eISSN No. : 2976-2278

UTM Computing Proceedings
Innovations in Computing Technology and Applications
Volume: 6 | Year: 2024 | ISBN No. : 2976-2278

A Comparative Study of Homomorphic Encryptions: Analyzing RSA and Paillier Performance for Data Security in Cloud Computing

Intan Syazriena Mohd Shahidon and Marina Md Arshad
Faculty of Computing
Universiti Teknologi Malaysia
81310 UTM Johor Bahru, Malaysia
isyazriena2@graduate.utm.my, marinama@utm.my

***Abstract*— In the 21st century, scientific computing has shifted from a fixed to a distributed work environment. Current trends in Cloud Computing (CC) enable access to corporate applications from any Internet-connected location. However, there are a number of obstacles that accompany the numerous benefits of CC. The main hitch of organizations to use CC is data security. Especially in the cloud context, when data is dispersed around the globe, this problem becomes significant. Encryption has emerged as a solution, and various encryption methods play a crucial part in cloud data security. The objective of data security is to restrict access to only relevant and authorized users. This thesis purpose is to compare and evaluate between two different encryption algorithms' performance which are RSA and Paillier in terms of key generation time, encryption and decryption time, memory utilization, throughput and file transfer time in a cloud environment. Research is conducted using the GoCJ dataset. The encryption development of both algorithms is done using Java programming language. CloudSim is used for cloud simulation tool where data will be uploaded and downloaded, and the size of files and time taken will be documented. Lastly, the key generation time, encryption and decryption time, memory utilization, throughput as well as encryption file transfer time for each algorithm are compared and discussed. The research result will show the best encryption algorithm among RSA and Paillier in CC security.**

Keywords — Encryption Algorithm, Cloud Computing, RSA, Paillier, Homomorphic Encryption

I. INTRODUCTION

CC offers clients a pool of resources and services in exchange for payment on a per-use basis. According to the National Institute of Standards and Technology (NIST), CC is an architecture that permits ubiquitous, accessible, on-demand network access to a shared pool of configurable computing resources that can be promptly supplied and released with minimal administration effort or service provider engagement [1]. Furthermore, the five main characteristics of CC were described as self-service on demand, wide network access, resource pooling, instant adaptability, and metered service. Unfortunately, cloud services also provide greater opportunity for hackers to gain access to sensitive information and violate privacy. Larry Ponemon, president of the data privacy research from Ponemon Institute, stated that encryption technology is unquestionably important for ensuring the security of network traffic [2]. To ensure the security of data, cryptographic techniques are used to protect information and communication and can be applied in the cloud environment.

Many security difficulties and dangers arise because CSP maintains DC in geographically distributed locations, making it difficult for end users to know where their data resides. CC is accompanied by a number of security problems, including access control, identity authentication, risk management, auditing and recording, integrity control, infrastructure concerns, and dependent hazards [3]. Due to the widespread availability and accessibility of cloud services, the chance of sensitive data slipping into the hands of the wrong people grows. In addition, the expansion of huge data, which must be evaluated, increases the difficulty of decryption. The basic objective of securing what is referred to as digital assets is to reduce or eliminate the potential that these assets would be illegally exposed or abused, given that these assets may be susceptible to external threats [4]. Companies cannot take chances with their critical data. To ensure that cloud-based services deliver not only outstanding quality but also high security features, its encryption mechanism and communication path must comply with the requirements.

To this day, one of the encryption methods that is utilized most frequently is known as RSA [5]. The explanation for this is that it is easy to comprehend, straightforward, and did not demand a significant amount of difficulty to decrypt. A brute

force attack has been up to now the only method available for forcibly decrypting encrypted data generated by the RSA algorithm. Aside from that, the difficulty of decryption increases proportionally with the length of the key that is generated by the RSA algorithm. Paillier encryption, on the other hand, was developed by Pascal Paillier in 1999, is another frequently utilized encryption method, particularly noteworthy for its unique property of homomorphic encryption [6]. The Paillier encryption scheme is asymmetric and relies on the hardness of certain problems in number theory for its security. Like RSA, the strength of the encryption grows with the key size, making brute force attacks infeasible. Its homomorphic properties often make it a more suitable choice for applications requiring privacy-preserving computations [7].

Therefore, this research proposed to compare the performance on the cloud security implementation of Paillier and RSA algorithms for data encryption. This project will implement both techniques in Java that encrypt data at the client side after it has been uploaded to a cloud server, providing an additional layer of data security. The performance of the algorithms will be measured with text files ranging in size from 21 bytes to 413 bytes that are based on the analysis of Google cluster traces. CloudSim cloud simulation tool will be utilized to facilitate the modelling of CC environment for file transfer simulation. Key generation time, encryption and decryption time, memory utilization, throughput and file transfer time on a cloud environment will be analyzed and evaluated to decide whether Paillier or RSA algorithms are better for securing the cloud data.

II. LITERATURE REVIEW

To lay the groundwork for this research, an examination of current literature is undertaken. This includes an investigation into the current state of CC frameworks, their corresponding vulnerabilities, security challenges, and encryption algorithms. Through this methodology, we aim to gain a more comprehensive understanding of the intricacies involved in the implementation of CC and encryption algorithms.

A. Cloud Computing Background and Framework

CC is an evolving kind of IT service in which DC operations are offered as a service. This technology is motivated by the emergence of large-scale, resource DCs developed in low-cost locations. The NIST in the USA defines CC as "A framework for enabling ubiquitous, accessible, on-demand access to a shared pool of configurable computing resources that may be delivered and released with minimal administrative effort or service provider engagement." [1]. This description makes it apparent that CC minimizes an organization's expenditures on resource management and lessens the user's obligation of software and hardware maintenance. When the burden is lessened, the organization spends less money and time on infrastructure management, and the time saved can be employed for creative pursuits [8]. This is a major advantage for users and businesses, since it often provides convenience and also enhances company performance by decreasing the time spent on infrastructure.

CC's architecture has been described by a variety of organizations and researchers. The totality of the architecture can be split down into its most fundamental components: the core stack and the management stack. The core stack consists of three layers: Application, Resource, and Platform [9]. The infrastructure layer comprising of physical and virtualized computing, memory and networking resources is the resource layer. This layer resides underneath the application layer. The platform layer seems to be the most complex component and can be subdivided into numerous layers. For instance, a computing framework is responsible for the dispatching and/or scheduling of transactions and/or tasks. A storage sub-layer allows you to store an infinite quantity of data. The entire system will not be slowed down by a single component because the application server and the other components provide the same application logic as previously and have either an on-demand functionality or flexible administration [9]. Fig. 1 depicts the architecture of CC.

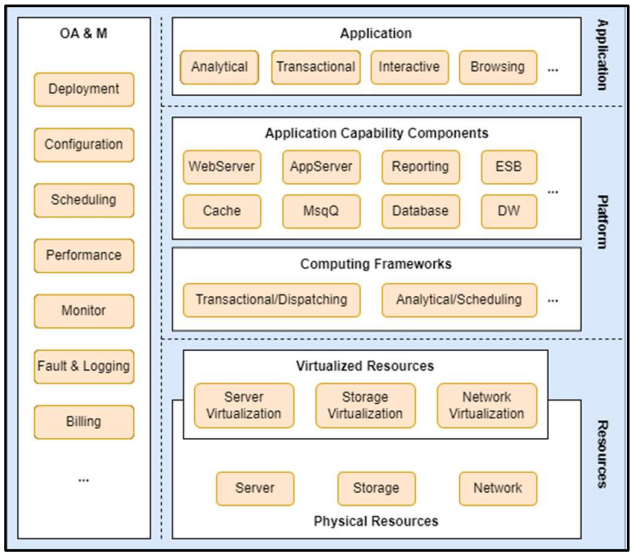

Fig. 1. Cloud computing architecture

B. Cloud Computing Vulnerabilities

When preparing to shift to a CC environment, there are a variety of key vulnerabilities that should be taken into consideration. Using a legitimate session key to obtain unauthorized entry to a computer system's data or services is referred to as session hijacking. This also relates to the stealing of a cookie used for remote server authentication [10]. It refers to vulnerabilities in web application structures, which allow hackers to engage in a variety of destructive behaviors. In session riding, hackers send orders to a web service on behalf of a user by tricking them into surfing the net or opening an email. Session riding deletes user information, conducts online transactions such as bids and orders, transmits spam to an intranet system via the internet, modifies system and web settings, and breaches the firewall [11].

As the primary methods for hacking cryptographic mechanisms and algorithms are known, attackers will be able to decode any cryptographic mechanism or algorithm. It is

common to identify vulnerabilities in cryptographic algorithm implementations. In severe circumstances, these flaws can result in no encryption at all [10]. In cloud virtualization, for instance, service providers employ virtualization software to divide servers into images, which are then made available to consumers as demand-based services [12]. Even if the use of Virtual Machines (VMs) within the DC of cloud providers offers a more convenient and dynamic setup than the use of traditional servers, these VMs do not have sufficient access to generate random numbers necessary for appropriate encryption of data [13].

C. Cloud Computing Security Challenges

Most security threats emerge from within an organization [14]. Given that cloud services are based on a multi-tenant model governed by a central management domain, this problem is complicated for cloud service users. Typically, organizations who subscribe to cloud services lack visibility into the provider's recruiting practices, data storage in multiple locations, and interactions with third-party vendors. Customers of cloud providers are frequently unaware of the hiring standards and procedures for cloud staff [15]. From the vantage point of industrial espionage, the casual attacker, or malicious insiders, this state and fact can give space for enemies. Unrestrictedly, a third-party vendor for the provider can obtain sensitive information and sell it to the victim's competitors.

External threats are amongst the most frightening concerns for any corporation since they directly entail the disclosure of sensitive data or the likely defacement of the organization[14]. This is also a recurring issue in Cloud technology, as Clouds are more interlinked than private networks and have many more interfaces to allow authorized users access data. Hackers and attackers take use of this reality by targeting Application Programming Interface (API) [16] weaknesses, connection tapping or breaching in, and through social engineering.

Organizations expect the same level of data integrity and security when migrating data to the Cloud as they did with on-premises storage. Since Clouds are multi-tenant settings [17], [18] and authentication process may not be at the exact level as on-premises, it is required to prevent unwanted access to sensitive data [19]. This is not as straightforward as it may look, as data loss and leakage can cause monetary, reputational, and consumer harm to the organization. The removal or alteration of data without a copy of the original content is an apparent example. Insufficient authentication, authorization, and internal control, unreliable usage of encryption and encryption keys, equipment breakdown, political pressures, and DC dependability are the most direct and indirect causes of data loss.

When organizations shift information or services to the internet, consumers are ignorant of their location [15], [13] because the supplier may host them anywhere within the Cloud. This is a major concern from the user's perspective, as firms lose control of their valuable information and are uninformed of any security precautions performed by the supplier [13].

D. Encryption Algorithms

In this research, Partially Homomorphic Encryption (PHE) algorithms is utilized in performance analysis of encryption algorithms for the security of data in CC. The performance of encryption algorithms are measured by their effiency, time and space complexity. Therefore, PHE algorithms are chosen because it allows computations to be performed directly on encrypted data, protecting the data's privacy during cloud processing.

1) RSA

The RSA algorithm was introduced in 1978 by Rivest, Shamir, and Adleman [20]. Since its inception, it has dominated as the most commonly used and accepted method for public-key encryption [20]. Furthermore, the RSA scheme possesses multiplicatively homomorphic properties [21]. RSA algorithm is depicted as in Fig. 2.

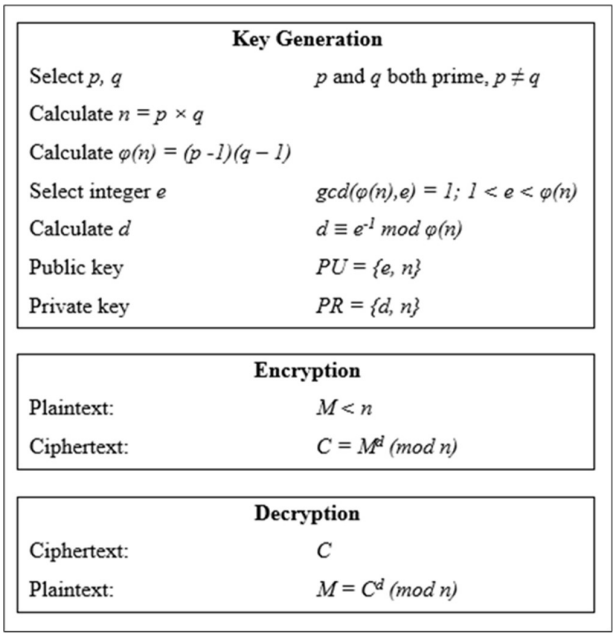

Fig. 2. RSA algorithm

2) Paillier

The Paillier algorithm, a probabilistic public-key scheme, was invented by French researcher Pascal Paillier in 1999 [20]. Characterized by an additive homomorphic property, it is perceived as an extension of the Okamoto-Uchiyama. Its innovation is evidenced under the Decisional Composite Residuality Assumption (DCRA) [21]. Fig. 3 shows the Paillier algoritm.

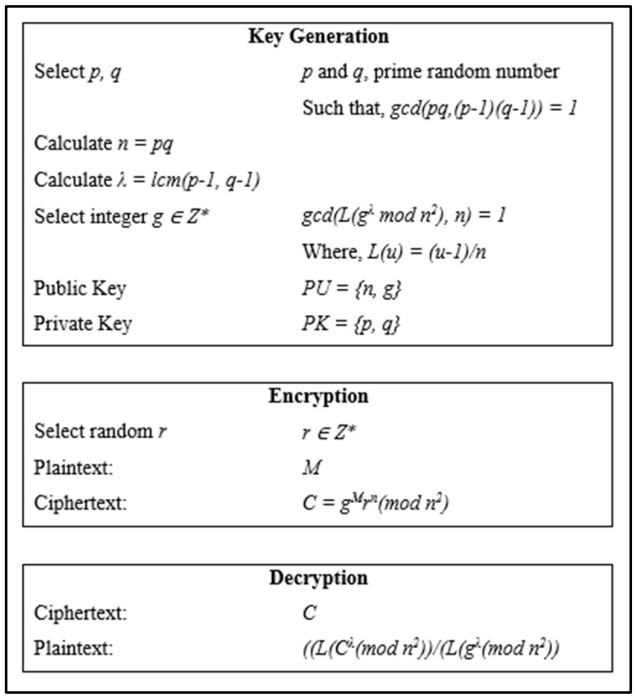

Fig. 3. Paillier algorithm

III. DATASET

The dataset employed in this study is the GoCJ dataset [27]. The primary reason for selecting this dataset is its public accessibility and authenticity, which is designed to inspire other researchers by providing a reliable, open-source benchmark for application comparison. Each file in this dataset comprises a specific number of rows, with each row signifying the MI size of a given job. The GoCJ dataset stands out as a superior choice for this investigation due to its thorough filtration and analysis, making it highly dependable for performance evaluation and assessment within the cloud research community [28]. For the purposes of this study, the text files extracted from this generated dataset will be utilized.

IV. FRAMEWORK FOR ENCRYPTION ALGORITHMS PERFORMANCE ANALYSIS

This research is structured into three distinct phases. The initial phase involves a comprehensive review and exploration of various techniques and characteristics, coupled with data collection. The subsequent phase pivots towards the execution of selected encryption algorithms and the simulation of cloud environment. The final phase is dedicated to analyzing and discussing the outcomes derived from the study.

A. Phase 1: Review of CC Issues, Security Algorithms and Selection of Attributes

The focus of this phase is on studying and reviewing CC challenges and encryption algorithms for the security of data in CC. Specifically, this involves conducting a thorough literature review to understand the existing data security threats in CC. Algorithms properties and analysis of different encryption algorithms are evaluated and two different algorithms, namely RSA and Paillier are identified.

Selection of attributes involves running the GoCJ generator, for creating the result of specific lines of jobs to creating the required dataset. This is initiated by entering an input, which acts as a command indicating the line of job sizes required for the text files. Specifically, the values 15, 25, 35, 45, and 55 are inputted into the generator. These inputs guide the generator to create five distinct results of lines of jobs sizes. The second part of the refinement process focuses on creating smaller lines to complete the dataset. To do this, the value 3 is inputted into the GoCJ generator, which results in the creation of a smaller file that is 21 bytes in size. This completes the dataset creation and provides us with a suitable variety of file sizes to effectively assess and compare the performance of various encryption algorithms.

B. Phase 2:Development of Encryption Algorithms Model and Cloud Environment

1) RSA

This phase involves the development of an RSA encryption model, featuring two simulated entities which are the User and the RSAFileEncryption. The User inputs the data file path for processing and triggers the required functions, while RSAFileEncryption runs these functions. Key generation, encryption, and decryption are the three key functions of the RSA model. Key generation, performed using Java's Big Integer class, uses a variety bit-key sizes, specifically 256, 512, 1024, 2048 and 4096 to encrypt and decrypt six text files of varying sizes ranging from 21 to 413 bytes. The encryption process converts the plaintext message into an integer, then encrypts it using the public key, which consists of Big Integer 'e' and 'n'. The Big Integer class's modPow(e, n) function facilitates this. The decryption process is carried out immediately afterward, utilizing the modPow function of the Big Integer class in Java, which performs modular exponentiation to reverse the encryption process and returns the original plaintext form of the message. RSA encryption model development is visualized in a sequence diagram as Fig. 4.

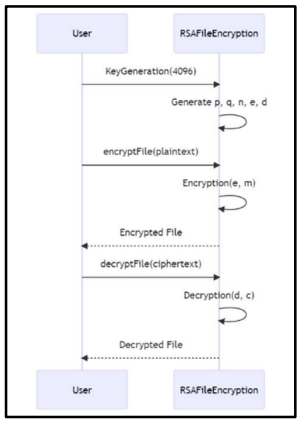

Fig. 4. RSA model sequential diagram

2) *Paillier*

The development of the Paillier encryption model for research and experimentation, similar to the RSA model, involves two main entities and three key functions which are the key generation, encryption, and decryption processes. The key generation is also set with various bit-key lengths of 256, 512, 1024, 2048 and 4096, identical to the RSA experiment, and used to encrypt and decrypt text files of varying sizes. This model leverages the Big Integer class in Java to facilitate operations. In the encryption phase, the plaintext message is transformed into a ciphertext using the public key components generated earlier. For decryption, the ciphertext is reverted back to the original plaintext form using the private key components. These three operations - key generation, encryption, and decryption - are the main functional elements used to evaluate the Paillier encryption model's performance in handling text files of different sizes. As depicted in Fig. 5 is the sequence diagram of the development of Paillier encryption algorithm model.

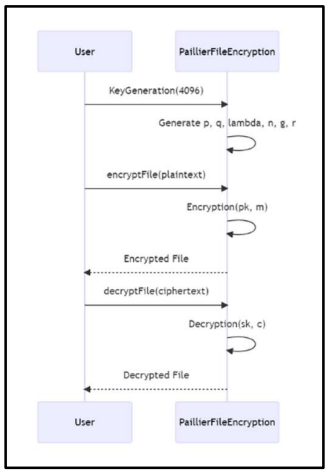

Fig. 5. Paillier model sequential diagram

3) *CloudSim*

The CloudSim model in this development operates through a range of entities such as the client, datacenter, VMs, cloudlets, and a broker, to simulate a cloud environment. The datacenter functions as the cloud service provider, offering infrastructure that houses host systems with processors, storage devices, and memory, which are abstracted and provided as cloud services. VMs, created and registered with a Datacenter Broker, perform file upload operations. The development process begins by initializing the CloudSim package, creating a datacenter with specific characteristics, and setting up a Datacenter Broker. Then, VMs are created and submitted to the broker, and cloudlets, simulating file uploads, are created and also submitted to the broker. Once the simulation runs and is completed, executed cloudlets details such as ID, status, data center ID, VM ID, execution time, start time, and finish time, are retrieved and printed. Host creation and datacenter characteristics are defined in the CloudSim framework, as well as the attributes of the VMs and parameters for cloudlets creation. Fig. 6 illustrates the cloud simulation process used in the development.

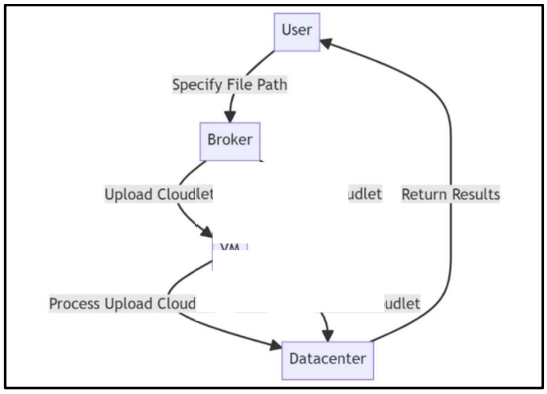

Fig. 6. CloudSim model

4) *Deteministic vs. Probabilistic Algorithm*

The RSA algorithm, in its deterministic nature, employs the Key Generation method where 'p' and 'q' are large prime numbers used to compute 'n' which is part of the public key. The public exponent 'e' is another part of the public key which is predefined in the code. The Encryption method implements the RSA encryption formula utilizing these predetermined values. Conversely, the Paillier encryption algorithm, being probabilistic, uses a setPublicKey method to initialize 'n', 'g', 'nsquare', and the 'bitLength'. These values are predetermined and, along with a randomly generated number 'r' in the Encryption method, are used

to ensure different results for each encryption, contributing to the algorithm's probabilistic nature.

C. Phase 3: Evaluate and Validate RSA and Paillier Encryption Algoritms Performance

The assessment of RSA and Paillier homomorphic asymmetric encryption algorithms is carried out based on selected parameters such as key generation time, encryption and decryption duration, memory consumption, throughput, and the time required for file transfer in a cloud environment.

The experimental procedures are executed on a 12th Gen Intel (R) Core (TM) i5-1235U processor clocked at 1.30Ghz, complemented with an 8.00GB RAM. The system operates on Windows 11 Home, version 22H2, with Java language running on Eclipse IDE version 4.28.0 used for obtaining the algorithm's experimental outcomes. The encryption models are executed 10 times [22] and the average result is discussed, yielding the results delineated in Tables I-VIII.

TABLE I. RSA AND PAILLIER AEVRAGE KEY GENERATION TIME (IN MILLISECONDS)

Method	Key Size				
	256	**512**	**1024**	**2048**	**4096**
RSA	53.3	71	139.7	177.9	955
Paillier	45.9	79.9	140.7	195.1	894.8

TABLE II. RSA AND PAILLIER AVERAGE ENCRYPTION TIME (IN MILLISECONDS)

Method	File Size				
	108	**187**	**257**	**333**	**413**
RSA	12.3	12.1	14.4	12.9	12.1
Paillier	108.1	117.4	132	144.3	158.2

TABLE III. RSA AND PAILLIER AVERAGE DECRYPTION TIME (IN MILLISECONDS)

Method	File Size				
	108	**187**	**257**	**333**	**413**
RSA	22.4	22.3	23.8	22.6	22.4
Paillier	180.8	180.1	180.9	179.2	184.5

TABLE IV. CIPHERTEXT MEMORY UTILIZATION WITH VARIABLE KEY-SIZE (IN BYTES)

Method	Key Size				
	256	**512**	**1024**	**2048**	**4096**
RSA	88	120	184	312	568
Paillier	120	184	312	568	1080

TABLE V. CIPHERTEXT MEMORY UTILIZATION WITH VARIABLE FILE SIZE (IN BYTES)

Method	File Size				
	108	**187**	**257**	**333**	**413**
RSA	568	568	568	568	568
Paillier	1080	1080	1080	1080	1080

TABLE VI. RSA AND PAILLIER AVERAGE ENCRYPTION THROUGHPUT (IN BYTES PER SECOND)

Method	File Size				
	108	**187**	**257**	**333**	**413**
RSA	42055	42533	38572.4	40317.9	39529.1
Paillier	9475.4	8775.6	7790	7118.4	6478.3

TABLE VII. RSA AND PAILLIER AVERAGE DECRYPTION THROUGHPUT (IN BYTES PER SECOND)

Method	File Size				
	108	**187**	**257**	**333**	**413**
RSA	4830.9	8395.7	11066.6	14763.3	18466.1
Paillier	597.4	1041.5	1421.3	1869.6	2259.4

TABLE VIII. RSA AND PAILLIER FILE TRANSFER TIME (IN MILLISECONDS)

Method	Key Size				
	256	**512**	**1024**	**2048**	**4096**
RSA	0.11	0.22	0.26	0.51	1.02
Paillier	0.22	0.25	0.51	1.02	2.05

V. RESULT ANALYSIS

After examining the characteristics of the chosen algorithms and analysing the results described in the tables, the following conclusions were obtained:

- Key generation time: Based on Table I, RSA follows a linear trend, while Paillier initially outperforms RSA for smaller keys but becomes slower for larger ones. Additional steps in Paillier's key generation contribute to the increased time compared to RSA. Overall, both methods show increasing key generation times with larger key sizes.
- Encryption and decryption time: Based on Table II and Table III, RSA outperforms Paillier in both encryption and decryption times, based on the observed results. This can be attributed to Paillier's more complex process involving exponentiation and multiplication operations under a larger modulus, and the need for random 'r' to be coprime with 'n', which demands additional computational effort. Furthermore, Paillier's decryption process includes more complicated steps such as modular exponentiation, division, and multiplication, increasing complexity compared to RSA's simple modular exponentiation approach.
- Ciphertext memory utilization: Based on Table IV and Table V, RSA has a more efficient memory usage than Paillier, as the latter's ciphertext falls within the set of integers modulo n^2, resulting in larger ciphertexts and thus requiring more storage. Regardless of the plaintext's size, the memory used for encryption stays constant - it's determined by the key size, not the plaintext size [20]. Hence, Paillier's larger key size increases memory demands.

- Encryption and Decryption Throughput: Based on Table VI and Table VII, When evaluating encryption and decryption throughput, RSA outperforms Paillier. This is because RSA's faster encryption and decryption times allow more data to be processed in a given timeframe, resulting in a higher throughput. Despite both algorithms operating on the same file size, the quicker speed of RSA means it processes more bytes per millisecond compared to Paillier.
- File transfer time: Based on Table VIII, RSA demonstrates superior performance over Paillier in file transfer time due to its smaller ciphertext size, leading to faster transfers. Conversely, Paillier's ciphertexts are roughly double the size of n, producing larger encrypted files that take more time to transfer, particularly on bandwidth-limited networks.

In conclusion, considering the performance metrics used in this comparison, RSA appears to offer a superior overall performance than the Paillier system. However, the choice of an encryption algorithm shouldn't be made based on these metrics alone. Other key factors must be taken into account such as the homomorphic properties where RSA is a multiplicative and Paillier is additive. Moreover, the nature of the encryption process, whether deterministic or probabilistic, also plays a vital role. Paillier, as a probabilistic encryption algorithm, can generate different ciphertexts for the same plaintext when encrypted with the same key, offering an additional layer of security. Conversely, RSA, being deterministic, will consistently produce the same ciphertext for a particular plaintext when encrypted with the same key.

VI. SUGGESTIONS FOR IMPROVEMENT AND FUTURE WORKS

This study compares two CC security encryption techniques. The dataset focused on text files. Encrypting Portable Network Graphic (png), Portable Document Format (pdf), and Document (doc) files potentially expand the scope. CC emphasises encryption algorithms for client data protection. Security measures can improve algorithm performance analysis in research. Security measures include brute force attack and chosen plaintext or ciphertext. Furthermore, it is recommended to consider the inclusion of other encryption algorithms to enhance the benchmarking process. One such algorithm that could be explored is the ElGamal algorithm, which is a PHE, a probabilistic algorithm and has multiplicative homomorphism. By incorporating ElGamal, the benchmarking process can be further improved allowing more comprehensive evaluation of encryption performance, ultimately contributing to advancements in encryption technology. Lastly, this research focuses on the multiple iteration of model execution for performance measurement which could be further expanded and analyzed using the big O notation by breaking down each operation present in an algorithm into its individual algorithmic steps, assign complexity in each step, sum up the time complexities and compare between the two algorithms [23].

VII. CONCLUSION

This research compares the performance of RSA and Paillier encryption algorithms in the process of encryption and decryption of data for CC. The algorithm which results in better performance in terms of key generation time, encryption and decryption time, memory utilization, encryption and decryption throughput, file transfer time on a cloud environment can be used to be implemented in CC data security.

Throughout the research, RSA and Paillier encryption algorithms are built, and the result is discussed. Phases 1 through 3 of the RSA and Paillier implementation for CC security are described. In phase 1, the dataset was refined to fit research execution according to requirements. RSA and Paillier attributes was then prepared with 6 text files ranging from 3 up to 55 lines of jobs as well as 21 up to 413 bytes of dataset. Next, RSA and Paillier encryption models, and cloud environment are built in phase 2 to achieve the objective in this phase which is generating models that satisfy the key generation, encryption and decryption process of algorithms and cloud simulation that mimics a cloud environment by determining the entities parameters. The results of performance measurements of both algorithms are obtained. In phase 3, the analysis of algorithms performance is evaluated and compared.

It can be seen that RSA model outperformed Paillier model in all key generation time, encryption and decryption time, memory usage, throughput and file transfer time performance measurement. By completing this research, cybersecurity personnel can use this as a reference to identify the encryption algorithms that can perform better in encrypting client's or business's data for security in CC, especially between RSA and Paillier.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Faculty of Computing and Universiti Teknologi Malaysia (UTM) for their invaluable support and resources, which played a crucial role in the successful research.

REFERENCES

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," 2011.
[2] P. Xu, C. Liu, and X.-M. Si, "Fully homomorphic encryption algorithm based on integer polynomial ring," *Jisuanji Gongcheng/ Computer Engineering,* vol. 38, no. 24, 2012.
[3] D. P. Timothy and A. K. Santra, "A hybrid cryptography algorithm for cloud computing security," in *2017 International conference on microelectronic devices, circuits and systems (ICMDCS)*, 2017: IEEE, pp. 1-5.
[4] R. S. Cordova, R. L. R. Maata, A. S. Halibas, and R. Al-Azawi, "Comparative analysis on the performance of selected security algorithms in cloud computing," in *2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 2017: IEEE, pp. 1-4.
[5] J. Thakkar. "Types of Encryption: 5 Encryption Algorithms & How to Choose the Right One." https://www.thesslstore.com/blog/types-of-encryption-encryption-algorithms-how-to-choose-the-right-one/ (accessed.
[6] I. San, N. At, I. Yakut, and H. Polat, "Efficient paillier cryptoprocessor for privacy-preserving data mining," *Security and Communication Networks,* vol. 9, no. 11, pp. 1535-1546, 2016, doi: https://doi.org/10.1002/sec.1442.

[7] K. Munjal and R. Bhatia, "Analysing RSA and PAILLIER homomorphic Property for security in Cloud," *Procedia Computer Science,* vol. 215, pp. 240-246, 2022/01/01/ 2022, doi: https://doi.org/10.1016/j.procs.2022.12.027.
[8] S. Bulusu and K. Sudia, "A study on cloud computing security challenges," ed, 2013.
[9] L. Qian, Z. Luo, Y. Du, and L. Guo, "Cloud computing: An overview," in *IEEE international conference on cloud computing*, 2009: Springer, pp. 626-631.
[10] M. A. Bamiah and S. N. Brohi, "Seven deadly threats and vulnerabilities in cloud computing," *International Journal of Advanced engineering sciences and technologies,* vol. 9, no. 1, pp. 87-90, 2011.
[11] T. Schreiber, "Session riding: A widespread vulnerability in today's web applications," *Whitepaper, SecureNet GmbH (December 2004) http://www. securenet. de/papers/Session_Riding. pdf,* 2004.
[12] B. Grobauer, T. Walloschek, and E. Stocker, "Understanding cloud computing vulnerabilities," *IEEE Security & privacy,* vol. 9, no. 2, pp. 50-57, 2010.
[13] A. Behl, "Emerging security challenges in cloud computing: An insight to cloud security challenges and their mitigation," in *2011 World Congress on Information and Communication Technologies*, 2011: IEEE, pp. 217-222.
[14] W. R. Claycomb and A. Nicoll, "Insider threats to cloud computing: Directions for new research challenges," in *2012 IEEE 36th annual computer software and applications conference*, 2012: IEEE, pp. 387-394.
[15] S. Gadia, "Cloud computing: an auditor's perspective," *ISACA Journal,* vol. 6, p. 24, 2009.
[16] A. Nayyar, "Private virtual infrastructure (pvi) model for cloud computing," *International Journal of Software Engineering Research and Practices,* vol. 1, no. 1, pp. 10-14, 2011.
[17] A. T. Velte, T. J. Velte, R. C. Elsenpeter, and R. C. Elsenpeter, "Cloud computing: a practical approach," 2010.
[18] B. Sosinsky, *Cloud computing bible*. John Wiley & Sons, 2010.
[19] P. You, Y. Peng, W. Liu, and S. Xue, "Security issues and solutions in cloud computing," in *2012 32nd International Conference on Distributed Computing Systems Workshops*, 2012: IEEE, pp. 573-577.
[20] S. J. Mohammed and D. B. Taha, "Performance Evaluation of RSA, ElGamal, and Paillier Partial Homomorphic Encryption Algorithms," in *2022 International Conference on Computer Science and Software Engineering (CSASE)*, 15-17 March 2022 2022, pp. 89-94, doi: 10.1109/CSASE51777.2022.9759825.
[21] H. J. Kiratsata and M. Panchal, "A Comparative Analysis of Machine Learning Models developed from Homomorphic Encryption based RSA and Paillier algorithm," in *2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 6-8 May 2021 2021, pp. 1458-1465, doi: 10.1109/ICICCS51141.2021.9432348.
[22] N. M. S. Iswari, "Key generation algorithm design combination of RSA and ElGamal algorithm," in *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2016: IEEE, pp. 1-5.
[23] R. Vaz, V. Shah, A. Sawhney, and R. Deolekar, "Automated big-o analysis of algorithms," in *2017 international conference on nascent technologies in engineering (ICNTE)*, 2017: IEEE, pp. 1-6.