	$3.00 \cdot 10^{-11}$	$3.00 \cdot 10^{-11}$	$2.99 \cdot 10^{-11}$
Reject H0?	✓	✓	✓	✓	✓

Table 1: Mann-Whitney test results related to the processing time metric

5.5 Discussions

The results were quite unfavorable to adopting homomorphic cryptography in a computational offloading scenario. Although some works in the literature have shown that conventional cryptographic algorithms are feasible in computational offloading[14], they still demand that the data be decrypted before task processing effectively, which can be an opportunity to access raw data. Therefore, by overcoming this disadvantage, homomorphic cryptography makes computational offloading even more secure, although significantly slower. This section presents ideas to speed up computing a task using homomorphic cryptography faster, maintaining data privacy.

The first method is caching to map input and output data. The server machine can save the results of operations performed in memory and link them to the received operands. Thus, when receiving a new offloading request with the same entry, the server does not need to compute the same operation again. Instead, it just returns the result saved in memory. Another approach would be to use a more efficient programming language to compute the task. In this case, the developer would need to use the [6] computational offloading multi-language approach. Although more laborious in terms of development, the technique could significantly reduce the processing time of the task and make offloading faster than local processing if the developer chooses the most appropriate server language. A third mechanism would be to improve the hardware of the server machine that will process the task submitted in offloading. The cloudlet used in our experiments has a good hardware configuration for a personal computer. However, it is still weak compared to powerful servers allocated in the Cloud and/or Fog. Therefore, using more powerful server machines can significantly accelerate task processing using homomorphic cryptography, especially when the task is computationally more complex and involves larger input parameters. Even parallel computing can be used to improve the performance of homomorphic encryption. The work [16] introduces a generic method to perform arithmetic operations on encrypted matrices using a homomorphic system and presents the using matrix operations in parallel.

6 Conclusion and Future Works

The present work presented an implementation of a client-server architecture focused on providing security using a homomorphic encryption algorithm for the offloading operation used in the scenario of applications for mobile devices, in addition to having an extensible solution for any encryption algorithm. The purpose is to present a possibility for developers of the *Android* platform and the Java programming language to safely use offloading resources from the implemented algorithms. As a result, it was shown that the module's implementation with the homomorphic cryptography algorithm is evaluated in terms of offloading time, which proved inefficient for the mobile application scenario. A limitation was observed when performing addition and multiplication operations using homomorphic cryptography for matrices with dimensions of 500x500 or greater due to the lack of resources related to the device's main memory when using this technique.

In future works, the following improvements are proposed in this work: *(i)* Implement a Java Annotation, which receives parameters (*e.g.*, security level, required performance) and, from them, defines the cryptographic algorithm to be used; and *(ii)* Analyze the energy consumption of offloading when performed using the proposed solution.

7 Acknowledgments

This work is partially supported by INES[3], CNPq grant 465614/2014-0, FACEPE grants APQ-0399-1.03/17 and APQ/0388-1.03/14, CAPES grant 88887.136410/2017-00

References

[1] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), jul 2018.

[2] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)*, 51(4):1–35, 2018.

[3] Ahmed Aliyu, Abdul Hanan Abdullah, Omprakash Kaiwartya, Syed Hamid Hussain Madni, Usman Mohammed Joda, Abubakar Ado, and Muhammad Tayyab. Mobile cloud computing: taxonomy and challenges. *Journal of Computer Networks and Communications*, 2020, 2020.

[4] Steve Burnett and Stephen Paine. *RSA Security's official guide to cryptography.* McGraw-Hill, Inc., 2001.

[5] Costa et al. Mpos: A multiplatform offloading system. 2015.

[6] Filipe F. S. B. de Matos, Paulo A. L. Rego, and Fernando A. M. Trinta. An empirical study about the adoption of multi-language technique in computation offloading in a mobile cloud computing scenario. In *Proceedings of the 11th International Conference on Cloud Computing and Services Science - CLOSER,*, pages 207–214. INSTICC, SciTePress, 2021.

[7] P. Drozdowski, N. Buchmann, C. Rathgeb, M. Margraf, and C. Busch. On the application of homomorphic encryption to face identification. In *2019 International Conference of the Biometrics Special Interest Group (BIOSIG)*, pages 1–5, 2019.

[8] Niroshinie Fernando, Seng W Loke, and Wenny Rahayu. Mobile cloud computing: A survey. *Future generation computer systems*, 29(1):84–106, 2013.

[9] Hans Fischer. *A history of the central limit theorem: From classical to modern probability theory.* Springer Science & Business Media, 2010.

[10] Caroline Fontaine and Fabien Galand. A survey of homomorphic encryption for nonspecialists. *EURASIP Journal on Information Security*, 2007:1–10, 2007.

[11] Joffre Gavinho Filho, Gabriel P Silva, and Claudio Miceli. A public key compression method for fully homomorphic encryption using genetic algorithms. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1991–1998. IEEE, 2016.

[12] Craig GENTRY. Fully homomorphic encryption using ideal lattices. *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[13] Francisco AA Gomes, Paulo AL Rego, Lincoln Rocha, José N de Souza, and Fernando Trinta. Caos: A context acquisition and offloading system. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 957–966. IEEE, 2017.

[14] Francisco AA Gomes, Paulo AL Rego, Fernando Antonio Mota Trinta, Windson Viana, Francisco Airton Silva, José AF de Macêdo, and José N de Souza. A study about the impact of encryption support on a mobile cloud computing framework. In *CLOSER*, pages 400–407, 2020.

[15] Dijiang Huang and Huijun Wu. Mobile cloud computing taxonomy. In *Mobile Cloud Computing*, pages 5–29. Elsevier, 2018.

[16] Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1209–1222, 2018.

[3] www.ines.org.br

[17] Karthik Kumar, Jibang Liu, Yung-Hsiang Lu, and Bharat Bhargava. A survey of computation offloading for mobile systems. *Mobile networks and Applications*, 18(1):129–140, 2013.

[18] Hai Lin, Sherali Zeadally, Zhihong Chen, Houda Labiod, and Lusheng Wang. A survey on computation offloading modeling for edge computing. *Journal of Network and Computer Applications*, 169:102781, 2020.

[19] Jibang Liu, Karthik Kumar, and Yung-Hsiang Lu. Tradeoff between energy savings and privacy protection in computation offloading. pages 213–218, 2010.

[20] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, 18(1):50 – 60, 1947.

[21] Shahryar Shafique Qureshi, Toufeeq Ahmad, Khalid Rafique, et al. Mobile cloud computing as future for mobile applications-implementation methods and challenging issues. In *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, pages 467–471. IEEE, 2011.

[22] Paulo AL Rego, Philipp B Costa, Emanuel F Coutinho, Lincoln S Rocha, Fernando AM Trinta, and Jose N de Souza. Performing computation offloading on multiple platforms. *Computer Communications*, 105:1–13, 2017.

[23] Wang Ren, Xin Tong, Jing Du, Na Wang, Shan Cang Li, Geyong Min, Zhiwei Zhao, and Ali Kashif Bashir. Privacy-preserving using homomorphic encryption in mobile iot systems. *Computer Communications*, 165:105–111, 2021.

[24] Mahadev Satyanarayanan, Paramvir Bahl, Ramón Caceres, and Nigel Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[25] Leandro V Silva, Pedro Barbosa, Rodolfo Marinho, and Andrey Brito. Security and privacy aware data aggregation on cloud computing. *Journal of Internet Services and Applications*, 9(1):1–13, 2018.