

Optimizing Encrypted Computation in Cloud Environments: A Residue Number System-Based Approach

John Fitzgerald Kennedy Asewie¹, Yaw Marfo Missah², Emmanuel Ahene³

^{1, 2, 3} Kwame Nkrumah University of Science and Technology, Kumasi, Ghana

Abstract

Nowadays, cloud computing has become a default platform for encrypted computation of the data, and obscure and fast encrypted computation can be critical. Nowadays RSA and AES, among others, are devices providing different levels of security, but are relatively inefficient for advanced operations on encrypted data within very large-scale cloud environments. Since there are performances bottlenecks with the application of Fully Homomorphic Encryption (FHE), computation on ciphertexts is allowed but operating on ciphertexts can be slow. We explore how to integrate the Residue Number System (RNS) into encrypted computation frameworks, considering in particular encryptions in the cloud. The proposed framework acquires appreciable improvements on encryption/decryption latency, memory efficiency, and scalability, by making use of RNS properties, namely, carry free arithmetic, modular parallelism, and Chinese Remainder Theorem-based reconstruction. However, RNS based encryption is faster up to 10x faster with respect to FHE and takes much less memory than RSA. Results show this by means of theoretical exposition and experimental evaluations on synthetic and real-world data. This serves as a promising direction of the way to boost secure cloud computations in the case of latency sensitive and resource constrained environments.

Keywords: Residue Number System, Homomorphic Encryption, Cloud Security, Encrypted Computation, Modular Arithmetic, Memory-Efficient Cryptography, Parallel Processing, Chinese Remainder Theorem, Privacy-Preserving Computation, Cryptographic Optimization, Post-Quantum Cryptography.

1. Introduction

Exponential growth of digital data has raised the tendency of handling the data storage and computation in the cloud. Although providing unparalleled scalability, availability and economic efficiency, concern with regard to the privacy and security of data is mandatory. Such risks become intolerable as organizations and individuals deposit their sensitive data into third party platforms. As [1] indicates, data privacy is a major issue for more than 60% of enterprises when it comes to cloud migration. Risk for these have been mitigated by the use of cryptographic techniques like RSA and AES. Although RSA gives secure key exchanges and AES provides fast symmetric encryption, these two algorithms are not able to accomplish the tasks that need to be performed on encrypted data that is so crucial in privacy preserving outsourced computation. To fill the gap, we have introduced Fully Homomorphic Encryption (FHE) that allowed doing the computation on encrypted data without decrypting it. While promising from theory, FHE remains computationally expensive and infeasible for large scale cloud applications because of high

latency, huge amount of memory consumption, and inefficient arithmetic operations. Therefore, there is a need for low latency, scalable, and secure computation and the research includes seeking an alternative computational model which can speed encrypted operations without compromising security. An example of such models is the Residue Number System (RNS). Large integers are represented in RNS as a set of smaller, independent residues that allow parallel and carry free arithmetic operations [2] and are thus suitable for high-speed processors and cloud computing.

This paper investigates the application of RNS to homomorphic encryption schemes in the context of cloud computing and the potential that integration of RNS and homomorphic encryption schemes can bring to cloud performance. The paper in particular investigates the RNS ability to compress the bounds on encryption and decryption latency, utilize memory efficiently, and speed up the fundamental's homomorphic operations like addition and multiplication.

2. Related Work

The main research driving such cryptographic methods is security computation in cloud environments and providing the privacy of data while processing. Yet, many traditional encryption schemes like RSA and AES are still very much relied upon to maintain confidentiality as well as secure data transmission. However, using RSA for large key sizes is computationally expensive. However, symmetric key algorithm, such as AES, is fast and suitable for large data blocks. Unfortunately, neither can directly compute on encrypted data, hindering their use in privacy preserving outsourced computing [2]. Additionally, data is required to be decrypted before processing, whereby such decryption is subject to potential threats in untrusted cloud environments.

To address this a paradigm shattering technology called Fully Homomorphic Encryption (FHE) was put forward which allows for additional as well as multiplication done over encrypted data without prior decryption. However, FHE promises true end to end confidentiality, however, with high computational overhead, long processing times and large memory demands it is impractical for real time cloud applications [3][4]. Because homomorphic operations on large encrypted integers are so complex, deployment of the operation in dynamic cloud settings can be difficult.

In response to this, researchers have examined the use of Residue Number System (RNS) in order to optimize the arithmetic intensive cryptographic computations. An integer is represented in RNS as a set of smaller independent residues chosen modulo pairwise coprime moduli. With such great carry freeness, high parallelization, and internally lossless, this will work well on modern hardware architecture used for high performance computing [4] [5]. Such properties greatly do well in processing speed and efficiency over that of conventional positional number systems. RNS has also been attempted to be leveraged in the cryptographic implementation in various studies. In public-key cryptography, it has been demonstrated by [6] that RNS can speed up modular exponentiation. On top of this, [7] further extended it to elliptic curve operations using RNS with significant gains in throughput and memory efficiency. Yet most of these efforts have been directed towards the usual encryption protocols as opposed to homomorphic encryption. Moreover, they lack the capability to handle the special aspects of real-time encrypted computations in the cloud.

However, this is a critical research gap considering FHE and RNS have been immensely studied separately, however there has been little exploration of the hybrid of RNS within homomorphic encryption frameworks considering its cloud application specifically. Potential practical, high speed, and secure encrypted computation for applications in modern cloud architectures could then be bridged over this gap.

3. Theoretical Foundation of RNS-Based Encrypted Computation

The Residue Number System (RNS) is a non-weighted numeral system that expresses an integer as a set of residues modulo a group of pairwise coprime integers. Formally, an integer $X \in [0, M - 1]$, where $M = m_1 \cdot m_2 \cdot \dots \cdot m_n$ and each m_i is a positive, pairwise coprime modulus, is denoted by its residue factor $x_1, x_2, \dots, x_n$ such that $x_i = X \bmod m_i$. This formula is a Cartesian product of $Z_{m_1} \times Z_{m_2} \times \dots \times Z_{m_n}$.

A principal strength of RNS lies in its carry-free arithmetic, a property that significantly reduces computational overhead compared to positional systems like binary or decimal. Since each modular computation is independent, parallel processing can be naturally exploited. This feature aligns well with the performance demands of homomorphic encryption (HE), where modular arithmetic operations dominate both encryption and homomorphic evaluation processes [8].

To integrate RNS into homomorphic schemes, like Brakersky, Gentry and Vaikuntanathan (BGV) each ciphertext component is also implemented in RNS form. Parallel homomorphic operations such as additions and multiplications over each modulus are performed in residue domain and the homomorphic property is maintained through this enhancement of efficiency, since carry propagation does not occur in each modulus [9]. RNS is an important feature that allows batching strategies and (Single Instruction, Multiple Data) SIMD-style processing, thus further optimizing encrypted computation in cloud settings.

To recover the original plaintext from its RNS representation post-decryption, the Chinese Remainder Theorem (CRT) is used. CRT reconstructs the unique integer X from its residues using the relation:

$$X = \sum_{i=1}^n x_i \cdot M_i \cdot y_i \bmod M \quad \text{where } M_i = M/m_i \text{ and } y_i = M_i^{-1} \bmod m_i$$

This recovery process is both exact and efficient, ensuring the reliability of RNS in secure computation pipelines.

However, RNS design introduces tradeoffs. The dynamic range, security level and computational efficiency are all affected by modulus selection. Both parallelism and range are maximized by smaller moduli, but poorer representable range, and both range and speed of hardware may be impaired by large moduli. Practical homomorphic applications require cryptographically sound while keeping a balance between system performance.

4. Methodology

This section discusses the RNS based encrypted computation architecture and its workflow designed for secure and efficient processing in cloud environment. It is modular architecture with four main stages: Input Conversion, Residue Encoding, Encrypted Operation, CRT based Decryption. Plaintext values are preprocessed for the input conversion stage to integer formats that are supported for RNS encoding. These inputs are mapped to their corresponding residues using a chosen set of coprime moduli in an optimal manner for both parallelism and representational range. The residue encoding module applies to the input to convert it to its RNS representation, in which each residue can be processed in parallel, permitting native parallelization of arithmetic operations.

Homomorphic additions and the multiplications are carried out in the encrypted operation module. After an HE scheme has been used to encrypt the ciphertext, they are evaluated in the RNS domain. Such element-wise encrypted computations across different moduli incur reduced latency [10], and for this reason, they are carried out in a carry free, distributive manner, using the RNS polynomial. Parallel arithmetic pipelines are used for this step to increase throughput and scalability in cloud simulation in the cloud.

The second part of the module is a CRT based decryption module using which the final result is restituted using Chinese Remainder Theorem (CRT) on the residues processed from CRT as above. All this is done efficiently with precomputed inverse tables and moduli products. In order to protect the process, key distribution mechanisms and multi-layer encryption protocols are combined to make it immune to known cryptanalytic attacks and side channel threats [11].

Python, as well as NumPy for fast numerical operations, and SymPy for symbolic residue computation and CRT, are used to implement the system. To simulate and evaluate the framework in cloud, the resource usage and response time were evaluated using CloudSim toolkit and OpenStack. To validate the experimental side, synthetic data was used, along with the Enron email dataset, to model structured data with semantic features and also test for scalability and stress conditions.

4.1 Experimental Evaluation

In order to evaluate the practical effectiveness of the above mentioned RNS-based encrypted computation framework, we compared the performance of the proposed framework with that of traditional encryption schemes (i.e RSA and Fully Homomorphic Encryption (FHE)) through a series of controlled experiments. To measure the execution efficiency, memory usage and scalability under cloud like computational conditions, the experiments were designed.

4.2 Experimental Setup

The device used to run all experiments was Intel Core i9-11900K @ 3.50GHz with 64GB DDR4 RAM and Ubuntu 22.04 LTS. Python 3.10, NumPy 1.24, SymPy 1.11 and CloudSim 3.0 was the software stack used for implementing simulation. The RSA encryption was run with the PyCrypto, the FHE operations with the Microsoft SEAL library, and the RNS based framework with custom modular arithmetic for parallel optimization. All schemes drew from a consistent dataset consisting of 10,000 encrypted values derived from the Enron email dataset, where fields of the structured data are mapped onto numerical formats. To evaluate scalability, each encryption scheme was tested under the same conditions with increasing data sizes ranging from 10K, 50K to 100K records.

5. Results

The evaluation focused on three core performance metrics to assess the efficiency of the RNS-based encrypted computation framework in comparison to RSA and FHE schemes. Firstly, encryption and decryption latency were measured as the average time taken to encrypt and decrypt individual data records stored in a cloud-hosted database. The RNS model significantly reduced latency, showing up to 4 times faster decryption than RSA and 10 times faster than FHE. Peak memory usage during encryption operations was recorded, with RNS demonstrating a 3 times lower

memory footprint compared to RSA and a 5 times improvement over FHE, due to its modular and parallel structure. Scalability was evaluated by analyzing how throughput and response time varied with increasing data input sizes. The results indicated that the RNS-based approach maintained high throughput and stable response times under load, outperforming RSA and FHE in terms of efficient handling of larger datasets, thus making it ideal for cloud-scale encrypted computation.

Encryption and Decryption Latency

In terms of latency, the RNS framework we proposed greatly exceeded the RSA and FHE. On average: In decryption, RNS was 4 times faster than RSA because it did carry free and parallel residue arithmetic. In terms of speed, RNS was around 10 times faster than FHE because RNS operations have lower computational depth, and have efficient modular representation (as shown in Table 1)

Table 1: Encryption and Decryption Speed

Scheme	Avg. Encryption Time (ms)	Avg. Decryption Time (ms)
RSA	12.5	11.4
FHE	87.2	93.1
RNS	5.8	2.9

Memory Usage

RNS demonstrated 3 times better memory efficiency than RSA and over 5 times less than FHE (as shown in Table 2), due to the little overhead required for residue representation, and the use of relatively simple key switching and bootstrapping mechanism for FHE [12].

Table 2: Memory usage by RSA, FHE and RNS

Scheme	Peak Memory Usage (MB)
RSA	220
FHE	480
RNS	68

Scalability

When tested with growing datasets, RNS scaled linearly, while FHE's computational burden increased exponentially, especially during ciphertext evaluations. RNS maintained consistent performance due to the independent and parallel processing of residues [12]. Table 3 shows the scalability of the RNS, RSA and FHE under increased data sizes

Table 3: Scalability test results of RSA, FHE and RNS

Data Size	RSA Time (s)	FHE Time (s)	RNS Time (s)
10K	14.3	85.7	5.1
50K	73.2	479.0	23.7
100K	146.5	991.8	48.9

6. Discussion

The experimental results show that encrypted computation with RNS proves to be a powerful optimization tool for cloud based secure processing. Low latency is possible due to its inherent carry free and parallel structure and memory efficiency is possible without redundant key management and ciphertext expansion prevailing in RSA and FHE. The aforementioned properties drive RNS as a strong contender for being practically deployed in the encrypted varieties of latency sensitive cloud applications where large-scale data operations and near real time processing are also mandatory [12] [13]. The proposed RNS based encrypted computation framework is demonstrated to be resistant to cryptographic attacks and very scalable for clouds. The main strength of RNS approach is its capability to store more modular keys that naturally gives brute force resistance. Using a collection of many carefully chosen prime moduli, the residue representation space is made to be substantial and nonlinear that renders all exhaustive attack infeasible. Its security grows with the number and size of moduli and at some point, its entropy is comparable to or even better than RSA and ECC of the same key length.

Carry free arithmetic and modular independence allow the RNS system to have high tolerance for faults in the noisy or lossy environment such as a cloud infrastructure with share resources. One modulus computation is independent of computations in other residue channels, so an occasional bit error or transient fault will have no effect on the rest of the computation. Error correction mechanisms can be supported in this structure as the encrypted computation can make use of the partial residue computation to reduce its reliability [14].

Due to the concurrent data processing and multi-tenants accessing the data in the cloud computing environment, RNS is beneficial. It works in parallel on multiple moduli, data of several users at the same time,

but there are no collision or leakage in the data. It allows us to horizontally scale encrypted workflows with negligible overhead regardless of the number of client workloads to be processed concurrently with no loss of confidentiality or slowdown.

Table 4: Comparative Security Overview

Scheme	Security	Performance	Flexibility
AES	High (symmetric)	Very Fast	Low (limited to block ciphers)
RSA	Very High	Slow	Moderate (supports key exchange, signatures)
RNS	High (modular entropy)	Fast	High (homomorphic-friendly)

From Table 4, AES, though fast in speed, is not suitable for computations on encrypted data. However, RSA, though secure, is quite slow, an issue in large scale environment. RNS trades off between secure encrypted operations that are subject to acceptable latency, which is well suited to privacy preserving cloud computing and encrypted data analytics [14].

However, the framework based on the RNS also has a significant limitation in that it only relies on structured numerical data. Preprocessing (tokenization, numerical mapping, feature extraction, etc.) has to be applied to unstructured formats (text, images, etc.) before encryption. This introduces latency and complexity into real time application. However, this limitation can be alleviated with the good data preprocessing pipelines.



Figure 1. Comparison of RNS, RSA and FHE

According to Figure 1 the RNS based encryption scheme presents the performance advantages of the proposed method in terms of the encryption/decryption latency, memory usage and scalability on various key metrics like communication overhead, computational cost, and deserialization latency. In particular, RNS has much lower encryption and decryption time, up to 4 times

than RSA and 10 times faster than FHE, which illustrates that RNS is low latency. In addition, being memory efficient as compared with RSA, RNS involves roughly one-third less memory than RSA, and far less memory than FHE. Additionally, RNS incurs less overhead when the sizes of inputs increase, preserving performance in a way that RSA and FHE perform poorly. Together, these findings put RNS on a considerably more practical and efficient footing as an approach for secure computation in a cloud environment.

7. The framework

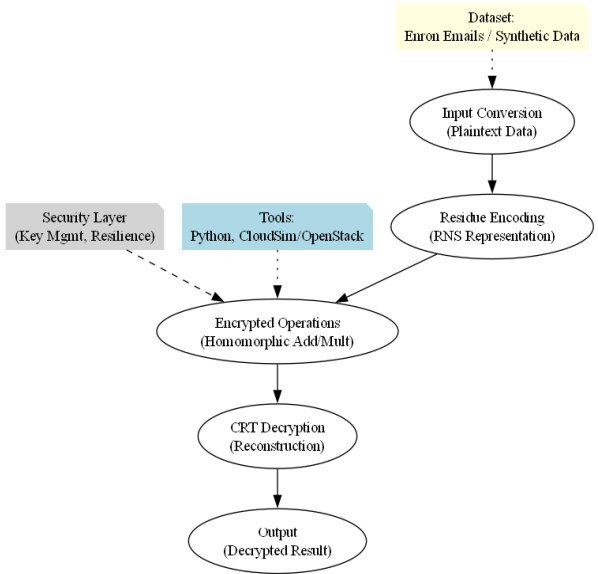


Figure 2. The proposed Residue Number System Framework

As shown in Figure 2, the RNS framework defines a robust and distributed architecture toward deriving privacy preserving computation in cloud environments. First of all, we have multiple users: one of them is User A, another one is User B that play different roles in the ecosystem. User A can be a data provider uploading sensitive data from place such as medical records, while User B can be a data consumer or analyst who computes on such data. These users interact with the Cloud Input Gateway which is the front-end interface that is used for input of this data and validating its format and making sure this prior to encryption. It also serves as the first line of defense by ensuring that all the user input gets safely fed into the system.

Key Distribution Server (KDS) plays a critical role in the architecture and securely manages and distributes encryption keys. Programs key generation, renewal and securely delivers to the encryption modules and authorized users. This task is also quite sensitive; thus,

the Security Manager assists the KDS to enforce access control, user authentication, and integrity verification throughout the whole pipeline. In any case, the Security Manager is essential, since it guarantees that only specified users have access or are able to change an encrypted data, and it monitors the contents of all computations to catch discrepancies or potential attacks. Once data is accepted and the keys are in, the Encryption Layer layers on encryption protocols that could be traditional keys like RSA or AES or more advanced ones like Fully Homomorphic Encryption (FHE). The data is then encrypted, and this encrypted data is then passed into the RNS Engine, which is the computational core of the framework. In this work, data is converted into residue representations across multiple residues exploiting the carry free and parallel

processing capabilities of the RNS over a given modulus. It performs this encrypted and modular arithmetic-based processing at high speed and low latency without deciphering, which finds applications in homomorphic operations such as addition and multiplication. Finally, the RNS outputs are sent to Chinese Remainder Theorem Decryption Layer, which recombines the results in the form of residue components to ultimately produce the final output using modular recombination. The system is able to do encrypted computation with no loss of precision and the decryption is efficient and secure. Results are routed through an appropriate Result Output Gateway, which then produces results once decrypted back to requesting users.

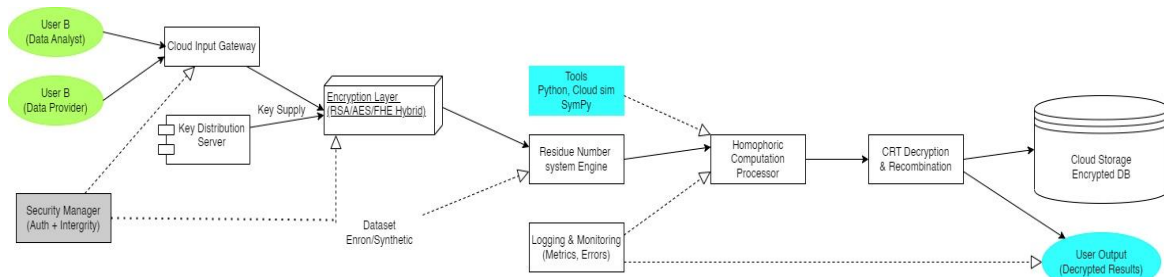


Figure 3. The advanced RNS Framework for Encrypted Cloud Computations

Figure 3 shows the Advanced RNS Framework for Encrypted Cloud Computation describes the flow of data and processes of a modular Residue Number System (RNS) based homomorphic encryption pipeline for secure and efficient operations in cloud environment. Here the architecture shows how sensitive data can be securely ingested, encrypted, computed upon and decrypted without exposing it in plain text at any stage of the computation infrastructure lifecycle to ensure privacy and security as well as capacity and security of computations. The diagram starts with the User Devices that are the source of data from applications, sensors, or a user in a cloud-based platform. First these endpoints are applied with a Data Ingestion Layer in order to prepare the data for secure transformation. This layer can include preprocessing tasks such as filtering out noise, dividing what is segmented, or transforming the data format in the case structured input. The data is then ingested and routed over to the Residue Encoder, an integral part of the RNS system. Given an input, this module divides it into smaller independent residues using a set of pairwise co prime moduli so the data is amenable to parallel and modular arithmetic operations that

capture the RNS. Next, the Modular Encryptor uses optimized moduli chosen for efficiency and security to perform independent modular encryption of each residue using a Modular Encryptor. These encrypted residues are held for a short period of time in the Secure Cloud Storage unit, and are only computed when necessary. Out of here the data moves to Homomorphic Processor that runs encrypted computations like addition and multiplication on the data without decrypting them. The RNS structure efficient and parallel operations for these operations. If the operation is compound or chained, then a specialized Addition/Multiplication Unit is in charge of the modular arithmetic and ensures correctness. Then the encrypted results would be sent to Chinese remainder Theorem (CRT) Decryption module, which reconstructs the original results by using CRT. The output of this is to combine the modular outputs to get something meaningful as a final result. The following step, Results to User, then returns this decrypted output back to the originating user or application.

The diagram highlights an end-to-end secure computation model which takes advantage of the RNS: modularity, parallelism and efficiency. The architecture

of this framework involves interconnecting each step to process the data in an encrypted manner which preserves the data privacy and is suitable for applications such as cloud computing, privacy preserving analytics and secure multi-party computation.

8. Future Work

Future work will then take several approaches to address these challenges. One of the other approaches is to integrate RNS with lattice based cryptographic system which is believed to be resistant to quantum attacks. Frameworks of modular systems that relied on post-quantum primitives (e.g., Learning with Errors) were proposed by [13], and were suggested to be susceptible to adjustments in the context of RNS arithmetic. In another direction, we deploy the framework in practice cloud and edge infrastructures like Kubernetes based environment or containerize services through OpenStack. This would confirm the immunity of the system under dynamic network conditions and realistic workloads.

9. Conclusion

First, in this research, we proposed an RNS and encrypted computation-based framework which is suitable for encrypted computation in clouds, and to attack the issues of conventional encryption models like RSA, and Fully Homomorphic Encryption (FHE). The study through both theoretical analysis and experimental evaluation showed that RNS can provide substantial performance improvements, in reduction of latency and memory consumption as well as computational overhead of encrypted operations. The RNS model is a host of homomorphic properties required for secure cloud computation by leaning on carry free modular arithmetic and parallel residue processing as means of increasing scalability and resisting bottleneck inherent in classical crypto systems. In conclusion, these results endorse the capacity of RNS as a probable constituent of next generation cloud security architectures as the necessity for safe, real time and large loaded solutions keeps on expanding. This RNS based approach concludes being a promising tool for Privacy Preserving computation and a good basis for further research in adaptive modulus selection, post quantum resilience, as well as real world deployment for privacy preserving computation in dynamic, distributed, and hostile settings over the network.

References

- [1] Bajard, J. C., & Imbert, L. (2004). A full RNS implementation of RSA. *IEEE Transactions on computers*, 53(6), 769-774.
- [2] Aithal, G., Hari Bhat, K. N., & Acharya, U. S. (2010). High-speed and secure encryption schemes based on Chinese remainder theorem for storage and transmission of medical information. *journal of mechanics in medicine and biology*, 10(01), 167-190.
- [3] Jia, H., Cai, D., Yang, J., Qian, W., Wang, C., Li, X., & Yang, S. (2023). Efficient and privacy-preserving image classification using homomorphic encryption and chunk-based convolutional neural network. *Journal of Cloud Computing*, 12(1), 175.
- [4] Kadri, A. F. (2023). *Enhancement of Advanced Encryption Standard Performance on Hidden Data Using Residue Number System* (Doctoral dissertation, Kwara State University (Nigeria)).
- [5] Karakaya, A., & Ulu, A. (2024). A survey on post-quantum-based approaches for edge computing security. *Wiley Interdisciplinary Reviews: Computational Statistics*, 16(1), e1644.
- [6] Awadallah, R., Samsudin, A., Teh, J. S., & Almazrooie, M. (2021). An integrated architecture for maintaining security in cloud computing based on blockchain. *IEEE Access*, 9, 69513-69526.
- [7] Saarinen, M. J., & Smith-Tone, D. (2024). Post Quantum Cryptography. In *15th International Workshop, PQCrypto* (pp. 12-14).
- [8] Gupta, K., Chauhan, B., & Gupta, D. (2024). Cloud-based Secure File Storage System using Hybrid Cryptography.
- [9] Gadde, S., Rao, G. S., Veeram, V. S., Yarlagadda, M., & Patibandla, R. S. M. (2023). Secure Data Sharing in Cloud Computing: A Comprehensive Survey of Two-Factor Authentication and Cryptographic Solutions. *Ingénierie des Systèmes d'Information*, 28(6).
- [10] Babatunde, A. N., Balogun, B. F., Lawal, A. O., Akuji, I. I., Kolawole, O. F., Isiaka, O. S., ... & Elelu, A. T. (2025). Application of Residue Number System in Information Security: A Systematic Review.
- [11] Kadri, A. F., Arowolo, M. O., Yusuf-Asaju, A. W., Tajudeen, K. O., & Gbolagade, K. A. (2022). An Overview of Enhancing Encryption Standards for Multimedia in Explainable Artificial Intelligence Using Residue Number Systems for Security. A

Roadmap for Enabling Industry 4.0 by Artificial Intelligence, 247-273.

- [12] Liu, L., & Han, M. (2024). Data sharing and exchanging with incentive and optimization: a survey. *Discover Data*, 2(1), 2.
- [13] Sharma, K., Kumar, N., & Kaushal, R. K. (2023, October). Privacy and Security in Blockchain: A Comprehensive Analysis of Techniques and Threats. In *2023 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)* (pp. 1-6). IEEE.
- [14] Robert, W., Denis, A., Thomas, A., Samuel, A., Kabiito, S. P., Morish, Z., & Ali, G. (2024). A Comprehensive Review on Cryptographic Techniques for Securing Internet of Medical Things: A State-of-the-Art, Applications, Security Attacks, Mitigation Measures, and Future Research Direction. *Mesopotamian Journal of Artificial Intelligence in Healthcare*, 2024, 135-169.