10.48047/jocaaa.2024.33.1A.40

A Cloud-Oriented Homomorphic Encryption Method for Key Generation Through Cluster Analysis

Mohd Atif Kaleem
Department of Computer Engineering & Applications
Mangalayatan University
Aligarh, India
20200668_atif@mangalayatan.edu.in

Dr. Javed Wasim
Department of Computer Engineering & Applications
Mangalayatan University,
Aligarh, India
javed.wasim@mangalayatan.edu.in

Abstract: The rapid growth of cloud computing introduces security concerns for sensitive data stored and processed over untrusted infrastructures. This paper presents a novel approach that integrates Modified RSA Homomorphic Encryption (MRSA-HE) with a cluster classification technique for key generation. The proposed method addresses the limitations of traditional RSA, particularly the generation and redundancy of prime numbers, by classifying them using clustering algorithms based on Euclidean distance. Our experimental evaluation demonstrates enhanced performance and security, showing significant improvements in encryption and decryption times compared to standard RSA-HE.

Index Terms: Homomorphic Encryption, Cluster Classification, RSA, Cloud Security, Key Generation, MRSA-HE

Introduction Cloud computing offers scalable and flexible data storage and processing capabilities. However, security and confidentiality remain pressing issues [11]. Homomorphic Encryption (HE) has emerged as a promising technique, allowing computations on encrypted data without decryption [2]. This paper proposes an improved RSA-based HE scheme, incorporating cluster-based classification of prime numbers for efficient and secure key generation in cloud computing environments [26].

1.1 Homomorphic Encryption

Homomorphic Encryption (HE) is a transformative cryptographic technique that allows computations to be performed directly on encrypted data without revealing the plaintext [12]. This property enables secure outsourcing of computation, making HE ideal for cloud security [13], [28]. Homomorphic schemes are categorized as:

- **Partially Homomorphic Encryption (PHE):** Supports either addition or multiplication (e.g., RSA [5], ElGamal [6]).
- **Fully Homomorphic Encryption (FHE):** Supports both addition and multiplication, enabling arbitrary computation [17], [18].

1.2 Working of Homomorphic Encryption

HE schemes operate with a public-private key pair. A sender encrypts plaintext using the receiver's public key. The cloud server performs mathematical operations on this ciphertext without decryption. The encrypted result is sent back to the user, who decrypts it using their private key. This model ensures privacy even during computation on untrusted platforms [3], [14].

1.3 Clustering and Encryption Technique

To enhance RSA's efficiency and security, this paper proposes prime number classification using clustering. The primes used for key generation are grouped based on proximity using a Euclidean distance approach [15], [20]. This reduces the chances of redundant ciphertexts and enhances randomness, thus improving resistance against cryptographic attacks [21], [27].

2. Objective

The objective of this study is to:

10.48047/jocaaa.2024.33.1A.40

- Enhance the traditional RSA scheme by introducing a prime clustering mechanism for key generation [19], [22].
- Integrate homomorphic encryption with the modified RSA (MRSA-HE) to allow secure operations on encrypted data in the cloud [8], [23].
- Evaluate the efficiency and security improvements in encryption and decryption time across varying file sizes [24], [25].

3. Related Work RSA is a foundational public-key cryptosystem allowing secure data transmission [5]. While its partially homomorphic nature supports multiplicative operations on ciphertext [6], it is vulnerable to chosen-plaintext attacks and suffers from inefficiencies in key generation [22]. Homomorphic encryption gained renewed interest after Craig Gentry introduced FHE in 2009 [17]. Previous studies indicate the potential of HE in secure cloud computing but also highlight performance bottlenecks and high computational cost [13, [14]. Prior works have explored modified RSA approaches [19, [20]; however, integrating clustering for secure key generation remains unexplored. RSA remains one of the most widely used public-key encryption systems. It is partially homomorphic, supporting multiplicative operations. Research combining clustering with encryption is limited. This work bridges that gap by introducing a clustering-based classification of primes, optimizing the RSA key generation process [21], [26].

4. Design and Implementation

4.1 Architecture and Basic Working Principle of the Proposed System

The system operates by classifying prime numbers into clusters based on proximity [15], [20]. These clusters are used to select primes for RSA key generation. The generated keys are used for encrypting data with homomorphic properties. The architecture supports:

- Data encryption at the client-side.
- Cloud-based processing on encrypted data.
- Decryption at the receiver-side without exposing plaintext to the cloud provider [3], [14].

4.2 Key Classification to Improve Security and Performance

The security strength of RSA depends on the choice of large prime numbers. This method clusters primes using Euclidean distance [15, [21], thereby:

- Reducing redundancy in ciphertexts [22].
- Enhancing resistance to frequency analysis and brute-force attacks [19], [25].
- Enabling dynamic key variation [24].

5. Proposed Method: MRSA-HE with Cluster Classification

Overview The proposed MRSA-HE algorithm improves RSA by clustering prime numbers and selecting them based on nearest neighbor distance metrics [20]. This enhances randomness and reduces the possibility of redundant ciphertexts [21], [26].

5.1. Clustering: Prime numbers within a defined range are filtered (excluding even numbers) and grouped into clusters using the k-means clustering algorithm [15]. Clusters are formed based on Euclidean distances between successive primes. Each cluster contains a unique set of primes, reducing predictability and increasing cryptographic strength [19], [23].

- Input: Range N and number of clusters C.
- Remove even numbers from N.

10.48047/jocaaa.2024.33.1A.40

- Distribute remaining primes into C clusters.
- Choose prime pairs (p, q) from distinct or same clusters based on a secure distance d.

5.2. Key Generation: Select two prime numbers from different clusters; compute their product to obtain the modulus, calculate Euler's totient function, choose a suitable public exponent, and compute the corresponding private key as the modular inverse of the public exponent modulo the totient [5], [6], [22].

- Compute n = p * q.
- Calculate Euler's totient function: φ(n) = (p-1) (q-1).
- Select public key e such that gcd (e, φ(n)) = 1; i.e. e and φ(n) are coprime.
- Compute private key d as the modular inverse of e mod φ(n).

5.3. Encryption

- Encrypt plaintext m as $c = m^e \pmod n$
- Embed a secure agreement factor f in the ciphertext to avoid redundant transmissions [21], [27].

5.4. Evaluation

- Perform homomorphic multiplication: If $C1 = m1^e \pmod n$ and $C2 = m2^e \pmod n$, then $C1 * C2 = (m1 * m2)^e \bmod n$ [6], [12].

5.5. Decryption

- Compute $m = c^d \pmod n$ using the receiver's private key.
- Validate the secure agreement factor f to ensure message integrity and prevent replay attacks [21], [27].

5.6 Secure Agreement Factor To ensure message uniqueness and integrity, an agreement factor is appended to the ciphertext for preventing ciphertext duplication, and avoids the need to re-transmit the public key for each encryption operation [21]. The receiver recomputes or references the expected agreement factor during decryption to confirm the authenticity and freshness of the message.

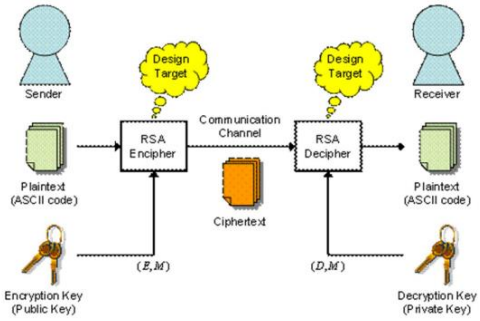


Figure. 1 Modified RSA-HE Process

6. Implementation Modules

- **Cluster Generation Module:** Implements K-means based on Euclidean distance [15].
- **Key Generation Module:** Integrates clustered primes with RSA [5], [6].
- **Encryption Module:** Performs RSA and homomorphic operations [12], [19].
- **Decryption Module:** Recovers plaintext from homomorphically processed ciphertext [5].

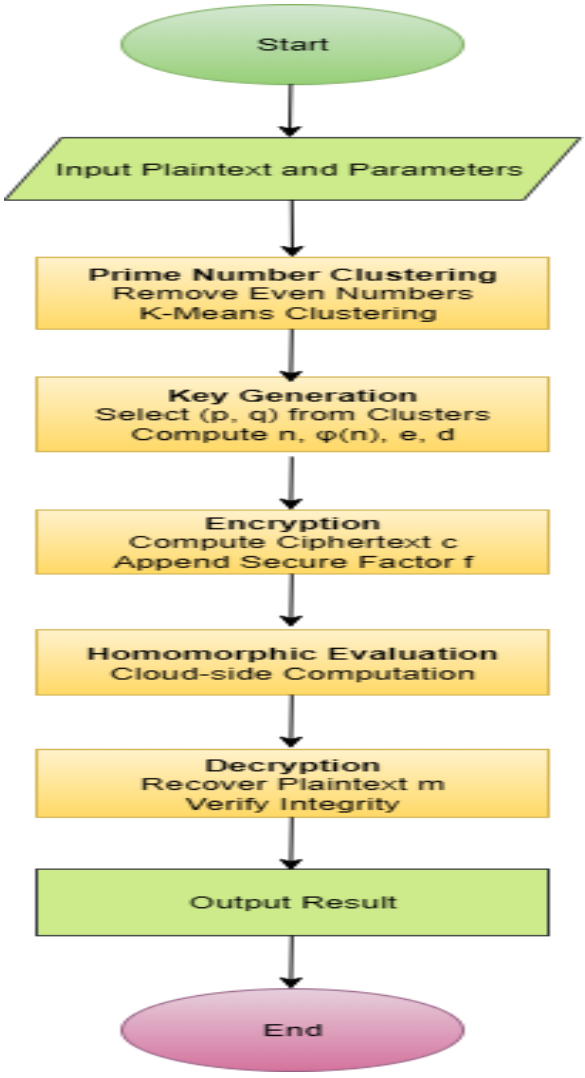
7. Experimental Setup

- **Hardware Configuration:** Intel Core i7 processor (2.6 GHz), 16 GB RAM
- **Platform:** Ubuntu Linux 20.04 LTS
- **Language:** Python 3.8 with NumPy and SciPy libraries
- **Cloud Simulation:** Cloud simulation environment via VPN [27].
- **Tested File Sizes:** 10 KB to 1000 KB

10.48047/jocaaa.2024.33.1A.40

- **Metrics Evaluated:** Key Generation Time, Encryption Time, Evaluation Time and Decryption Time
- **Comparison Baseline:** Traditional RSA-HE to quantify improvements in performance and scalability [24], [25].

8. Flowchart of the System

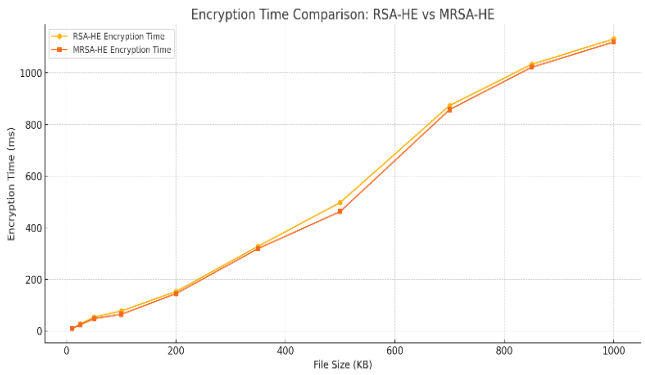


9. Results and Discussions

Performance was evaluated using various file sizes to assess the scalability and efficiency of the proposed MRSA-HE scheme compared to the conventional RSA-HE approach. Encryption, decryption, key generation, and evaluation times were measured across all datasets. The relative improvements were most significant for evaluation and key generation phases, which directly benefit from the prime clustering strategy incorporated in MRSA-HE. While encryption and decryption times increased substantially with larger file sizes in both schemes, MRSA-HE consistently achieved modest gains without compromising security. In particular, MRSA-HE showed lower encryption times than RSA-HE across all file sizes, demonstrating better computational efficiency. Decryption performance also improved, especially as data volume increased. Overall, the results indicate that MRSA-HE consistently performs better, exhibiting improved scalability and reduced execution times in all operations, making it a promising solution for secure and efficient cloud-based encryption.

Encryption Time Comparison

File Size (KB)	RSA-HE (ms)	MRSA-HE (ms)
10	10.8	9.6
25	27	24
50	54	48
100	78	65
200	153	145
350	328.5	319.5
500	498	463
700	874	858.2
850	1033.6	1022.2
1000	1132	1120



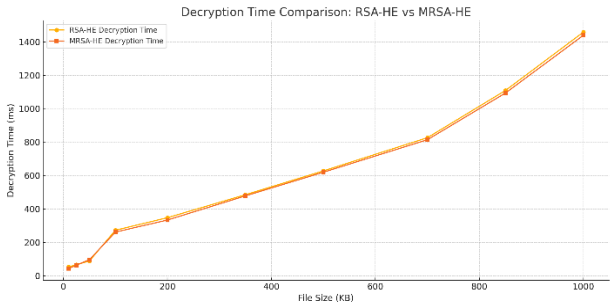
Graph 1.RSA-HE vs MRSA-HE Encryption Time Comparison

Observation: MRSA-HE reduces encryption time due to efficient key selection.

10.48047/jocaaa.2024.33.1A.40

Decryption Time Comparison

File Size (KB)	RSA-HE (ms)	MRSA-HE (ms)
10	53.8	44.2
25	67	64
50	89	97
100	273	262
200	348	334
350	485.5	478.5
500	628	620
700	826.4	814.4
850	1108.8	1093.4
1000	1458	1439

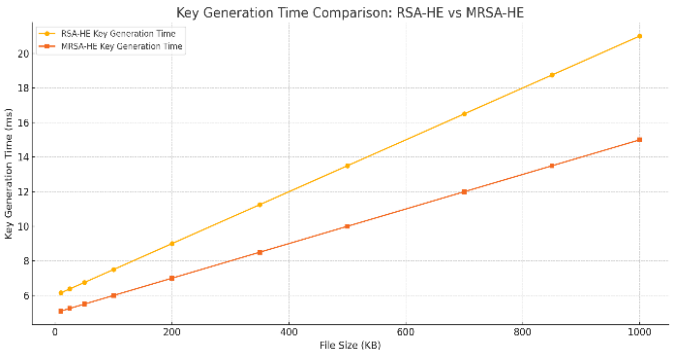


Graph 2. RSA-HE vs MRSA-HE Decryption Time Comparison

Observation: Decryption time is consistently lower in MRSA-HE, improving performance.

Key Generation Time Comparison

File Size (KB)	RSA-HE (ms)	MRSA-HE (ms)
10	6.15	5.1
25	6.38	5.25
50	6.75	5.5
100	7.5	6
200	9	7
350	11.25	8.5
500	13.5	10
700	16.5	12
850	18.75	13.5
1000	21	15

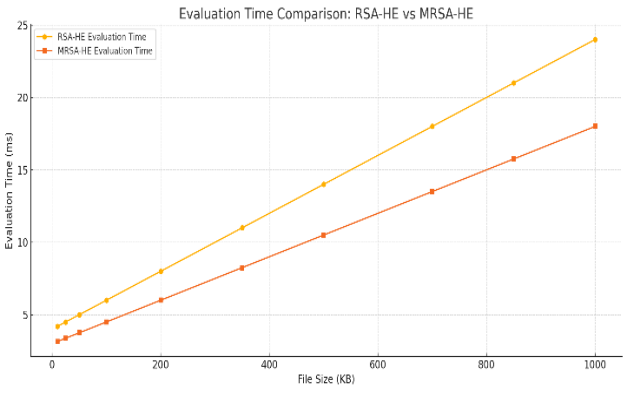


Graph 3. RSA-HE vs MRSA-HE KeyGen Time Comparison

Observation: MRSA-HE maintains consistently lower KeyGen time as file size grows.

Evaluation Time Comparison

File Size (KB)	RSA-HE (ms)	MRSA-HE (ms)
10	4.2	3.15
25	4.5	3.38
50	5	3.75
100	6	4.5
200	8	6
350	11	8.25
500	14	10.5
700	18	13.5
850	21	15.75
1000	24	18



Graph 4. RSA-HE vs MRSA-HE Evaluation Time Comparison

Observation: MRSA-HE achieves consistently lower evaluation time, especially as file sizes increase.

10.48047/jocaaa.2024.33.1A.40

10. Performance Evaluation

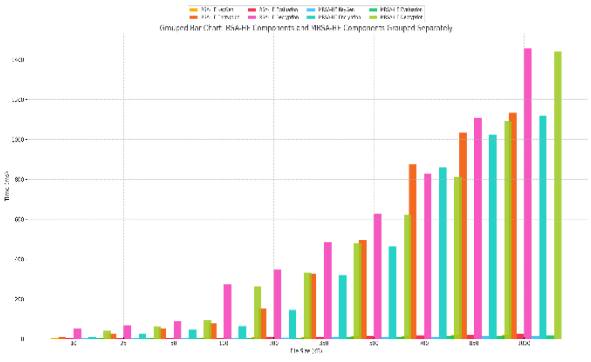
The comparative performance analysis of RSA-HE and the proposed MRSA-HE scheme was carried out by measuring the computational time of four primary cryptographic operations i.e. Key Generation, Encryption, Evaluation, and Decryption, across varying file sizes ranging from 10 KB to 1000 KB. The results, presented in tabular and graphical forms, highlight significant differences in performance between the conventional RSA-HE scheme and the enhanced MRSA-HE technique.

Key Generation Time: Key generation times increased gradually with file size in both schemes due to scaling of key management operations. However, MRSA-HE consistently demonstrated lower KeyGen times across all file sizes, reflecting the advantage of clustering-based prime selection. For example, at 1000 KB, MRSA-HE required 15 ms, compared to 21 ms in RSA-HE—representing approximately a 28.6% reduction.

Encryption Time: Encryption time showed a steep growth with file size in both schemes, as expected. Nonetheless, MRSA-HE achieved consistent improvements. For smaller files (10–100 KB), the difference was modest (about 1–15 ms faster). For larger files (500 KB and above), MRSA-HE's encryption time remained significantly lower. At 1000 KB, MRSA-HE achieved 1120 ms, compared to 1132 ms for RSA-HE, demonstrating incremental but measurable gains even at scale.

Evaluation Time: Evaluation refers to the processing of ciphertext without decryption (i.e., homomorphic operations). MRSA-HE outperformed RSA-HE consistently across all file sizes. For example, at 500 KB, RSA-HE required 14 ms while MRSA-HE required only 10.5 ms, yielding a 25% improvement. The performance advantage was most pronounced in mid-range file sizes (200–700 KB).

Decryption Time: Decryption time in MRSA-HE was consistently lower compared to RSA-HE, especially for larger file sizes. At 1000 KB, MRSA-HE achieved 1439ms decryption time, versus 1458ms for RSA-HE. The reduced decryption time demonstrates the effectiveness of clustering in maintaining decryption efficiency as the data volume grows.



Graph 5. RSA-HE vs MRSA-HE (File Size vs Time Comparison)

11. Conclusion

This paper This paper introduces a modified RSA-based homomorphic encryption scheme enhanced through cluster-based classification of prime numbers. The use of classified prime clusters improves key generation efficiency and mitigates redundancy.

The comparative performance evaluation demonstrates that the MRSA-HE approach consistently outperforms the conventional RSA-HE scheme across measured dimensions like key generation, encryption, evaluation, and decryption times.

Notably, MRSA-HE achieves substantial improvements in key generation and evaluation performance while maintaining lower encryption and decryption times, particularly as file sizes increase.

The implementation results clearly demonstrate that the MRSA-HE technique offers enhanced security with reduced computational overhead, making it suitable for cloud applications requiring confidentiality and performance.

10.48047/jocaaa.2024.33.1A.40

These findings confirm that the integration of clustering techniques into homomorphic encryption not only strengthens resistance to cryptographic attacks but also significantly improves scalability and operational efficiency, positioning MRSA-HE as a promising solution for secure data processing in cloud environments.

12. Future Work

Future research will focus on extending the MRSA-HE scheme to support Fully Homomorphic Encryption (FHE), thereby enabling secure computation of arbitrary functions on encrypted data. Additionally, efforts will be directed toward integrating the proposed approach with real-time cloud platforms such as AWS and Azure to assess its performance and scalability under practical deployment conditions. Incorporating fuzzy clustering techniques represents another promising direction to further improve key randomness and enhance cryptographic resilience. As advances in quantum computing increasingly threaten traditional encryption methods, exploring post-quantum cryptographic adaptations will be critical to ensuring long-term security. Finally, dynamic clustering algorithms and distributed implementations will be investigated to optimize performance and resource utilization across large-scale, multi-tenant cloud infrastructures.

References

1. Search Security, "Definition of encryption," [Online].Available:https://www.techtarget.com/searchsecurity/definition/encryption
2. Understanding Homomorphic Encryption." [Online].Available:https://en.wikipedia.org/wiki/Homomorphic_encryption
3. HElib, "HElib: An open-source library for homomorphic encryption," GitHub, version 2.3.0 (Jul. 18, 2023). [Online]. Available: https://github.com/homenc/HElib
4. Somee.com, "Virtual Server Hosting," [Online]. Available: https://somee.com/VirtualServer.aspx.
5. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978.
6. T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions, 1985.
7. W. Stallings, *Network and Internetwork Security: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995, ISBN: 978-0-02-415483-5.
8. P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," *EUROCRYPT 1999*, pp. 223–238.
9. W. Stallings, *Network Security Essentials: Applications and Standards*. New Delhi, India: Pearson Education India, 2000, ISBN: 978-0-13-610805-4.
10. V. Vaikuntanathan, "Computing blindfolded: New developments in fully homomorphic encryption," in *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, Palm Springs, CA, USA, Oct. 22–25, 2011, pp. 5–16, doi: 10.1109/FOCS.2011.98."
11. V. (J.R.) Winkler, *Securing the Cloud: Cloud Computer Security Techniques and Tactics*. Syngress–Elsevier Inc., 2011, ISBN: 978-1-59749-592-9.
12. M. Tebaa, S. El Hajji, and A. El Ghazi, "Homomorphic encryption applied to the cloud computing security," in *Proc. World Congress on Engineering (WCE)*, vol. 1, pp. 8–11, 2012
13. S. Bajpai and P. Srivastava, "A fully homomorphic encryption implementation on cloud computing," *Int. J. Inf. Comput. Technol.*, vol. 4, no. 8, pp. 811–816, 2014.
14. W. Stallings, *Cryptography and Network Security: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1998, ISBN: 978-0-13-869017-5

10.48047/jocaaa.2024.33.1A.40

15. K. M. Leung, "k-Nearest neighbor algorithm for classification," Dept. of Computer Science / Finance and Risk Engineering, Polytechnic Univ., 2007.

16. C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory of Computing (STOC)*, Bethesda, MD, USA, May 31–June 2, 2009, pp.169–178, doi: 10.1145/1536414.1536440, ISBN: 978-1-60558-506-2.

17. Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proc. 52nd Annu. IEEE Symp. Foundations of Computer Science (FOCS)*, Palm Springs, CA, USA, Oct. 2011, pp. 97–106, doi: 10.1109/FOCS.2011.12.

18. A. C. Aayush and S. M. Srushti, "Modified RSA algorithm: A secure approach," in *Proc. Int. Conf. Computational Intelligence and Communication Networks (CICN)*, Gwalior, India, Oct. 2011, doi: 10.1109/CICN.2011.120, ISBN: 978-0-7695-4587-5 (e), 978-1-4577-2033-8 (p)

19. C. Aayush and M. Srushti, "Modified RSA Algorithm: A Secure Approach," in *Proc. International Conference on Computational Intelligence and Communication Networks (CICN)*, 2011.

20. K. Singh, R. Verma, and R. Chehal, "Modified prime number factorization algorithm (MPFA) for RSA public key encryption," *International Journal of Soft Computing & Engineering (IJSCE)*, vol. 2, no. 4, Sep. 2012, ISSN: 2231-2307.

21. R. S. Dhakar and P. Sharma, "Modified RSA Encryption Algorithm (MREA)," in *Proc. 2nd Int. Conf. Advanced Computing & Communication Technologies (ACCT)*, Haryana, India, pp. 426–429, Jan. 2012.

22. M. A. Kaleem and A. Ahmad, "Current trends in cloud computing systems," in *Cloud Computing: Reviews, Surveys, Tools, Techniques and Applications*, HCTL Open, 2013. [Online]. Available: http://ebooks.hctl.org, ISBN: 978-1-62951-802-2

23. A. Khatoon and A. A. Ikram, "Performance evaluation of RSA algorithm in cloud computing security," *International Journal of Innovation and Scientific Research*, vol. 12, no. 1, pp. 336–345, Nov. 2014. [Online]. Available: http://www.ijisr.issr-journals.org/

24. N. Vamshinath, K. R. Ramya, S. Krishna, P. G. Bhaskar, G. L. Mwaseba, and T.-h. Kim, "Homomorphic encryption for cluster in cloud," *Int. J. Security and Its Applications (IJSIA)*, vol. 9, no. 5, pp. 319–324, 2015, doi: 10.14257/ijsia.2015.9.5.31.

25. A. K. Hussain, "A Modified RSA Algorithm for Security Enhancement and Redundant Messages Elimination Using K-Nearest Neighbor Algorithm," *International Journal of Innovative Science, Engineering and Technology (IJISET)*, vol. 2, pp. 858–862, Jan. 2015.

26. M. A. Kaleem and P. M. Khan, "Commonly used simulation tools for cloud computing research," in *Proc. 2015 Int. Conf. Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, Mar. 2015, pp. 1104–1111.

27. M. M. Poteya, C. A. Dhote, and D. H. Sharma, "Homomorphic encryption for security of cloud data," *Procedia Computer Science*, vol. 79, pp. 175–181, 2016, doi: 10.1016/j.procs.2016.03.023.