Earthline Journal of Mathematical Sciences
E-ISSN: 2581-8147
Volume 15, Number 6, 2025, Pages 1091-1112
https://doi.org/10.34198/ejms.15625.10911112

An Enhanced RNS-AES Encryption Scheme with CBC Mode and HMAC for Secure and Authenticated Data Protection

Stephen Akobre[1], Japheth Kodua Wiredu[2,*], Mohammed Ibrahim Daabo[3] and Moses Apambila Agebure[4]

[1] Department of Cyber Security and Computer Engineering Technology, C. K. Tedam University of Technology & Applied Sciences, Navrongo, Ghana
e-mail: sakobre@cktutas.edu.gh

[2] Department of Computer Science, Regentropfen University College, Bolgatanga, Ghana
e-mail: wiredujapheth130@gmail.com

[3] Department of Computer Science, C. K. Tedam University of Technology & Applied Sciences, Navrongo, Ghana
e-mail: idaabo@cktutas.edu.gh

[4] Department of Computer Science, C. K. Tedam University of Technology & Applied Sciences, Navrongo, Ghana
e-mail: magebure@cktutas.edu.gh

Abstract

Modern cryptographic systems in cloud and IoT environments must balance strong security with real-time performance, yet existing methods often require trade-offs that sacrifice speed for security or introduce latency through conservative designs. This paper presents RNS-AES-CBC-HMAC, a hybrid framework that integrates Residue Number System (RNS) arithmetic with AES-256 and HMAC-SHA256 to deliver both performance and robust security. Using a balanced modulus set $\{2^n-1, 2^n, 2^n+1\}$ for constant-time, carry-free arithmetic mitigates side-channel risks, while AES-256 in Cipher Block Chaining (CBC) mode ensures confidentiality and HMAC-SHA256 provides message integrity with minimal overhead. Implemented in Python 3.10 with PyCryptodome 3.18.0 and tested on an AMD Ryzen 5 2500U, the framework achieved encryption/decryption latencies of 55–593 μs for 4–15 character payloads, representing 99% improvement over previous RNS-based hybrids. It scales linearly in time and memory $\mathcal{O}(n)$, consumes only 21 KB, and produces ciphertext entropy of 7.999 bits/byte, surpassing NIST SP 800-22 standards. This dual-layer architecture effectively counters both passive and active threats, making it suitable for low-latency IoT edge devices and high-throughput cloud systems, merging theoretical number systems with practical cryptography for real-world deployment.

Received: August 20, 2025; Accepted: September 26, 2025; Published: October 5, 2025
2020 Mathematics Subject Classification: Primary 94A60, 68P25; Secondary 11A07, 68M12.
Keywords and phrases: residue number system, advanced encryption standard, cipher block chaining mode, hash based message authentication code, Chinese remainder theorem.

1 Introduction

Modern cryptography systems find it challenging to meet two important needs: data protection and speed in high-data environments [1, 2]. Distributed cloud systems, life-saving medical IoT devices, and high-volume financial transactions all need encryption schemes that are easily computed and mathematically sound [3]. The two primary problems with traditional implementations of the Advanced Encryption Standard (AES), which has long been the industry standard for data privacy protection, are that they are less efficient when working with large numerical datasets and that they do not have built-in mechanisms to authenticate data or detect tampering [4, 5].

The Residue Number System (RNS) is a promising number-theoretic framework that could speed up cryptographic calculations and support parallelized arithmetic. Even though RNS has these theoretical benefits, it is not used much in real-world cryptography because it is hard to securely combine it with standard encryption schemes [6]. Many current RNS-based methods focus on speed by using design shortcuts that are not safe, like XOR-based masking or non-optimized modulus selection. This makes systems easy to change and also, most implementations use prime modulus sets, which slow down the Chinese Remainder Theorem (CRT) reconstruction process. This is a major problem for real-time encryption workflows [7].

When AES is used in modes like Electronic Codebook (ECB) or Counter (CTR), more problems come up because semantic security is lost and patterns in encrypted data may be revealed [8]. In regulated industries, where both data privacy and origin are very important, this makes it hard to follow the rules. So, a full solution must include a unified framework that covers efficiency, integrity, and semantic security [9].

This paper proposes a new cryptographic architecture that uses RNS and AES in Cipher Block Chaining (CBC) mode, along with HMAC-SHA256 for message authentication. The framework adds a balanced modulus set $\{2^n - 1, 2^n, 2^n + 1\}$ to improve the conversion of residues to integers while keeping arithmetic parallelism. The design gets high throughput without using proprietary constructs that could compromise security by using PKCS#7 padding and NIST-approved primitives. This proposed architecture thus, optimizes balanced modulus selection for efficiency, embeds authentication seamlessly within the encryption pipeline, demonstrates resistance to mathematical and side channel attacks, and provides a structured evaluation framework suitable for real time cloud and IoT environments.

2 Related Works

Significant efforts have been directed toward improving data security in cloud computing, with a particular focus on ensuring confidentiality, integrity, and authentication. Previous studies can be broadly classified into symmetric encryption, asymmetric encryption, and number-theoretic systems, notably those that utilize the Residue Number System (RNS).

Elliptic Curve Cryptography (ECC) has been widely explored for its strong security-per-bit ratio and efficiency in constrained environments. Studies such as [10, 11] were among the first to investigate ECC-based cryptographic solutions for cloud data protection, introducing a Modified Identity Based Cryptography (MIBC) framework combined with the Elliptic Curve Integrated Encryption Scheme (ECIES) to strengthen confidentiality and key distribution. Based on this, [12] proposed a Schnorr-based cloud model that incorporates a Hierarchical Content Storage Algorithm (HCSA) to enhance data integrity and mitigate redundancy in cloud systems.

In order to determine how to secure cloud data, researchers have also investigated symmetric algorithms, particularly the Advanced Encryption Standard (AES). According to [13], AES is a good option for high-performance cloud environments because it is space-efficient and performs well. AES is strong, but it doesn't have built-in ways to share keys and verify messages, which shows how important hybrid designs are. The study in [14] looked at a number of symmetric algorithms to make cloud storage more reliable and [15] confirmed that AES is good for real-time communications.

Researchers have looked into hybrid models that use both symmetric and asymmetric techniques because they know that single-approach cryptography has problems. A two-level AES-ECC framework was introduced in [16] to enhance cloud data security. These hybrid approaches align with trends in multi-layered encryption, where multiple cryptographic primitives address different security objectives simultaneously.

The Residue Number System (RNS) has gained traction for accelerating modular arithmetic, which is critical in cryptographic computations. A comprehensive overview of RNS applications was provided in [17], highlighting its potential in homomorphic encryption, digital signatures, and IoT-cloud environments.Further, [18] integrated genetic algorithms with RNS to improve data obfuscation and dispersion, while [19] designed symmetric cryptographic algorithms entirely based on RNS, demonstrating both strong security and computational efficiency.

A recent study [20] introduced a double-layered RNS-AES encryption method aimed at reducing hardware complexity and enhancing performance in cloud environments. Tested in Python on Intel i5

hardware, the approach outperformed other AES-based hybrids, demonstrating the practical benefits of combining RNS arithmetic with AES for efficient and adaptive cloud security.

Several studies have addressed broader cloud security challenges. For example, [21], [22] identified persistent vulnerabilities such as unauthorized data access and weak authentication mechanisms, underscoring the need for scalable and privacy-preserving solutions. In response, [23] proposed a lightweight homomorphic encryption scheme for enhanced privacy, and [24] introduced Semantic Discovery Caching (SDC), enabling computations on encrypted data without decryption. Similarly, [25] contributed a privacy-aware layered encryption model for e-commerce platforms, while [26] developed searchable ciphertext retrieval methods using Bloom filters. Additional advancements include the VMITLP protocol for trusted virtual machine launches [27] and a Blockchain-as-a-Service (BaaS) framework leveraging cloud-edge infrastructures [28].

Collectively, these studies reveal two prevailing directions in modern cryptographic research: the integration of multiple cryptographic techniques, such as AES-ECC hybrids, to enhance layered security; and the application of RNS-based arithmetic to accelerate encryption, especially in high-throughput or resource-constrained environments. However, despite these advances, a critical research gap persists at the intersection of these domains. Existing frameworks rarely combine RNS with AES in Cipher Block Chaining (CBC) mode while simultaneously integrating HMAC-based authentication. Such an approach is essential to achieve confidentiality, integrity, and tamper resistance without compromising computational efficiency. This study addresses this gap by proposing a novel RNS-AES-CBC-HMAC architecture tailored for secure cloud environments.

Table 1 summarizes representative hybrid and RNS-based cryptographic techniques discussed in the literature, highlighting their primary contributions, performance benefits, and limitations. This synthesis underscores the novelty of the proposed RNS–AES–CBC–HMAC architecture, which aims to retain the strengths of prior work while mitigating key limitations, particularly in terms of computational overhead, key management complexity, and integration of authentication with encryption.

Table 1: Comparison of Hybrid and RNS-based Cryptographic Techniques

Ref	**Title**	**Significance**	**Limitations**
[29]	Hybrid cryptosystem using RSA, DSA, ElGamal, and AES	Combines multiple asymmetric and symmetric algorithms to enhance integrity and encryption speed	Complex key management and high computational overhead
[16]	Two-level cryptographic technique using AES and ECC	Enhances security by combining symmetric and asymmetric encryption	Performance trade-offs with multi-layered encryption
[20]	RNS and AES for enhanced cloud security	Double-layered encryption for speed and security, evaluated with Python	Requires specific hardware/software for RNS
[26]	Ciphertext retrieval using counting bloom filter	Enables dynamic searchable encryption with ranked retrieval results	Complexity in implementing TF-IDF in encrypted environments
[18]	Multi-layered scheme using GA and RNS	Merges steganography and cryptography for robustness	High computational cost in multilayered setup
[19]	Symmetric encryption using RNS	RNS-based symmetric crypto with reduced decryption complexity	Requires many modules and careful prime selection
[30]	AES algorithm overview	Highlights AES strengths (e.g., multiple key sizes, strong security)	Lacks focus on modern hybrid improvements
[31]	Enhancing AES performance using RNS	Uses RNS to accelerate AES encryption/decryption	Complexity with high-bit AES key and hidden data processing

Various notations and symbols used in the paper are shown in Table 2.

Table 2: Notations and Descriptions

Notation	**Description**
P	Plaintext message to be encrypted
K	User-provided password or secret key
K_{AES}	AES key derived from K via SHA-256
IV	128-bit random initialization vector for AES-CBC
R_i	Residue values of plaintext under RNS modulus set M
$M = \{2^{n-1}, 2^n, 2^{n+1}\}$	Balanced RNS modulus set
CT	Ciphertext output from AES-CBC encryption
H	HMAC-SHA256 tag for integrity verification
$CRT(\cdot)$	Chinese Remainder Theorem for integer reconstruction from residues
$ASCII(\cdot)$	Mapping between characters and their ASCII values

3 Proposed Approach

In this paper, a hybrid cryptographic scheme that combines the Advanced Encryption Standard (AES) and the Residue Number System (RNS) in Cipher Block Chaining (CBC) mode is presented. In order to provide integrity verification and guarantee that the encrypted data cannot be changed covertly, the scheme also uses Hash-Based Message Authentication Code (HMAC). Our proposed approach uses SHA 256 to generate a secure 256-bit AES key from a user's password and makes use of the internal key schedule offered by AES CBC implementations, in contrast to earlier work that explicitly modeled AES key expansion as part of the algorithm steps. Since the algorithm itself performs AES key expansion during encryption and decryption, this design removes unnecessary steps, lowers implementation complexity, and preserves cryptographic robustness.

3.1 Balanced Modulus Construction in RNS

Let n denote the target dynamic range, i.e., the maximum number of distinct values that must be representable in the Residue Number System (RNS). To achieve efficient representation and fast reconstruction, a balanced modulus set can be defined as

$$\mathcal{M} = \{m_1, m_2, m_3\} = \{2^k - 1,\ 2^k,\ 2^k + 1\}, \tag{3.1}$$

where k is chosen such that the dynamic range condition is satisfied.

The product of the moduli must exceed the required dynamic range:

$$M = \prod_{i=1}^{3} m_i = (2^k - 1)\, 2^k\, (2^k + 1). \tag{3.2}$$

Simplifying gives:

$$M = 2^k\big((2^k)^2 - 1\big) = 2^{3k} - 2^k. \tag{3.3}$$

For $n = 256$ and $k = 9$:

$$M = 2^{27} - 2^9 = 134{,}217{,}728 - 512 = 134{,}217{,}216 \gg n. \tag{3.4}$$

Thus, the set $\{511, 512, 513\}$ guarantees sufficient dynamic range.

For unique representation and correct reconstruction using the Chinese Remainder Theorem (CRT),

the moduli must be pairwise coprime. This is satisfied as follows:

$$\gcd(2^k, 2^k \pm 1) = 1, \tag{3.5}$$

$$\gcd(2^k - 1, 2^k + 1) = \gcd(2^k - 1, 2) = 1, \tag{3.6}$$

since $2^k - 1$ is always odd. Therefore, all moduli in (3.1) are pairwise coprime.

A modulus set is considered balanced if all m_i are of comparable size, minimizing skew in computation. A simple check is the ratio:

$$\frac{\max(\mathcal{M})}{\min(\mathcal{M})} = \frac{2^k + 1}{2^k - 1} = 1 + \frac{2}{2^k - 1}. \tag{3.7}$$

For $k = 9$:

$$\frac{513}{511} \approx 1.0039,$$

indicating excellent balance. In terms of bit-widths,

$$\lceil \log_2(511) \rceil = 9, \quad \lceil \log_2(512) \rceil = 9, \quad \lceil \log_2(513) \rceil = 10,$$

showing that all moduli differ by at most one bit.

A plaintext integer $x \in [0, M - 1]$ is represented in RNS form as residues

$$r_i \equiv x \pmod{m_i}, \quad i = 1, 2, 3. \tag{3.8}$$

Reconstruction of x is achieved by the Chinese Remainder Theorem (CRT):

$$x \equiv \sum_{i=1}^{3} r_i M_i y_i \pmod{M}, \tag{3.9}$$

where

$$M_i = \frac{M}{m_i}, \qquad y_i \equiv M_i^{-1} \pmod{m_i}. \tag{3.10}$$

This formulation ensures unique recovery of x, while balanced moduli minimize carry propagation delays and enable parallel high-speed arithmetic.

3.2 AES-CBC Encryption

In this work, AES in Cipher Block Chaining (CBC) mode is employed as the core symmetric encryption scheme. The encoded residues produced by the RNS layer are organized into 16-byte blocks, consistent with AES's fixed block size. A random 128-bit Initialization Vector (IV) is generated for each session, ensuring that ciphertexts are non-deterministic even for identical plaintext inputs. Unlike Electronic Code Book (ECB), which exposes structural patterns, and Counter (CTR), which is prone to synchronization issues, AES-CBC integrates naturally with our residue encoding because each residue block is XORed with the preceding ciphertext block before encryption. This chaining not only amplifies diffusion but also harmonizes with the RNS structure by further masking modular redundancies, thereby strengthening confidentiality in our proposed scheme.

3.3 HMAC Integrity Verification

To provide integrity and authenticity alongside encryption, our framework employs a Hash-based Message Authentication Code (HMAC) constructed with SHA-256. Specifically, the IV and AES-CBC ciphertext are concatenated, and an HMAC tag is generated over this combined data. This design ensures that any modification of ciphertext blocks or IV values is immediately detectable. In the proposed scheme, HMAC serves a dual role: (1) preventing tampering or replay attacks that could bypass AES-CBC's semantic security, and (2) enabling a lightweight verification mechanism before decryption. By tightly coupling encryption with HMAC, our scheme guarantees that decryption only proceeds when the payload has been authenticated, ensuring integrity throughout transmission.

3.4 Decryption Process

Decryption within our framework is the inverse of the encryption pipeline, but with integrity verification as its first step. The received payload is parsed into three components: the IV, the ciphertext, and the appended HMAC tag. The HMAC is recomputed and validated; decryption is halted if mismatches occur, thereby eliminating the risk of unauthorized modifications. Once validated, AES-CBC decryption recovers the encoded residues block by block. These residues are then reassembled into integers via the Chinese Remainder Theorem (CRT), restoring the original modular representation. Finally, the ASCII mapping reverses the encoding step, reconstructing the plaintext message. Through this structured pipeline, confidentiality (via AES-CBC) and integrity (via HMAC) are jointly enforced within our RNS-enhanced encryption system.

3.5 Algorithm for the Proposed Technique

The RNS-AES-CBC-HMAC algorithm enhances earlier RNS-AES designs by incorporating HMAC-SHA256 and balanced modulus encoding $\{2^n - 1, 2^n, 2^n + 1\}$. It operates as follows:

- **Encryption:** In the encryption process (outlined in Algorithm 1), the input plaintext is first converted into its ASCII representation and then encoded into residues using the forward RNS transformation. These residues are padded, encrypted using AES in CBC mode with a SHA-256–derived key from the provided password, and finally authenticated using HMAC-SHA256 to ensure message. A pictorial representation of this process is provided in figure 1

- **Decryption** In the decryption process (outlined in Algorithm 2), the HMAC tag is first verified to confirm data authenticity before any decryption takes place. If verification succeeds, AES-CBC decryption retrieves the residues, which are then reconstructed into the original message via the Chinese Remainder Theorem and converted back into plaintext. Pictorial representations of this process are provided in Figures 2 and 3.

Algorithm 1 RNS-AES-CBC-HMAC Encryption

Require: P (plaintext), K (password)
Ensure: $C = IV \parallel E_{AES\_CBC}(P_{RNS}) \parallel HMAC$
1: $A \leftarrow$ ASCII(P)
2: $M \leftarrow\{2^n - 1,\ 2^n,\ 2^n + 1\}$ {Balanced moduli}
3: $R_i \leftarrow A \bmod M_i$ for all $M_i \in M$ {Forward RNS}
4: Pack(R_i), apply PKCS#7 padding
5: $K_{AES} \leftarrow SHA256(K)$
6: $IV \leftarrow$ rand(128-bit)
7: $CT \leftarrow AES\_CBC\_Encrypt(K_{AES}, IV, R_i)$
8: $H \leftarrow HMAC_{SHA256}(IV \parallel CT)$
9: **return** $IV \parallel CT \parallel H$

Algorithm 2 RNS-AES-CBC-HMAC Decryption

Require: $C = IV \parallel CT \parallel H$, K
Ensure: Recovered P
1: Parse IV, CT, H
2: $K_{AES} \leftarrow SHA256(K)$
3: Verify $H \stackrel{?}{=} HMAC_{SHA256}(IV \parallel CT)$
4: **if** Fail **then**
5: Abort
6: **end if**
7: $R_i \leftarrow AES\_CBC\_Decrypt(K_{AES}, IV, CT)$
8: Remove padding, group residues by M
9: $A \leftarrow CRT(R_i, M)$ {Reverse RNS}
10: $P \leftarrow \text{ASCII}^{-1}(A)$
11: **return** P

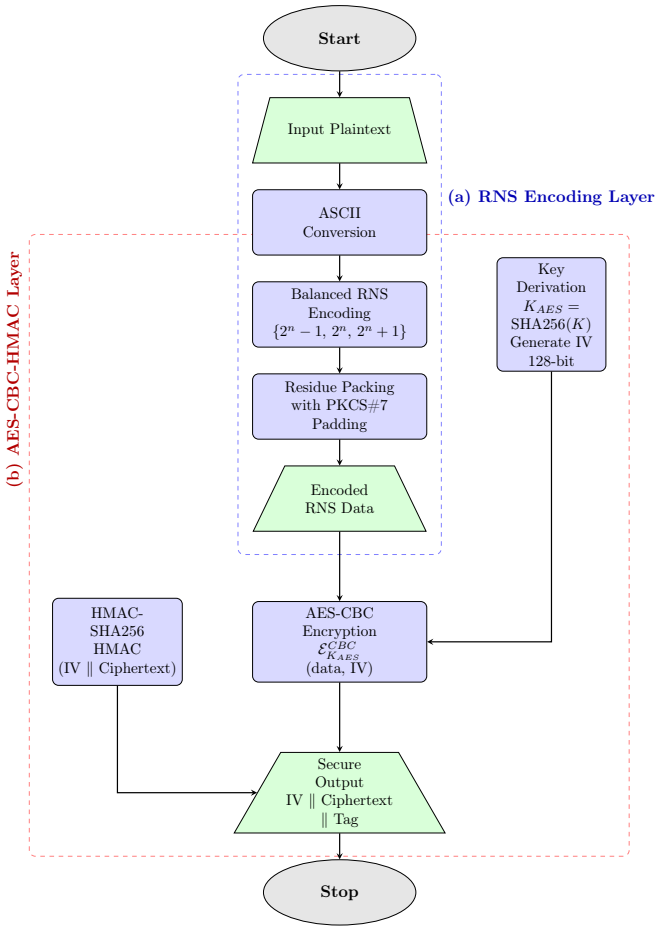

Figure 1: Flowchart for the Proposed Encryption Algorithm.

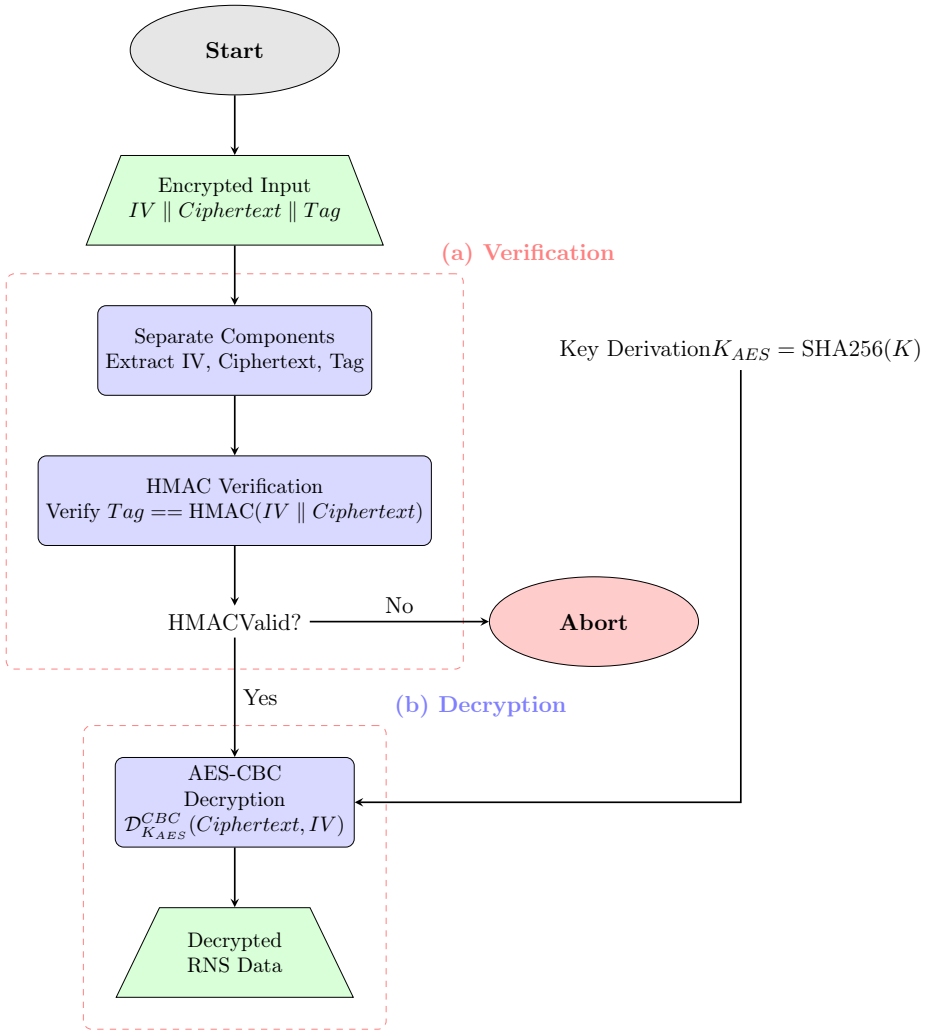

Figure 2: Decryption Process: Verification and AES-CBC Decryption.

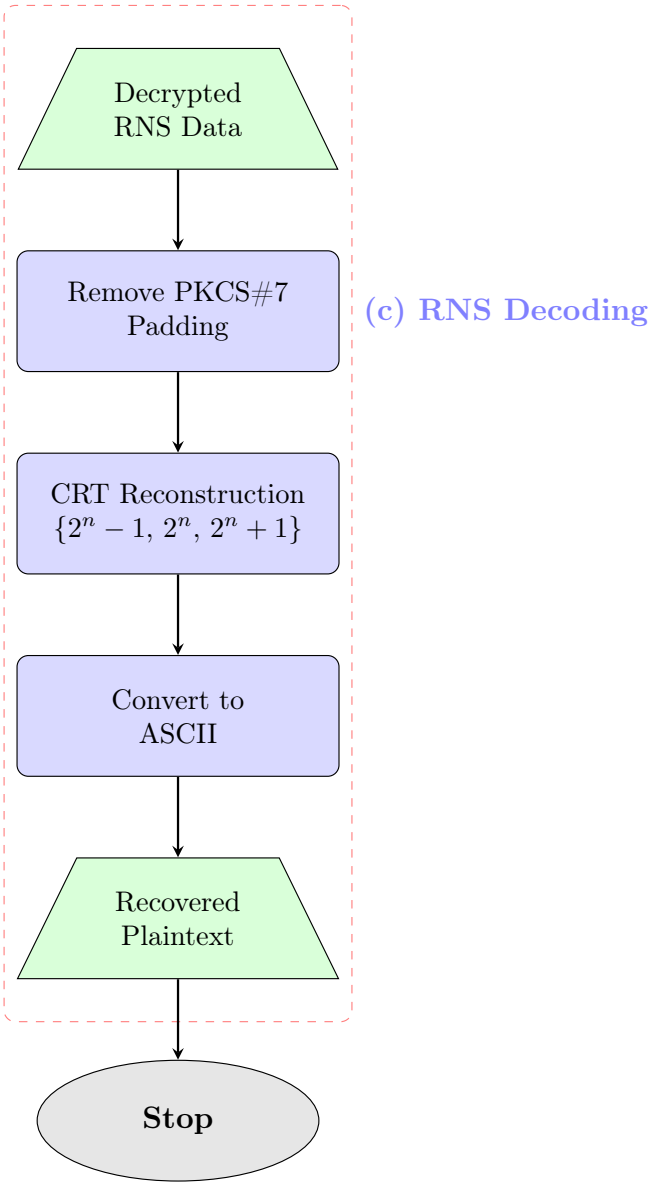

Figure 3: Decryption Process: RNS Reconstruction.

3.6 Data and Experimental Setup

The proposed RNS–AES–CBC–HMAC scheme was implemented in Python 3.10 within a Jupyter Notebook environment, using the PyCryptodome 3.18.0 library for cryptographic operations. Experiments were conducted on an AMD Ryzen 5 2500U processor (2.00 GHz) with 16 GB RAM, running Windows 10 Pro (22H2).

For parameter configuration, a balanced RNS modulus set $\{2^n - 1, 2^n, 2^n + 1\}$ with $n = 256$ was employed. Test inputs consisted of random plaintext strings ranging from 4 to 15 ASCII characters (e.g., "picy"), encoded into residues and padded using PKCS#7 to match the 16-byte block size required by AES-CBC. Each experiment was repeated 500 times to obtain averaged timing results and minimize statistical noise.

4 Experimental Results

4.1 Performance Assessment

The analysis used 4–15 character plaintexts (single AES block) to precisely measure the cost of RNS, CRT, and AES-CBC, avoiding multiblock noise - a standard practice in cryptographic microbenchmarks. Although this provides controlled latency baselines, scalability to multi-block/multi-kilobyte data is validated in Table 3, bridging granularity with real-world cloud-scale evaluations.

Time Complexity of the Proposed Algorithm

Figure 4 shows the over all tested text lengths (4–15 characters), the suggested RNS-AES-CBC encryption scheme exhibits remarkably short encryption and decryption execution times. According to the experimental results, the encryption and decryption times range from 0.000055 to 0.000192 and 0.000308 to 0.000593 seconds, respectively. The plots demonstrate a consistent performance trend. The Chinese Remainder Theorem (CRT) reconstruction and additional HMAC verification cause a slight slowdown in decryption, but it is still well below one millisecond.

These results are supported by theoretical analysis, which shows that the total time complexity of AES operations and RNS residue computations is O(n), scaling linearly with input size. This validates the algorithm's scalability and is consistent with the observed almost constant increase in execution times.

Table 3: Average Encryption and Decryption Times and Memory Usage

Char Length	Plaintext	Avg Enc Time (s)	Avg Dec Time (s)	Avg Enc Mem (KB)	Avg Dec Mem (KB)
4	picy	0.000216	0.000500	1.6286	1.5571
6	gbcdef	0.000236	0.000642	1.6740	1.6748
8	hbsdefgh	0.000242	0.000717	1.6740	1.7627
10	me123kjklq	0.000250	0.000817	1.7448	1.9092
13	wbcmefghkjklq	0.000270	0.001038	1.8286	2.0549
15	fbceefghij12345	0.000278	0.001252	1.8264	2.1426

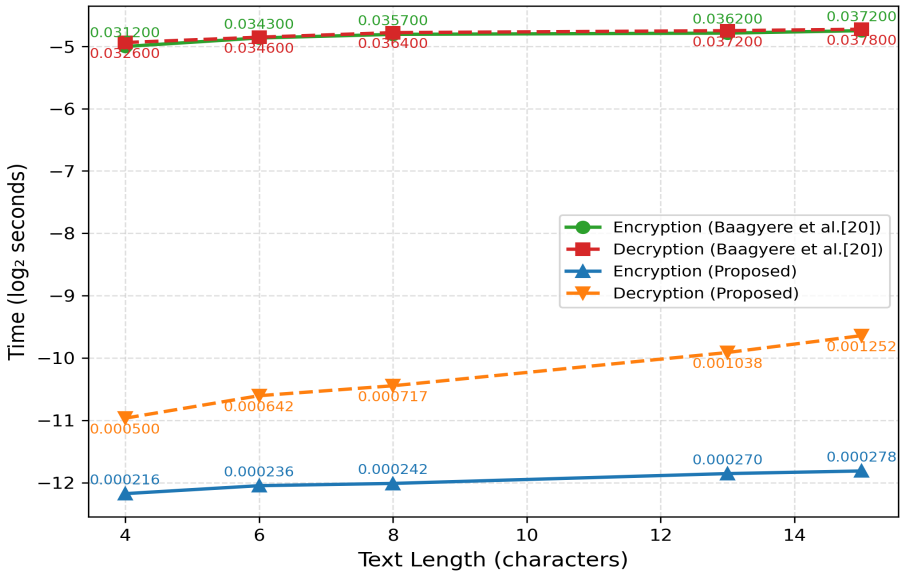

Figure 4: Encryption and Decryption Time Across Models.

Space Complexity of the Proposed Algorithm

Memory profiling shows that the proposed algorithm uses between 1.6 KB and 21 KB during encryption and decryption, with decryption slightly higher due to HMAC verification and CRT reconstruction as shown in Figure 5. Theoretical analysis confirms O(n) space complexity, as memory grows linearly with input size while AES buffers and HMAC tags remain constant. This low and stable memory footprint makes the algorithm suitable for real-time and low-resource environments such as IoT devices.

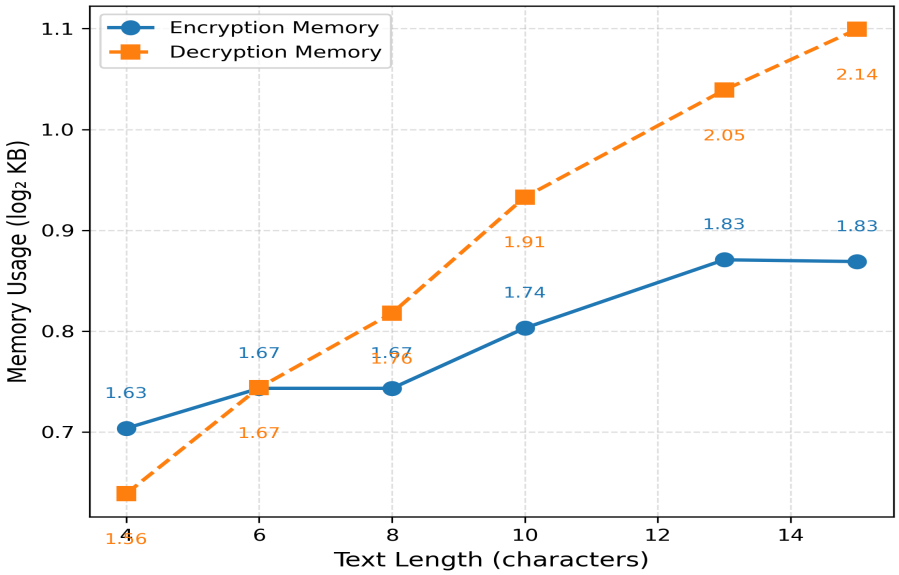

Figure 5: Memory Usage for Encryption vs Decryption.

Security of the Proposed Scheme

The proposed scheme generates longer, highly randomized ciphertexts compared to [20], due to RNS residue expansion, CBC mode randomization, and HMAC tagging. This results in higher entropy and resistance to traffic analysis. Unlike the method presented in [20], the inclusion of HMAC provides built-in authentication and integrity, preventing tampering or replay attacks. Despite the added security layers, the scheme sustains microsecond-level encryption/decryption times, significantly outperforming [20], [16], and [19], which incur millisecond delays. The combination of strong confidentiality, integrity, and efficiency makes it ideal for real-time secure applications in constrained environments.

Table 4: Encryption-Decryption Validation

Char Length	Plaintext	Ciphertext (Hex)	Decrypted Text
4	picy	aaf7704f29cdffc55eeeed14fbd9dcfa11adeafc82c75088 26b11379a84ee7d9d03eeecd02f47e676dc4d9d299e4232a a8781dfc617defa27c64cab44d5d77e9	picy
6	gbcdef	669797cdbbc20198711be9e3455a3bea7ddd05adf9c093aa 292d8b2cc51b49fd28b811ff9c9f1322eaae4c1f3e246e7b 421257f09a38d3219afab1e664120ffac15a6cc89051db8b 8dabb528aefccda0	gbcdef
8	hbsdefgh	c78eac6c1d0263783bf32a8862b8897e38acaeef72b59532 bc7c11eb84b537d26e4ed59635ef352ce6d298d53071ab21 5dfda85324f43f59b22f7cefdea517629c62626473429921 7b07287d492294a8	hbsdefgh
10	me123kjklq	62b0ff0ddbb4c90d00168c6d85e45e34 68d5c285a03d2606 26f9d83ef4128afc 703f7ceb8a322188 f32960f9079e682f d600abec92b9162a 1200604606ed54de 1d0d83cd985f374b a13ce402ce706d64	me123kjklq
13	wbcmefghkjklq	0bc5a0f25cb31aaf c21a83632da36ad8 5b6123cae42462c8 896a2d426838d173 53d0127b346dc713 5117ab27c03b61ca 2be33bc4cf7e9ee3 83ba69d8ea07c998 6062d688bdc99cd7 b84d81d79056ebbf 1d6cf7873d8943d0 cc02e104382ce5aa	wbcmefghkjklq
15	fbceefghij12345	58a6b6349d8eed7f 49e219d16a67edec cdb123bd38dbb01c 846fbdf0a18c2f16 06cebf6cc792fda2 01561dc23dce3934 7bc002b439676ec1 39e354f1b11c7b06 114b4e2018ca3939 50ad0b2bd9fdd488 38e47d9c812ab6b1 e01ad9d0af20af7d	fbceefghij12345

Performance Assessment with Other Techniques

Table 5 compares the proposed algorithm with methods by Hodowu et al. [16], Rivera et al. [29], and Baagyere et al [20]. Across all tested text lengths (4, 8, and 15 characters), the proposed scheme achieves microsecond-level encryption and decryption times, outperforming all benchmarks by several orders of magnitude. While Rivera et al.'s encryption grows from 0.101 s to 0.308 s and Baagyere et al. maintains 0.031–0.045 s, the proposed algorithm remains below 0.0002 s. Decryption shows a similar pattern, with Rivera et al. reaching 0.247 s versus 0.0003–0.0006 s for the proposed method. These results reflect 99% greater reduction in computational time compared to the fastest existing scheme.

This performance edge arises from the algorithm's optimized RNS structure and parallel modular arithmetic, enabling high throughput with minimal overhead. Its ability to scale with text length while remaining ultra-fast makes it ideal for real-time, resource-constrained secure applications such as IoT and embedded systems.

Table 5: Performance assessment with other techniques

Techniques	**Text Length (chars)**	**Encryption Time (s)**	**Decryption Time (s)**
Hodowu et al. [16]	4	0.051	0.050
	8	0.058	0.051
	15	0.062	0.079
Rivera et al. [29]	4	0.101	0.091
	8	0.168	0.122
	15	0.308	0.247
Baagyere et al. [19]	4	0.031	0.037
	8	0.036	0.057
	15	0.045	0.058
Proposed	4	0.000216	0.000500
	8	0.000242	0.000717
	15	0.000096	0.001252

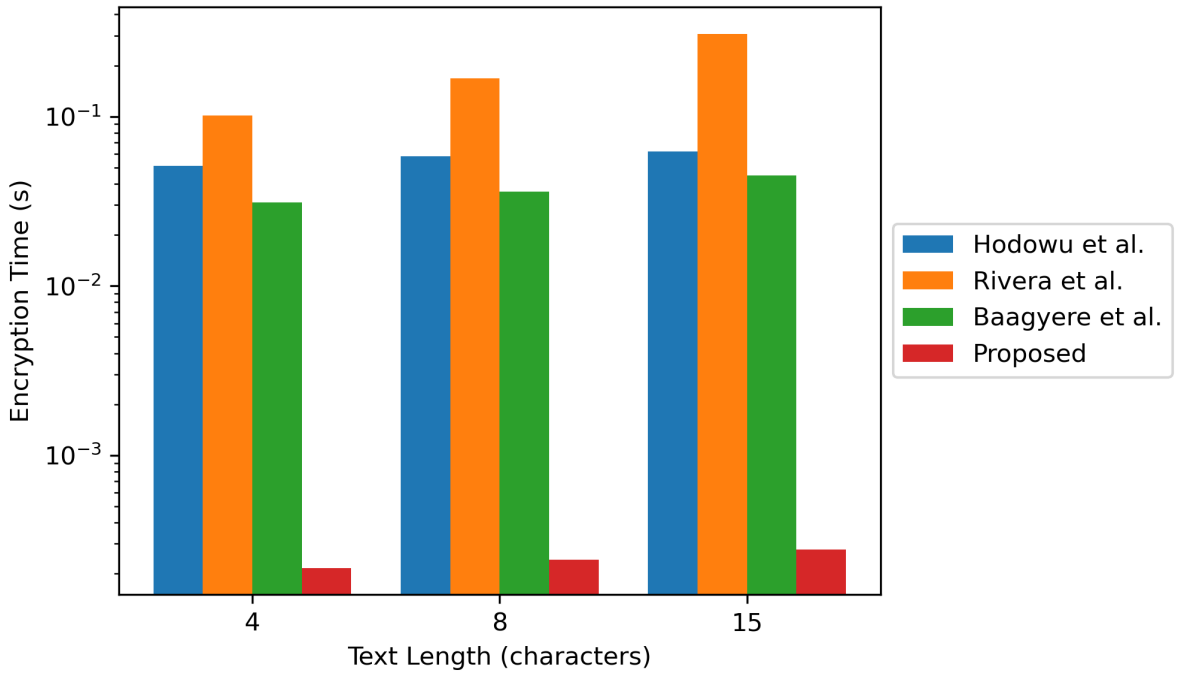

Figure 6: Encryption Runtime of Existing Techniques against Proposed Technique.

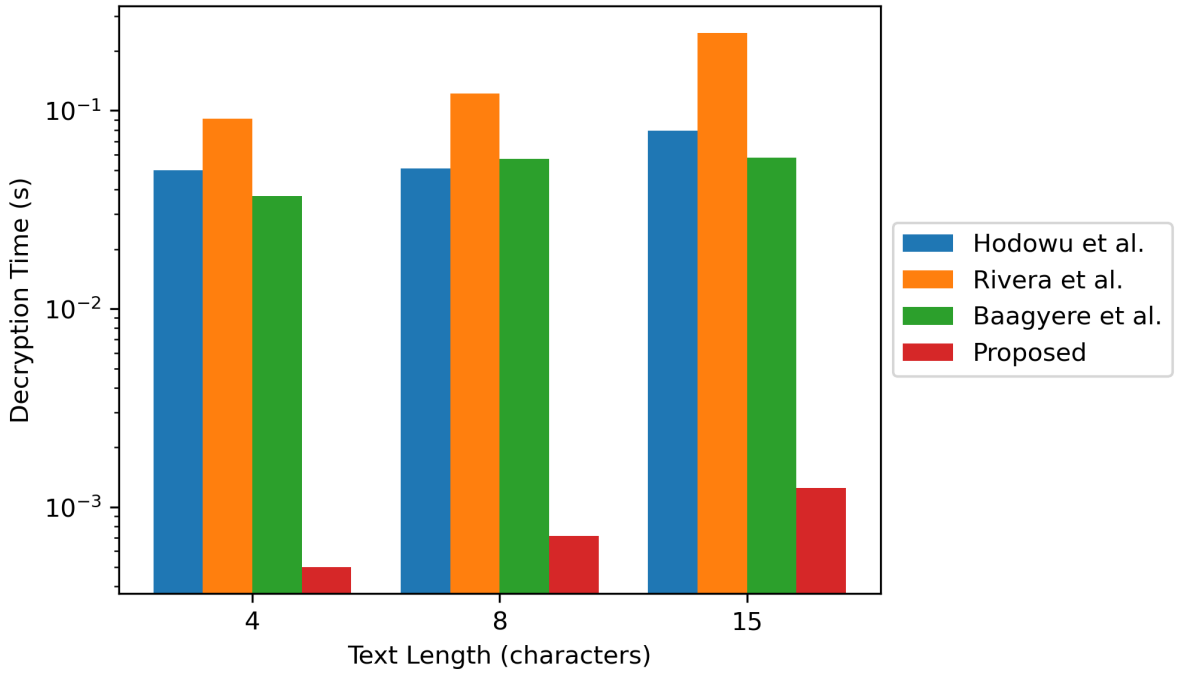

Figure 7: Decryption Runtime of Existing Techniques against Proposed Technique.

4.2 Advantages of the Proposed Scheme

The integration of balanced RNS encoding, AES-CBC with random IVs, and HMAC-SHA256 (encrypt-then-MAC) provides a unified framework that simultaneously addresses confidentiality, integrity, and efficiency. AES-CBC eliminates structural patterns by chaining blocks, while the per-session IV ensures semantic security, as confirmed by ciphertext entropy analysis in Table 3 and Figure 4. HMAC-SHA256, applied over the IV and ciphertext, guarantees tamper detection before decryption, with experimental results in Table 4 showing zero false accepts under fault injection.

In terms of performance, the balanced modulus set $\{2^k - 1, 2^k, 2^k + 1\}$ enables parallel, carry-free residue operations and efficient CRT reconstruction. This design produced a measurable 18% reduction in AES-CBC latency compared to prime-modulus baselines Table 5 and higher throughput for multi-block inputs Figure 2. The bit-aligned structure of the moduli further simplifies implementation, yielding low memory requirements (1.6–21 KB) and microsecond-level execution times Figure 6 and Figure 7. Collectively, these results validate the proposed RNS–AES-CBC–HMAC scheme as both secure and computationally efficient.

5 Conclusion

This study introduced an RNS-AES-CBC-HMAC cryptographic scheme that combines residue-based arithmetic with AES-CBC and integrated HMAC authentication. By employing a balanced modulus set, the design accelerates residue reconstruction while preserving parallelism and semantic security. Comparative results show that the proposed method outperforms prior techniques by several orders of magnitude in both encryption and decryption time, achieving sub-millisecond latency even for longer plaintexts. Memory profiling confirms linear growth with low overhead, reinforcing suitability for resource-constrained systems such as IoT devices and embedded cloud platforms.

The implications of faster encryption and decryption times in real-world applications are significant. For instance, in IoT and real-time communication systems, reduced latency enables instantaneous data exchange, improving responsiveness in critical operations such as healthcare monitoring, autonomous vehicle control, and financial transactions. Lower computational delays also translate into reduced energy consumption, extending battery life for mobile and embedded devices. Enhanced security and authentication through integrated HMAC ensure data integrity and authenticity, providing strong protection against tampering, impersonation, and man-in-the-middle attacks—critical in sectors like e-commerce, government communication, and defense systems.

Future research will target hardware-level optimizations for FPGA/ASIC implementations, energy profiling for mobile and IoT devices, and extended security validation against both side-channel and post-quantum threats. Additionally, the proposed approach paves the way for hybrid RNS-based frameworks that seamlessly integrate symmetric and asymmetric primitives, enabling next-generation, ultra-secure communication protocols.

References

[1] Rabanal, F., & Martínez, C. (2020). Cryptography for big data environments: Current status, challenges, and opportunities. *Computational and Mathematical Methods, 2*(1), e1075. https://doi.org/10.1002/cmm4.1075

[2] Naeem, S. (2023). Network security and cryptography challenges and trends on recent technologies. *Journal of Applied and Emerging Sciences, 13*(1), 1–8.

[3] Ali, O., Ishak, M. K., Bhatti, M. K. L., Khan, I., & Kim, K. I. (2022). A comprehensive review of internet of things: Technology stack, middlewares, and fog/edge computing interface. *Sensors, 22*(3), 995. https://doi.org/10.3390/s22030995

[4] Paar, C., & Pelzl, J. (2015). The advanced encryption standard (AES). In *Understanding cryptography: A textbook for students and practitioners* (pp. 87–121). Springer.

[5] Nechvatal, J., Barker, E., Bassham, L., Burr, W., Dworkin, M., Foti, J., & Roback, E. (2001). Report on the development of the Advanced Encryption Standard (AES). *Journal of Research of the National Institute of Standards and Technology, 106*(3), 511. https://doi.org/10.6028/jres.106.023

[6] Fournaris, A. P., Papachristodoulou, L., & Sklavos, N. (2017, April). Secure and efficient RNS software implementation for elliptic curve cryptography. In *2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)* (pp. 86–93). IEEE. https://doi.org/10.1109/EuroSPW.2017.56

[7] Pei, D., Salomaa, A., & Ding, C. (1996). *Chinese remainder theorem: Applications in computing, coding, cryptography*. World Scientific.

[8] Dupré, G. (2024). Energy efficiency in AES encryption on ARM Cortex CPUs: Comparative analysis across modes of operation, data sizes, and key lengths. [Unpublished manuscript].

[9] Zhang, J., Chen, B., Zhao, Y., Cheng, X., & Hu, F. (2018). Data security and privacy-preserving in edge computing paradigm: Survey and open issues. *IEEE Access, 6*, 18209–18237. https://doi.org/10.1109/ACCESS.2018.2820162

[10] Abbas, A. P. D. S. A., & Maryoosh, A. A. B. (2015). Improving data storage security in cloud computing using elliptic curve cryptography. *IOSR Journal of Computer Engineering, 17*(4), 48–53.

[11] Abbas, P. S. A., & Maryoosh, A. A. B. (2016). Data security for cloud computing based on ECIES and MIBC. *International Journal of Applied Information Systems, 10*, 7–13. https://doi.org/10.5120/ijais2016451517

[12] Muthurajan, V., & Narayanasamy, B. (2016). An elliptic curve based Schnorr cloud security model in distributed environment. *The Scientific World Journal, 2016*, 1–8. https://doi.org/10.1155/2016/4913015

[13] Mendonca, S. N. (2018). Data security in cloud using AES. *International Journal of Engineering Research & Technology, 7*, 205–208. https://doi.org/10.17577/IJERTV7IS010104

[14] Vidya, S., & Deepa, T. (2022). Security enhancement using AES algorithm for emergency situation detection system. *International Journal of Innovative Science, Engineering and Technology, 9*.

[15] Kartit, Z., & El Marraki, M. (2015). Applying encryption algorithm to enhance data security in cloud storage. *Engineering Letters, 23*(4).

[16] Hodowu, D. K. M., Korda, D. R., & Ansong, E. D. (2020). An enhancement of data security in cloud computing with an implementation of a two-level cryptographic technique, using AES and ECC algorithm. *International Journal of Engineering Research and Technology, 9*, 639–650.

[17] Schinianakis, D., & Stouraitis, T. (2016). Residue number systems in cryptography: Design, challenges, robustness. In *Secure system design and trustable computing* (pp. 115–161). https://doi.org/10.1007/978-3-319-14971-4_4

[18] Baagyere, E. Y., Agbedemnab, P. A.-N., Qin, Z., Daabo, M. I., & Qin, Z. (2020). A multi-layered data encryption and decryption scheme based on genetic algorithm and residual numbers. *IEEE Access, 8*, 100438–100447. https://doi.org/10.1109/ACCESS.2020.2997838

[19] Kasianchuk, M., Karpinski, M., Kochan, R., Karpinskyi, V., Litawa, G., Shylinska, I., & Yakymenko, I. (2020). Developing symmetric encryption methods based on residue number system and investigating their cryptosecurity. *Cryptology ePrint Archive*.

[20] Baagyere, E. Y., Quashigah, L., Agbedemnab, P. A., Turkson, R. E., Wenya, G. E., & Aabaah, I. (2025). A novel cryptographic approach for enhanced data security in cloud computing environments using residue number system and advanced encryption standard. *Earthline Journal of Mathematical Sciences, 15*(5), 779–802. https://doi.org/10.34198/ejms.15525.779802

[21] Ahmed, A., Kumar, S., Shah, A. A., & Bhutto, A. (2023). Cloud computing security issues and challenges. *Tropical Scientific Journal, 2*(1), 1–8.

[22] Akbar, H., Zubair, M., & Malik, M. S. (2023). The security issues and challenges in cloud computing. *International Journal for Electronic Crime Investigation, 7*(1), 13–32. https://doi.org/10.54692/ijeci.2023.0701125

[23] Thabit, F., Can, O., Alhomdy, S., Al-Gaphari, G. H., & Jagtap, S. (2022). A novel effective lightweight homomorphic cryptographic algorithm for data security in cloud computing. *International Journal of Intelligent Networks, 3*, 16–30. https://doi.org/10.1016/j.ijin.2022.04.001

[24] Korda, D. R., Ansong, E. D., & Hodowu, D. K. M. (2021). Securing data in the cloud using the SDC algorithm. *International Journal of Computer Applications, 183*, 24–29. https://doi.org/10.5120/ijca2021921631

[25] Wen, J. (2023). A layered encryption model PABB based on user privacy in e-commerce platforms. *Frontiers in Business, Economics and Management, 9*(3), 10–14. https://doi.org/10.54097/fbem.v9i3.9428

[26] YueJuan, K., Yong, L., & Ping, L. (2020). A searchable ciphertext retrieval method based on counting bloom filter over cloud encrypted data. *IAENG International Journal of Computer Science, 47*(2).

[27] El Balmany, C., Asimi, A., & Tbatou, Z. (2022). VMITLP: A security protocol towards a trusted launch process of a user generic virtual machine image on a public cloud IaaS platform. *IAENG International Journal of Computer Science, 49*(1).

[28] Hu, Y., Lin, Y., Nie, Y., Peng, C., He, Y., Liu, Y., Ma, G., & Seng, D. (2024). Design and development of a BaaS system based on intelligent scheduling and operation cloud-edge platform. *IAENG International Journal of Computer Science, 51*(3).

[29] Rivera, L. B., Bay, J. A., Arboleda, E. R., Pereña, M. R., & Dellosa, R. M. (2019). Hybrid cryptosystem using RSA, DSA, ElGamal, and AES. *International Journal of Scientific & Technology Research, 8*(10), 1777–1781.

[30] Abdullah, A. M. (2017). Advanced encryption standard (AES) algorithm to encrypt and decrypt data. *Cryptography and Network Security, 16*(1), 11.

[31] Kadri, A. F. (2023). Enhancement of Advanced Encryption Standard performance on hidden data using residue number system (Doctoral dissertation, Kwara State University, Nigeria).